PORTABLE NAVIGATION UTILIZING SENSOR TECHNOLOGIES IN WEARABLE AND PORTABLE DEVICES

by

Medhat Omr

A thesis submitted to the Department of Electrical and Computer Engineering

In conformity with the requirements for

the degree of Doctor of Philosophy

Queen's University

Kingston, Ontario, Canada

(April, 2015)

Abstract

Portable navigation may seem similar to other navigation problems where system availability, reliability, and accuracy are the objectives. However, more challenges arise due to the portability side of the problem such as having no constraints on the usage of the portable devices and its orientation or position relative to the user or platform. Sensors' cost, computation capabilities, and battery life are additional challenges. An example of some of these challenges is when the designer tries to use more computation power to enhance the accuracy, even if the device's computation capability tolerate this increase, it may shorten the battery life of the device.

This research targets the development of methods to enhance portable navigation for portable and wearable devices in all environments, even in the absence of wireless positioning technologies (like GNSS and WiFi). The enhancement is through benefiting from the presence of multiple connected devices, each including at least one inertial sensor assembly.

This research proposes new methods involving the integration of multiple devices and investigates how it can enhance portable navigation. The proposed methods have a computational demand sufficient to meet battery life constraints for portable devices. In addition, the integrated solution can scale to any number of devices at any point of time. Moreover, the method does not assume any predefined configurations of the devices or relative orientations among them and with respect to each other.

The contributions of this research are achieved through three methods. The first enhances the accuracy of step length estimation. The second method was developed to accurately estimate the misalignment angle between the portable devices and the hosting platform, for example a person while walking or a vehicle while driving. Finally, the overall accuracy of the estimated position and velocity is enhanced while maintaining an efficient computational cost by developing a new method utilizing a new source of Kalman filtering update.

The proposed methods were tested on a large number of real world trajectories collected using several portable/wearable devices during walking, running, and driving. The proposed methods resulted in significantly improving the accuracy of the portable navigation solution.

Acknowledgements

I would like to express my sincere gratitude to my supervisor, Dr. Aboelmagd Noureldin. Dr. Noureldin not only introduced me to the topics of inertial navigation and GNSS, and from whom I learnt a lot, but also supported, encouraged, and guided me throughout my PhD research. He understood me and knew when to help me and when to push me. For his wisdom, he will be one of the most influential people in my life. It was my great pleasure to work under his supervision.

I would like to express my heartfelt gratitude to Dr. Jacques Georgy. I am very fortunate to work under his management, first, during my MITACS Accelerate internship, then during my Natural Science and Engineering Research Council of Canada - Collaborative Research and Development (NSERC - CRD) grant. Both grants were with Trusted Positioning Inc. (TPI), a fully owned subsidiary of Invensense Inc. since its acquisition in August 2014 (now known as Invensense Canada). Dr. Georgy has shared his technical and professional experience with me. He has always been generous in his time and has always entertained my questions. There are no enough words to express my gratitude to Dr. Georgy.

I would like also to thank Dr. Michael Korenberg who introduced me to the topic of Fast Orthogonal Search. As well as thank all my colleagues and friends at the Navigation and Instrumentation (NavINST) research group led by Dr. Noureldin, who were influential in helping me understand the numerous facets of navigation. Dr. Umar Iqbal, Dr. Tashfeen Karamat, Dr. Ali Masoud, Dr. Mohamed Tamazin, Dr. Malek Karaim, Mr. Atif Rahman, and Mr. Ahmed Salah. And special thanks to Dr. Mostafa El-Houshi and Mr. Ahmed Wahdan who helped me a lot and for being my best friends throughout the past three and half years.

For the industrial side of my experience throughout this PhD, and for the financial support, I wish to thank TPI. The work in this thesis was funded through MITACS Accelerate Internship, NSECR CRD, and TPI.

I would like to express my gratitude to TPI team, Dr. Naser El-Sheimy, Dr. Chris Goodall, Dr. Zainab Syed, Dr. Jacques Georgy, Dr. Abdelrahman Ali, Mr. Walid Abd El-Fatah, Dr. Zhi Shen, Mr.

Husain Syed, Mr. James Chan, Mr. David Auld, Ms. Sarah Carmichael, Ms. Alisa Bialas, and everyone else in TPI. Everyone in TPI helped me in one way or another, either in realizing algorithms on consumer products or in collecting data to develop and test the algorithms themselves. All of them have been collaborative with me and I owe them all gratitude for being my colleagues and being one of their team.

Finally, I owe my inmost gratitude to my parents and my brother for their unlimited love, support and encouragement. I am also grateful to my wife Maha and my son Yusuf for their patience, support and continuous encouragement. To my family I dedicate this thesis.

Table of Contents

Abstract ii

Acknowledgements iv

List of Figures xi

List of Tables xvii

List of Abbreviations xviii

1. Introduction 1

1.1 Motivation 1

1.2 Problem Statement 2

1.3 Objectives 3

1.3.1 Accuracy 4

1.3.2 Computation Demand 4

1.3.3 Scalability 4

1.3.4 Ease of Usage 5

1.4 List of Contributions 5

1.5 Thesis Outline 6

2. Background 9

2.1 Introduction 9

2.2 Background 12

2.2.1 Reference Frames 12

2.2.1.1 Earth-Centered Inertial Frame 12

2.2.1.2 Earth-Centered Earth-Fixed Frame 13

2.2.1.3 Local-Level Frame 13

2.2.1.4 The Body Frame 13

2.2.1.5 The Platform Frame 14

2.2.2 Reference Frames Transformations 15
2.2.2.1 Transformation between ECI and ECEF 15
2.2.2.2 Transformation between ECEF and LLF 16
2.2.2.3 Transformation between Body Frame and LLF 16
2.2.2.4 Transformation from Body Frame to ECI and ECEF 17
2.2.3 GNSS 17
2.2.4 Inertial Navigation Systems (INS) 18
2.2.5 Inertial Mechanization 19
2.2.5.1 Attitude Equations 19
2.2.5.2 Velocity Equations 21
2.2.5.3 Position Equations 21
2.2.6 INS/GPS Integration 22
2.2.7 Kalman Filter 23
2.2.8 Example: Loosely Coupled INS\GPS Integratio 25
2.3 Portable Navigation 29
2.3.1 Outdoor Positioning 30
2.3.2 Indoor Positioning 30
2.3.3 Pedestrian Dead-Reckoning (PDR) 31
2.3.4 Hybrid Indoor Positioning Solutions 34
2.4 Multiple IMU's Usage 35
2.5 Nonlinear system identification 40
2.5.1 Artificial Neural Networks (ANN) 41
2.5.2 Fast Orthogonal Search (FOS) 42
3. Experimental Setup 43
3.1 Experimental work strategy 43

3.2 Hardware platform 44
3.3 Software platform 47
3.4 Statistics about personnel involved and data collected 50
4. Step Length Estimation Using Multiple Portable and Wearable Devices 52
4.1 Current State of the art 52
4.2 Step Length Estimation based on Nonlinear System Identification Methods 56
4.2.1 Fast Orthogonal Search (FOS) 56
4.2.2 Why FOS? 60
4.2.3 How FOS practically works? 61
4.2.4 Model Building Phase 70
4.2.5 Model Usage Phase 71
4.3 Using multiple FOS models 71
4.3.1 Using Machine Learning to cluster and classify the training data 76
4.4 Using multiple sensor assemblies 81
4.4.1 Output Level Weighted Average Fusion Architecture 83
4.4.2 Input Level Weighted Average Fusion Architecture 85
4.4.3 Multi-Level Weighted Average Fusion Architecture 86
4.4.4 Centralized Filter Fusion Architecture 87
4.5 Further Enhancements 88
4.5.1 Using Scoring for Weighting 88
4.5.2 Using Dynamic Weighting Scheme 89
4.6 Results 90
4.6.1 Single Device Step Length Experimental results 90
4.6.1.1 Varying step length estimation for Walking with different speeds 90
4.6.1.2 Varying step length estimation for Running with different speeds 93

4.6.2 Multiple FOS Multiple Devices Step Length results 95
4.6.2.1 Three devices results: Smartphone, Smartwatch, and Smart Glasses 96
4.6.2.2 Two devices results 104
4.7 Conclusion 105
5. Misalignment Angle Estimation 108
5.1 Introduction 108
5.2 Enhanced Multiple Devices Misalignment 111
5.2.1 Misalignment Angle Estimation Applications 115
5.3 Results 116
5.3.1 Driving Results 117
5.3.2 Walking Results 121
5.3.3 Effect of Misalignment Estimation on Final Position Solution 126
5.4 Conclusion 130
6. Position and Velocity Aiding 132
6.1 Introduction 132
6.2 Single Device Navigation Solution 134
6.3 Towards Integrated Multiple Sensor Triads' Navigation System 139
6.3.1 Known Fusion Architectures 141
6.3.1.1 Sensor observation fusion architecture 141
6.3.1.2 Centralized filter architecture 142
6.3.1.3 Weighted average architecture 147
6.3.2 Proposed Fusion Architecture 148
6.3.3 3D Position Equations 149
6.3.4 3D Velocity 151
6.3.5 Multiple Sensor Assemblies Constraint 152

6.4 Explanatory Example of the underlying concept for the proposed method 159
6.5 Experimental results 163
6.5.1 Driving Trajectories 164
6.5.2 Walking Trajectories 172
6.6 Conclusion 182
7. Conclusions and Recommendations 184
7.1 Conclusions 184
7.2 Recommendations 186
List of Publications 188
References 190

List of Figures

Figure 1-1: Different components of error and its relationship to contributions 8

Figure 2-1: An illustration of the b-frame and v frame definitions 14

Figure 2-2: A block diagram depicting the mechanization of an INS in the LLF 22

Figure 2-3: Illustration of recursive process of ‘prediction’ and ‘correction’ 24

Figure 2-4: A block diagram of a loosely coupled INS/GPS integration 25

Figure 2-5: PDR 32

Figure 2-6: Step length definition 33

Figure 2-7: Example step detection algorithm 33

Figure 2-8: Some Examples on Today's Wearable and Portable Devices 36

Figure 2-9: The Weighted Average (WA) method architecture 37

Figure 2-10: The sensors observations fusion method architecture 39

Figure 2-11: The centralized filter method architecture 39

Figure 3-1: Experimental work strategy 44

Figure 3-2: Example smartphones, Samsung S3, Samsung S4, and Google Nexus 4 45

Figure 3-3: Example smart watches, Samsung Galaxy Gear Watch 1 & 2 45

Figure 3-4: Example smart glasses 45

Figure 3-5: Example Navigation System Building Blocks 49

Figure 4-1: Step length definition 53

Figure 4-2: Example step detection algorithm 53

Figure 4-3: LN Parallel Cascade 63

Figure 4-4: First Model - Input/Output Sample 64

Figure 4-5: First Model - Input/Output Probability Distribution Functions 64

Figure 4-6: First Model - Output from the Winner Model in the Last Interval 66

Figure 4-7: First Model - Output from the Winner Model in the Last Interval (Zoomed) 66
Figure 4-8: First Model - First Model MSE while adding term by term 67
Figure 4-9: Second Model - Output from Model in the Last Interval 68
Figure 4-10: Second Model - Output from the Winner Model in the Last Interval (Zoomed) 69
Figure 4-11: Second Model - MSE while adding term by term 69
Figure 4-12: Summary of model building phase and model usage phase 71
Figure 4-13: Step Frequency Vs. Previous Step Frequency 72
Figure 4-14: Step Frequency Vs. Previous Step Frequency Vs. Step Length 73
Figure 4-15: Vertical Acceleration Variance Vs. Previous Vertical Acceleration Variance 74
Figure 4-16: Vertical Acceleration Var. Vs. Previous Vertical Acceleration Var. Vs. Step Length 74
Figure 4-17: Step Frequency Vs. Vertical Acceleration Variance 75
Figure 4-18: Step Frequency Vs. Vertical Acceleration Vs. Step Length 75
Figure 4-19: Step Frequency Vs. Previous Step Frequency Vs. Vertical Acceleration Variance . 76
Figure 4-20: Decision Tree Classifier Training Process 78
Figure 4-21: Example decision tree 79
Figure 4-22: Summary of new model building phase and model usage phase after using the previously generated decision tree classifier (a) model building phase (b) model usage phase 80
Figure 4-23: Multi-sensor assemblies step length estimation concept idea 81
Figure 4-24: Outputs-level fusion architecture 84
Figure 4-25: Inputs-level fusion architecture 85
Figure 4-26: Multi-level fusion architecture 87
Figure 4-27: Centralized Filter Fusion Architecture 88
Figure 4-28: Slow walking speed trajectory 92
Figure 4-29: Normal walking speed trajectory 92

Figure 4-30: Fast walking speed trajectory 93
Figure 4-31: Slow running speed trajectory 94
Figure 4-32: Normal running speed trajectory 94
Figure 4-33: Extreme Fast Running speed trajectory 95
Figure 4-34: Trajectories 1-3 98
Figure 4-35: Trajectories 4-6 99
Figure 4-36: Trajectories 7-9 100
Figure 4-37: Trajectories 10-12 103
Figure 4-38: Trajectories 13-15 106
Figure 4-39: Trajectories 16-18 107
Figure 5-1: Misalignment angle definition in both walking (left) and driving (right) 109
Figure 5-2: The proposed misalignment method flow chart 113
Figure 5-3: Example on flow chart different cases 114
Figure 5-4: Emulated smartwatch (left) and smartphone (right) axes definition 118
Figure 5-5: Vehicle Trajectory – Using belt clip as a reference to calculate smartwatch misalignment 119
Figure 5-6: Vehicle Trajectory – Using belt clip as a reference to calculate free smart glasses misalignment 120
Figure 5-7: Vehicle Trajectory – Using belt clip as a reference to calculate free smartphone misalignment 120
Figure 5-8: Emulated smartwatch (left) and smartphone (right) axes definition 122
Figure 5-9: Walking Trajectory 1 124
Figure 5-10: Walking Trajectory 2 125
Figure 5-11: Single IMU vs. Multiple IMU Misalignment 127
Figure 5-12: Single IMU Result 127

Figure 5-13: Multiple IMU Result 128
Figure 5-14: Single IMU vs. Multiple IMU Misalignment 129
Figure 5-15: Single IMU Result 129
Figure 5-16: Multiple IMU Result 130
Figure 6-1: Sensor Observations Fusion Architecture 141
Figure 6-2: Sensor Observations Fusion Overall Performance 142
Figure 6-3: Centralized Filter Architecture 143
Figure 6-4: Centralized Filter Overall Performance 143
Figure 6-5: Number of mathematical operations for states propagation 145
Figure 6-6: No. of mathematical operations for covariance propagation 146
Figure 6-7: No. of mathematical operations for Kalman gain calculation 146
Figure 6-8: Weighted Average Architecture 147
Figure 6-9: Weighted Average Overall Performance 148
Figure 6-10: Proposed Architecture 156
Figure 6-11: Proposed Method Overall Performance 157
Figure 6-12: Two opposite sides biases - WA method 160
Figure 6-13: Two opposite side biases - Proposed method 161
Figure 6-14: Two same side biases - WA method 162
Figure 6-15: Two same side biases - Proposed method 163
Figure 6-16: Vehicle trajectory 1 map 165
Figure 6-17: Vehicle Trajectory 1 – outage map 166
Figure 6-18: Vehicle Trajectory 1 – outage graph 166
Figure 6-19: Vehicle trajectory 2 map 167
Figure 6-20: Vehicle Trajectory 2 – outage map 168
Figure 6-21: Vehicle Trajectory 2 – outage graph 168

Figure 6-22: Vehicle trajectory 3 map – Proposed method solution VS. GPS 170
Figure 6-23: Vehicle trajectory 3 – overlay of all solutions after zoom-in on parkade area 171
Figure 6-24: Vehicle trajectory 4 – smartphone result 171
Figure 6-25: Vehicle trajectory 4 – smartwatch result 172
Figure 6-26: Vehicle trajectory 4 – Proposed method solution 172
Figure 6-27: Walking trajectory 1 – glasses device 174
Figure 6-28: Walking trajectory 1 – belt clip 174
Figure 6-29: Walking trajectory 1 – watch 174
Figure 6-30: Walking trajectory 1 – smartphone 174
Figure 6-31: Walking trajectory 1 – Proposed method solution 175
Figure 6-32: Walking trajectory 2 – glasses device 176
Figure 6-33: Walking trajectory 2 – belt clip 176
Figure 6-34: Walking trajectory 2 – watch 176
Figure 6-35: Walking trajectory 2 – smartphone 176
Figure 6-36: Walking trajectory 2 – Proposed method solution 177
Figure 6-37: Walking Trajectory 3 – smartwatch device 178
Figure 6-38: Walking Trajectory 3 – smartphone device 178
Figure 6-39: Walking Trajectory 3 – Proposed method solution 178
Figure 6-40: Walking Trajectory 4 – smart glasses device 180
Figure 6-41: Walking Trajectory 4 – smartphone device 180
Figure 6-42: Walking Trajectory 4 – smartwatch device 180
Figure 6-43: Walking Trajectory 4 – Proposed method solution 180
Figure 6-44: Walking Trajectory 5 – smart glasses device VS. GPS 181
Figure 6-45: Walking Trajectory 5 – smartwatch device VS. GPS 182
Figure 6-46: Walking Trajectory 5 – Proposed method solution VS. GPS 182

List of Tables

Table 3-1: Model names of portable devices used in data collection 46

Table 4-1: Phone Results 101

Table 4-2: Watch Results 101

Table 4-3: Glasses Results 102

Table 4-4: Multiple Sensors Assemblies' Results 102

Table 4-5: Data set Summary Results 104

Table 5-1: Use case description for each segment in the vehicle trajectory 121

Table 5-2: Use case description for each loop in the walking trajectory 1 123

Table 5-3: Use case description for each loop in the walking trajectory 2 125

Table 6-1: Architecture Preference as a Function of Development Characteristics 158

Table 6-2: Use case description for each loop in the walking trajectory 173

List of Abbreviations

ANN	Artificial Neural Networks
DR	Dead Reckoning
EKF	Extended Kalman Filter
FOS	Fast Orthogonal Search
GNSS	Global Navigation Satellite System
GPS	Global Positioning System
GSM	Global System for Mobile Communications
HMM	Hidden Markov Model
IMU	Inertial Measurement Unit
INS	Inertial Navigation System
KF	Kalman Filter
kNN	k-Nearest Neighbour
LKF	Linearized Kalman Filter
MEMS	Micro-Electro-Mechanical Systems
MLE	Maximum Likelihood Estimator
MMSE	Minimum Mean Square Error
PDR	Pedestrian Dead Reckoning
RBF	Radial Basis Function
RFID	Radio Frequency Identification
WAAS	Wide Area Augmentation System
WLAN	Wireless Local Area Network
ZUPT	Zero-velocity Update

Chapter 1

Introduction

Portable navigation is a forked problem. On one hand, the portable navigation algorithm designer needs to take care of most common portable applications' problems such as: sensors' cost, portable devices' computation capabilities, and battery life. On the other hand, the designer needs to take care of navigation systems' requirements as well, such as: availability, reliability, and accuracy.

From the above mentioned examples of the types of problems which face the navigation algorithms designer, one can see how difficult the portable navigation problem is. For example, in order to enhance the navigation solution accuracy, one approach is using more accurate but expensive sensors, which will affect the overall cost (budget) of the device. Another solution may require more computation power, which has be within the portable device's computation capability. Even if it is within the computation capability limit, this increase in the computation demand will shorten the battery life of the portable device and again make the solution infeasible to be implemented in a real commercial product.

1.1 Motivation

Global Navigation Satellite Systems (GNSS), such as Global Positioning System (GPS), have been widely used for vehicular and outdoor navigation (Kaplan and Hegarty 2006, Grewal, Weill et al. 2007, Jyh Ching and Yu-Hsuan 2009, Zheng, Mingquan et al. 2010). High accuracy is one of the advantages of using GNSS. However, portable navigation is more challenging. Users spend most of their time indoors or in urban canyons, places where GNSS signals suffer from multipath error and/or partial or even complete signal blockage (Georgy, Noureldin et al. 2012,

Groves 2013, Noureldin, Karamat et al. 2013). One solution for providing location services in such challenging environments is using sensors other than GNSS. GSM, Wi-Fi, and Inertial sensors, such as accelerometers and gyroscopes, are good candidates and more preferable in such challenging environments (Liu, Darabi et al. 2007, Gu, Lo et al. 2009).

Smartphones, tablets, wearable computing devices (e.g. smart glasses and smart watches), and even **app**lication ac**cessories** or appcessories for short (e.g. smart belt clip), all have at least one inertial measurement unit (IMU), which includes at least one accelerometer and one gyroscope sensor triad. These sensors are micro electro mechanical sensors (MEMS). Some of the advantages of MEMS are their small size, light weight, low power consumption, and last but not least, low cost. These advantages have caused wide spread integration of MEMS into most of today's portable and wearable devices. However, there are two significant drawbacks of MEMS, high noisy measurements and high random drifts, which make it insufficient for reliable location services.

The average number of portable and wearable devices carried by a user is increasing every day. Smartphones were the first most common portable device, then tablets became prevalent, and now, smart glasses and smart watches have started to increase in popularity. Also, location services now have an increasing number of demanding applications such as: finding the nearest point of interest (e.g. a restaurant or a theater), indoor navigation in an unfamiliar building (e.g. a big shopping center or an airport), and location based gaming. These are all examples of applications which require a good positioning solution to fulfill the necessary level of location awareness.

1.2 Problem Statement

The major challenges related to portable navigation problem includes: (1) the poor accuracy of the low cost sensors used inside portable and wearable devices; (2) the limited computation power and battery life of these devices; (3) the misalignment between the portable device and the moving platform. That is the navigation solution should not affect the usual free

style of user's usage of their portable or wearable devices, for example through forcing some constraint on how to orient the device relative to the user's body or relative to other devices carried by the same user. In other words, the users should remain free to use their devices in any orientation or position relative to their bodies.

Many methods in the literature approached the multiple sensors fusion problem from different angles. Some of them focused on accuracy while ignoring computation power demand. Some other methods did quite the opposite by focusing on the computation power demand and scarifying the accuracy. And a final group of methods took both aspects into consideration while forcing extra constraints such as predefined configurations for the sensor assemblies. Unfortunately, any of these methods is most appropriate for the portable navigation problem.

In this thesis, the multiple sensor fusion problem is approached yet from another different angle, which is trying to balance among reliability, accuracy, computation demand, scalability, and without forcing extra constraints on sensors configurations. All of these previously mentioned aspects are important for the portable navigation problem, and that is why balancing among them make the proposed work the most suitable one for approaching the portable navigation problem.

1.3 Objectives

The ultimate objective of this research is to provide reliable, accurate and continuously available positioning and navigation system using the sensors available in most portable and wearable devices that are widely used by people at the present time. For the scope of this research work, four main objectives were set. These are the accuracy of the portable navigation system, the required computation resources, the scalability of the design, and the ease of usage. These four major objectives are further discussed below.

1.3.1 Accuracy

The first main objective here is providing competitive accurate navigation solution using low cost/inaccurate sensors such as those found in most of nowadays portable or wearable devices. One way to achieve this and overcome sensors' inaccuracy is by exploiting all the multiple sensor triads found in portable or wearable devices connected together, and investigating different fusion architectures.

1.3.2 Computation Demand

The second objective addresses computation demand. First of all, processing units in portable or wearable devices is not as capable as neither those found in personal computers correspondents nor in the specialized navigation processors found in high end Inertial Navigation Systems (INS) systems. In addition, lower computation power demand can be interpreted as longer battery life time. Since battery consumption is one of the major concerns for portable devices applications, a better navigation solution is the one which enjoys competitive accuracy but lower computation power demand. Consequently, it can run in real-time on a computational capable portable device and consume no more energy than any other typical application.

1.3.3 Scalability

Third objective is scalability. Two scalability measures are considered here:

1. Maximum Feasible Number of IMUs: Computational efficiency may degrade on adding more devices to be integrated in the solution. After adding certain number of devices, the solution will not be feasible given the fixed and limited computation capabilities of the host device(s). This number of devices is called the maximum feasible number of IMUs, which can be interpreted as a constraint. One of the objective of this research is to remove this constraint in the proposed navigation solution.

2. Dynamic number of IMUs: Dynamic number here means that the number of IMUs used by the navigation solution may vary through time and not be fixed throughout the entire trajectory. In other words, forcing no constraints on when a device can join or leave the other devices already being used in the solution. For example, the user may start the navigation solution using only two devices, such as, a smart phone and smart glasses. Afterwards, the user can put on a smart watch, at that time the solution should integrate the smart watch smoothly and without any interruption to or intervention from the user.

To wrap up, scalability here can be stated as: any number of devices can be used, and join or leave the solution at any time. That is, the user is free to have any number of portable or wearable device(s) that may change dynamically at any time.

1.3.4 Ease of Usage

The ease of use aim means imposing no constraints on how the user of the portable/wearable devices use, orient, or position their devices relative to their bodies. The user should be able to use all the devices as freely as usual. Since this research is targeting portable devices, then no specific assumption or constraint should be made concerning any specific relative orientation or position between any of the participating devices and the user body, or assuming predefined configurations such as skew redundant IMU as in (Waegli, Guerrier et al. 2008, Yuksel and El-Sheimy 2011). Both of these objectives can be achieved by resolving the misalignment angle between each device and the user.

1.4 List of Contributions

The list of contributions of this thesis is as follows:

1. Development of a novel step length estimation method using nonlinear system identification method such as Fast Orthogonal Search (FOS) and a single inertial sensor assembly (a triad of accelerometers is enough).

2. Development of a more advanced step length estimation method which involves multiple FOS models, machine learning, and a single inertial sensor assembly.
3. Development of a framework which can further enhances any underlying step length and step detection methods using multiple sensor assemblies.
4. Development of a novel misalignment angle estimation method, which uses multiple sensor assemblies, and computationally efficient that enables it from running in parallel with any other misalignment angle estimation method without degrading the computation efficiency, while boosting the reliability and the robustness of the overall misalignment estimation algorithm.
5. Development of a new source of update for Kalman filter, and using it in position and/or velocity aiding using multiple sensor assemblies.

1.5 Thesis Outline

This thesis is divided into seven chapters. The first two chapters after the current one give the necessary background about the portable navigation problem and its underlying techniques, and explains the experimental work strategy that have been followed throughout this research work. Then the following three chapters discusses different proposed methods to improve portable navigation using multiple devices/sensor triads. The first two chapters of them are targeting improving specific components/modules of portable navigation, while the third chapter discusses a new KF source of update to improve the overall error of the integrated solution of multiple devices/sensor triads. Finally, at the last chapter, concluding remarks and recommendations are discussed.

In Chapter 2, a brief introduction to the standard navigation systems and its underlying techniques is presented, then a more focus is given to portable navigation and literature review on

multiple sensors fusion. Finally, the chapter finishes with the necessary background about nonlinear systems identification.

In Chapter 3, the experimental work strategy is described, followed by some details about both hardware and software platforms used during this research, and finishes with some statistics about data collected and personnel involved in this data collection to test and verify the performance of the proposed methods.

Chapter 4 introduces a method which employs nonlinear system identification and a single sensor triad to better estimate step length. Followed by a further enhancement to the previously introduced method using machine learning, more specifically clustering and classification. Then the chapter finishes by another method which involves multiple devices/sensors assemblies to further enhance any underlying single device step length estimation method such as the one introduced in the beginning of that chapter. Enhancing step length estimation is important and its importance can be related to improving along track error as illustrated in Figure 1-1.

Chapter 5 targets another key component of portable navigation which is misalignment angle estimation. The chapter introduces a novel method to estimate the misalignment angle of a device/sensor assembly using multiple devices/sensor assemblies. The method is computationally efficient enough to be implemented and run in parallel with any of the known methods for misalignment angle estimation using a single sensor triad, and which enables it to further enhance the reliability and the robustness of the overall misalignment angle estimation algorithm. Misalignment angle estimation affects most the across track error as illustrated in Figure 1-1, and plays a very important role in solving the portable navigation problem. This was illustrated through the results at the end of that chapter which shows its effect on the final position solution of a device.

Chapter 6 aims at improving the overall navigation solution at a higher more abstract level, so that, whatever improvements which took place at a lower components/level such as the improvements discussed in the previous two chapters. This higher level improvement can further

improves the overall accuracy of the navigation solution regardless which underlying method estimates step length or which one estimates misalignment angle as long as Kalman filter is finally used to estimate the navigation state. That chapter starts by a brief background on the state of the art in sensor fusion architectures in navigation domain. Then a new source of Kalman filter update is introduced. Finally, the chapter finishes with some results for the proposed method and their discussion.

Finally, in Chapter 7, conclusions and recommendations are drawn for future work.

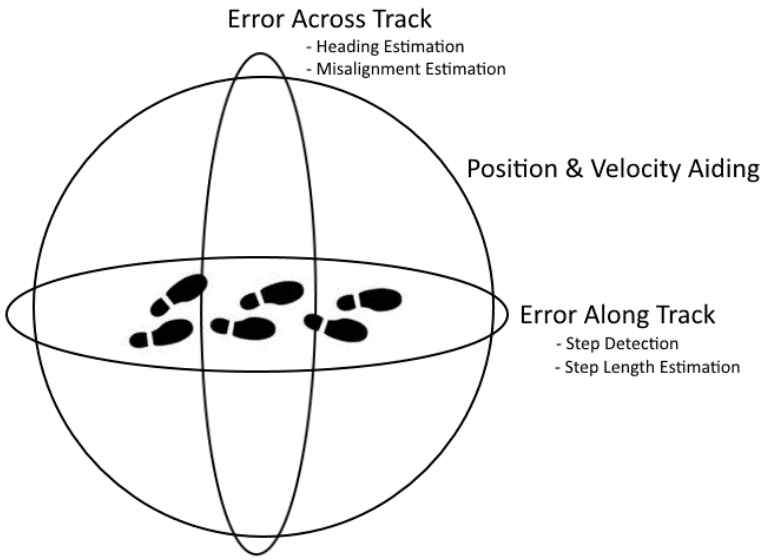


Figure 1-1: Different components of error and its relationship to contributions

Chapter 2

Background

2.1 Introduction

Portable navigation of smartphones and/or smart wearable devices is gaining worldwide attention. With this technology, museums can offer tours with turn-by-turn directions to their visitors, as well as enabling visitors to customize or create their own tours to explore. Retailers can send offers to their customers at the right times and in the right places, as well as collecting valuable data about where their customers spent the majority of their time in the store. All of these scenarios represent the vast range of applications that depend on reliable and accurate portable indoor and outdoor navigation solutions.

Navigation field embraces two concepts. The first is the determination of the position and velocity of a moving platform like: pedestrian, land, marine, or air vehicle, with respect to a known reference frame. And the second is the planning and maintenance of a course from one location to another. This is sometimes known as guidance, pilotage, or routing, depending on the platform type. This thesis is concerned only with the former concept, the position and velocity of a moving platform.

On the other hand, Positioning is the determination of the position of a body, but not its velocity or attitude. Many navigation technologies, though strictly positioning systems, operate at a high enough data rates for velocity to be derived from the rate of change of position.

Most navigation techniques are based on either of two fundamental methods: position fixing and dead reckoning (DR) (Titterton and Weston 2004, Noureldin, Karamat et al. 2013).

The main idea behind position fixing is determining one’s absolute location by measurement of distances from well-known points surrounding the navigator. These points can be

the location of a light house as was the case of sailors in the past, or a radio-based broadcasting station, on land or in the outer space (as in the case of satellites) for the most of nowadays navigation applications.

The most common example for position fixing is Global Navigation Satellite System (GNSS) (Grewal, Weill et al. 2007, Noureldin, Karamat et al. 2013). First, in 1973, the Global Positioning System (GPS) project was developed by the U.S. Department of Defense to overcome the limitations of previous navigation systems, and in 1978 became operational, and in 1994 became globally available and be the first and most utilized satellite navigation system in the world. Second, is the formerly Soviet, and now Russian, Global'naya Navigatsionnaya Sputnikovaya Sistema (or in English: Global Navigation Satellite System), or GLONASS for short. In 1995 it was fully functional. But after the collapse of the Soviet Union, it fell into disrepair, leading to gaps in coverage and only partial availability. In 2011, GLONASS was recovered and restored again. Two more GNSS systems are BeiDou and Galileo, the first one is a Chinese satellite navigation system, while the other one is being built by the European Union (EU), both are still under development.

GNSS provides highly accurate estimates of position and distance traveled. GNSS uses satellites to transmit signals to receivers on the ground. Each GNSS satellite transmits data that indicates its location and the current time. All GNSS satellites synchronize operations so that these repeating signals are transmitted at the same instant. The signals, moving at the speed of light, arrive at a GNSS receiver at slightly different times because some satellites are farther away than others. The distance to the GNSS satellites can be determined by estimating the amount of time it takes for their signals to reach the receiver. When the receiver estimates the distance to at least four GNSS satellites, it can calculate its position in three dimensions.

GNSS receivers, however, require an unobstructed view of the sky, so they are used only outdoors and they often do not perform well within forested areas or near tall buildings. In these situations, an individual using a GNSS is without an estimate of both distance traveled and position.

Conversely, in dead-reckoning, one calculates the current position by using a previously determined position, and advances that position based upon measured or estimated speeds over elapsed time, and attitude (or heading) (Noureldin, Karamat et al. 2013). Two important things to note here is that. DR requires some know initial position, and the second is keep tracking the changes in platform's speed and heading.

In order for DR to work, the navigator needs a way to measure his attitude, and a way to measure the distance moved. Attitude can be measured by a magnetic or electronic compass, or by measuring the rate of change in platform heading using gyroscope sensors (gyros), Integrating these gyros readings over elapsed time gives platform's heading knowing some initial heading value. Distance is determined by double integrating acceleration measured using accelerometer sensors. By integrating the accelerometer sensors readings first, the platform speed is obtained, by further integrating speed, distance traveled is obtained.

The most important advantage about DR is that it is self-contained, which means, in this method, one does not need to communicate with any other type of external device to calculate one's current position. For example, no need to communicate with a satellite, cell phone tower, or Wi-Fi's access point as it is the common case with position fixing techniques. This makes DR more preferable in different indoor scenarios and GNSS-denied environments like forests and urban canyons where GNSS signal suffers a lot from either total blocking or multi-path errors. This navigation system, however, is highly prone to errors, which when compounded can lead to highly inaccurate position and distance estimates.

Therefore, a need exists for a system that integrates the best navigation features of known navigation techniques to provide an individual with estimates of position and distance traveled,

regardless of where they might travel. Usually, a DR system such as Inertial Navigation System (INS) is integrated with one or more position fixing system(s) such as GNSS or Wi-Fi, this is due to the complementary features of each system to each other.

2.2 Background

In this section, first, a general idea on reference frames needed by navigation systems and the transformation from one to another is given, then a brief overview on GNSS and INS as the most popular navigation methods used is presented, finally, a brief summary of GNSS/INS integration techniques and a small example using Kalman filter are given.

2.2.1 Reference Frames

Reference frames are fundamental to describe any navigation state element such as: position, velocity, and attitude. Next are reference frames definitions that are commonly used in navigation field and used in this research.

2.2.1.1 Earth-Centered Inertial Frame

An inertial frame is a reference frame that is not accelerating but may be in a uniform motion. An inertial frame is therefore a reference in which Newton's laws of motion apply. All inertial sensors produce measurements relative to an inertial frame, resolved along the instrument's sensitivity axis.

Earth-Centered Inertial (ECI) is an inertial frame, has its origin at the center of mass of the Earth and axes which are non-rotating with respect to the fixed stars. The z-axis always points along the Earth's axis of rotation. The x-axis is defined as the direction from the Earth to the Sun at the Vernal equinox. The y-axis points 90° ahead of the x-axis in the direction of the Earth's rotation. Both the x-axis and y-axis lie within the equatorial plane, but do not rotate with the Earth. The axes

of the ECI frame are denoted with superscript i as x^i, y^i, z^i, and in this thesis the ECI frame will be referred to as the i-frame.

2.2.1.2 Earth-Centered Earth-Fixed Frame

This frame is similar to the ECI frame because it shares the same origin and z-axis, however, Earth-Centered Earth-Fixed (ECEF) is not an inertial frame since it is Earth-Fixed (i.e. rotating with the Earth). The origin of ECEF is at the center of mass of the Earth. The z-axis is along the Earth's axis of rotation. However, the x-axis lies along the intersection of the plane of the meridian with the Earth's equatorial plane, and the y-axis completes the right-hand orthogonal set (90° east meridian). The axes of the ECEF frame are denoted with superscript e as x^e, y^e, z^e, and in this thesis the ECEF frame will be referred to as the e-frame.

2.2.1.3 Local-Level Frame

Local-Level Frame (LLF) is a local geographic frame which has its origin at the location of the navigation system, and axes aligned with the directions of **N**orth, **E**ast and the local vertical (**D**own) perpendicular to reference ellipsoid and therefore referred to as NED. The axes of the LLF frame are denoted with superscript l as x^l, y^l, z^l, and in this thesis the LLF frame will be referred to as the l-frame.

2.2.1.4 The Body Frame

The body frame (b-frame) is an orthogonal axis set which is aligned with the roll, pitch and yaw axes of the device containing the sensors.

The origin usually coincide with the center of gravity of the device. The x-axis points towards the forward direction. It is also called the roll axis as the roll angle is defined around this axis using the right-hand rule. The y-axis points towards the transverse direction. It is also called the pitch axis, as the pitch angle corresponds to rotations around this axis using the right-hand rule.

Finally, the z-axis points towards the vertical direction down completing a right-handed coordinate system. It is also called the yaw axis as the yaw angle corresponds to the rotations around this axis using the right-hand rule.

2.2.1.5 The Platform Frame

The platform frame (v-frame) is an orthogonal axis set which is aligned with the roll, pitch and yaw axes of the platform within which the navigation capable device is present. The platform can be a vehicle in case of driving scenarios, or the user's body in case of walking or running.

The origin usually coincide with the center of gravity of the platform. The x-axis points towards the forward direction. The y-axis points towards the transverse direction. Finally, the z-axis points towards the vertical direction down completing a right-handed coordinate system. Figure 2-1 shows the difference in definition between the platform frame and the body frame.

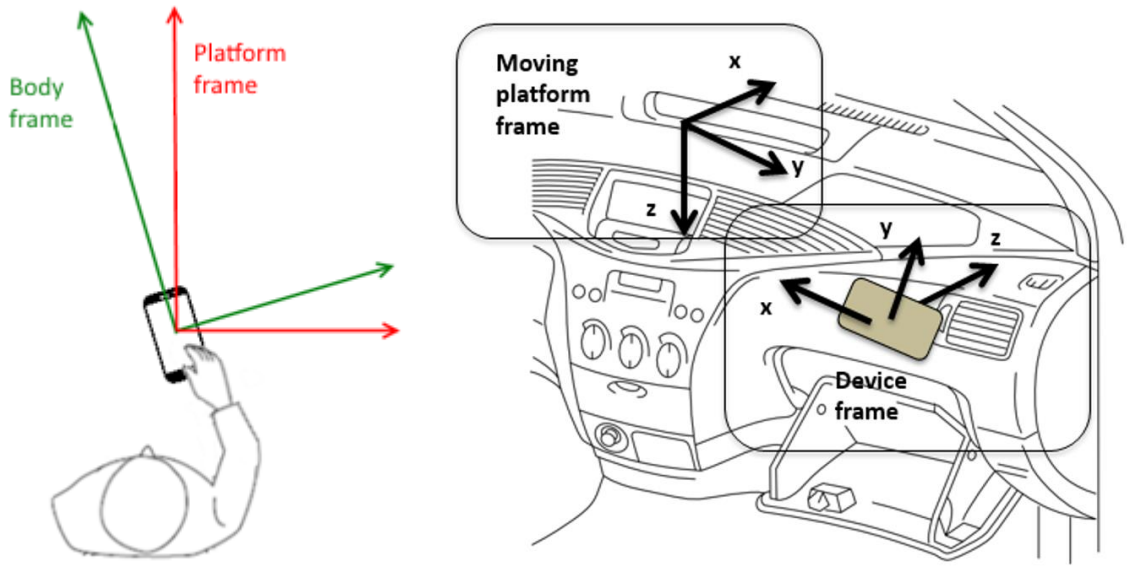


Figure 2-1: An illustration of the b-frame and v frame definitions in both walking (left) and driving (right)

2.2.2 Reference Frames Transformations

The navigation problem usually involves multiple reference frames. For example, inertial sensors produce measurements relative to an inertial frame resolved along the instrument's sensitivity axis. GNSS system measures the position and velocity of a receiver's antenna with respect to a constellation of satellites. However, the user wants to know their position and velocity with respect to ECEF or LLF depending on the application. Therefore, transforming navigation state elements such as position, velocity, and attitude representation with respect to certain reference frame to another is inevitable.

The technique used in the following sub-sections for coordinate transformations is through direction cosine matrices (DCM). The DCM matrix is C_x^y represented by a notation where the subscript represents the coordinate frame from where the vector is transformed and the superscript represents the coordinate of the destination frame. For example, a $\boldsymbol{r}^x$ vector in a coordinate frame x can be represented by another vector $\boldsymbol{r}^y$ in a coordinate frame y through a rotation matrix C_x^y as follows:

$$\boldsymbol{r}^y = C_x^y \boldsymbol{r}^x \tag{2-1}$$

The coordinate transformations described here are a standard in navigation systems, and the following presentation of these transformations is according to (Titterton and Weston 2004).

2.2.2.1 Transformation between ECI and ECEF

The transformation from ECI to ECEF is a simple rotation about the z-axis, where the z-axis is shared between the two frames. The angle of rotation about the z-axis is $\omega_e t$, where ω_e denotes the Earth's rotation rate and t is the time elapsed since the two frames were coincident. The rotation matrix from ECI to ECEF is denoted by C_i^e which is expressed as follows:

$$C_i^e = \begin{bmatrix} \cos\omega_e t & \sin\omega_e t & 0 \\ -\sin\omega_e t & \cos\omega_e t & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2-2}$$

The inverse transformation from ECEF to ECI is done through C_e^i which is the inverse of C_i^e. Since rotation matrices are orthogonal, their inverse is equal to their transpose, thus:

$$C_e^i = (C_i^e)^{-1} = (C_i^e)^T \tag{2-3}$$

2.2.2.2 Transformation between ECEF and LLF

The DCM from the e-frame to the l-frame is expressed in terms of the geodetic latitude φ and longitude λ as:

$$C_e^l = C_y\left(-\varphi - \frac{\pi}{2}\right) C_z(\lambda)$$

$$= \begin{bmatrix} -\sin\varphi\cos\lambda & -\sin\varphi\sin\lambda & \cos\varphi \\ -\sin\lambda & \cos\lambda & 0 \\ -\cos\varphi\cos\lambda & -\cos\varphi\sin\lambda & -\sin\varphi \end{bmatrix} \tag{2-4}$$

The inverse transformation from LLF to ECEF is done through C_l^e which is the inverse of C_e^l. Since rotation matrices are orthogonal, their inverse is equal to their transpose, thus:

$$C_l^e = \begin{bmatrix} -\sin\varphi\cos\lambda & -\sin\lambda & -\cos\varphi\cos\lambda \\ -\sin\varphi\sin\lambda & \cos\lambda & -\cos\varphi\sin\lambda \\ \cos\varphi & 0 & -\sin\varphi \end{bmatrix} \tag{2-5}$$

2.2.2.3 Transformation between Body Frame and LLF

The DCM from the b-frame to the l-frame is expressed in terms of the Euler angles; roll, pitch, and yaw, denoted as r, p, y respectively, as:

$$C_b^l = C_z(-y)C_y(-p)C_x(-r)$$

$$= \begin{bmatrix} \cos y & -siny & 0 \\ \sin y & \cos y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos p & 0 & \sin p \\ 0 & 1 & 0 \\ -\sin p & 0 & \cos p \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos r & -\sin r \\ 0 & \sin r & \cos r \end{bmatrix}$$

$$= \begin{bmatrix} \cos p\cos y & -\cos r\sin y + \sin r\sin p\cos y & \sin r\sin y + \cos r\sin p\cos y \\ \cos p\sin y & \cos r\cos y + \sin r\sin p\sin y & -\sin r\cos y + \cos r\sin p\sin y \\ -\sin p & \sin r\cos p & \cos r\cos p \end{bmatrix} \tag{2-6}$$

2.2.2.4 Transformation from Body Frame to ECI and ECEF

Two other important frame transformations are from body frame to ECEF and ECI frames. The rotation matrices for these transformations can be computed from the previously defined matrices as follows:

For body to ECEF frame:

$$C_b^e = C_l^e C_b^l \tag{2-7}$$

For body to ECI frames:

$$C_b^i = C_e^i C_b^e \tag{2-8}$$

The inverse transformations of the above matrices are given as:

$$C_e^b = (C_b^e)^{-1} = (C_b^e)^T \tag{2-9}$$

$$C_i^b = (C_b^i)^{-1} = (C_b^i)^T \tag{2-10}$$

2.2.3 GNSS

First artificial satellite, Sputnik was launched in 1957 and inspired the navigational and electronics scientist to utilize radio navigation by means of artificial satellites. GNSS are the implementation of this idea and have changed navigation forever. GNSS is used to locate the position of people and places, and to provide navigation information for moving platforms such as ships, aircrafts, and automobiles, anywhere on the surface of the Earth and out to near space. There are currently two operating GNSSs, namely: The Global Positioning System (GPS) and Global Orbiting Navigation Satellite System (GLONASS). GPS consists of a constellation of 24 satellites and some active spares orbiting the earth approximately uniformly dispersed around six circular orbits with four or more satellites each. The geometry of the constellation ensures availability of at least four satellites (the minimum number required for a navigation solution) at any location on, or near the Earth surface (Kaplan and Hegarty 2006). GLONASS also uses 24 satellites, but these are distributed approximately uniformly in three orbital planes of eight satellites each.

Although the solution provided by GNSS is sufficiently accurate (especially when used in differential mode), it is unable to fulfill the requirements of continuity and reliability in some situations. Being a satellite-based navigation system, GNSS requires line-of-sight (LOS) between the receiver antenna and the satellites. However, in the case of a land vehicle, the LOS criterion may not always be met, because a land vehicle typically moves in urban and under dense foliage environments which prevent satellite signals from reaching the antenna. Signal attenuation and even blockage is one of the primary reasons which affects the continuity and reliability of the navigation solution from GNSS. Signal interruption is not the only source of error for GNSS, there are many other problems with GNSS especially in urban and indoor environments such as multipath and interference problems (Grewal, Weill et al. 2007, Noureldin, Karamat et al. 2013), but GNSS errors is out of scope of this thesis.

2.2.4 Inertial Navigation Systems (INS)

In order to obtain continuous and reliable navigation solution, the GNSS is usually combined with another navigation system such as INS. Following is a very brief description of important terminology and building blocks of INS.

The most basic building block of INS is inertial sensors. According to (Noureldin, Karamat et al. 2013), inertial sensors detect and measure motion based on physical laws of nature (specifically Newton's laws) and do not depend on external signals. Accelerometers measure acceleration, while gyroscopes measure rotation rate.

An Inertial Sensor Assembly (ISA) is a unit of inertial sensors in which the raw data from the inertial sensors is the only data output from the unit. On the other hand, an Inertial Measurement Unit (IMU) is an ISA, but the raw data output is compensated for errors like biases and scale factors. Usually, an IMU consists of an orthogonal triad of accelerometers that measures the specific forces

$\mathbf{f}^{\boldsymbol{b}}$ along the Body frame axes, and an orthogonal triad of gyroscopes that measures the rotation rate $\boldsymbol{\omega}_{\boldsymbol{ib}}^{\boldsymbol{b}}$ around the axes of the Body frame.

Likewise, an Inertial Navigation System (INS) is an IMU but with a processing unit and the output from the IMU is fed to a navigation algorithm that evaluates the position, velocity, and attitude of the moving platform. The unit also provides the compensated raw data, which can be used for control or stabilization purposes.

The inertial systems are classified into Strategic Grade, Navigation Grade, Tactical Grade, and Low cost. For the low cost sensors, the most common is MEMS based and which is most common in portable devices, and so, most related to this research.

2.2.5 Inertial Mechanization

Inertial mechanization is the operation of transforming the inertial sensors' readings into position, velocity, and attitude relative to some reference frame. This section describes the inertial mechanization in the local-level frame. Position vector **r** is in the form of latitude, longitude, and altitude. Velocity vector **v** is along North, East, and Down directions of the LLF. Attitude is the roll, pitch, and yaw angles which specify the orientation of the moving platform in the LLF, i.e. it specifies the rotation of the Body frame from the Local-level frame C_b^l.

As mentioned in (Grewal, Weill et al. 2007), mechanization of the IMU readings is a recursive function based only on the new IMU readings and the previous navigation states. Initially, the starting attitude is determined by an initial alignment procedure, the starting velocity and position are given from another instrument such as GPS.

2.2.5.1 Attitude Equations

The attitude (orientation) of the moving body is determined by solving the time derivative equation of the transformation matrix C_b^l (rotation matrix between body frame and LLF). As

explained in (Titterton and Weston 2004) for the LLF mechanization, the differential equation of the transformation matrix is:

$$\dot{C}_b^l = C_b^l \Omega_{lb}^b \tag{2-11}$$

The skew symmetric matrix Ω_{lb}^b of the angular velocity of the body frame with respect to the LLF expressed in the body frame can be expressed as follows:

$$\Omega_{lb}^b = \Omega_{li}^b + \Omega_{ib}^b = -\Omega_{il}^b + \Omega_{ib}^b = \Omega_{ib}^b - \Omega_{il}^b \tag{2-12}$$

Substituting (2-12) in (2-11) yields

$$\dot{C}_b^l = C_b^l (\Omega_{ib}^b - \Omega_{il}^b) \tag{2-13}$$

where

$$\Omega_{il}^b = \Omega_{ie}^b + \Omega_{el}^b \tag{2-14}$$

Since

$$\Omega_{ie}^b = C_l^b \Omega_{ie}^l R_b^l \quad \text{and} \quad \Omega_{\text{el}}^{\text{b}} = C_l^b \Omega_{el}^l C_b^l$$

Therefore

$$\Omega_{il}^b = C_l^b \Omega_{ie}^l C_b^l + C_l^b \Omega_{el}^l C_b^l = C_l^b (\Omega_{ie}^l + \Omega_{el}^l) C_b^l \tag{2-15}$$

Substituting (2-15) in (2-13) yields

$$\dot{C}_b^l = C_b^l [\Omega_{ib}^b - C_l^b (\Omega_{ie}^l + \Omega_{el}^l) C_b^l] \tag{2-16}$$

where Ω_{ib}^b is the skew symmetric matrix of the gyroscopes' readings $\boldsymbol{\omega}_{ib}^b = [\omega_x, \omega_y, \omega_z]^T$

$$\Omega_{ib}^b = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix} \tag{2-17}$$

and Ω_{ie}^l is the skew symmetric matrix of the rotation of the Earth with respect to the inertial frame expressed in LLF $\boldsymbol{\omega}_{ie}^l = [\omega_e \cos\varphi, 0, -\omega_e \sin\varphi]^T$

$$\Omega_{ie}^l = \begin{bmatrix} 0 & -\omega_e \sin\varphi & 0 \\ \omega_e \sin\varphi & 0 & -\omega_e \cos\varphi \\ 0 & \omega_e \cos\varphi & 0 \end{bmatrix} \tag{2-18}$$

and Ω_{el}^{l} is representing the skew symmetric matrix of the turn rate of the LLF with respect to ECEF expressed in LLF $\boldsymbol{\omega}_{el}^{l} = \left[\frac{v_e}{R_N+h}, -\frac{v_n}{R_M+h}, -\frac{v_e \tan\varphi}{R_N+h}\right]^T$

$$\Omega_{el}^{l} = \begin{bmatrix} 0 & \frac{v_e \tan\varphi}{R_N+h} & -\frac{v_n}{R_M+h} \\ -\frac{v_e \tan\varphi}{R_N+h} & 0 & -\frac{v_e}{R_N+h} \\ \frac{v_n}{R_M+h} & \frac{v_e}{R_N+h} & 0 \end{bmatrix} \tag{2-19}$$

2.2.5.2 Velocity Equations

The velocity vector of the moving object is $\mathrm{v}^{l} = [v_n, v_e, v_d]^T$, the velocity equation is:

$$\dot{\mathbf{v}}^{l} = C_b^l \mathbf{f}^{\boldsymbol{b}} - \left(2\Omega_{ie}^{l} - \Omega_{el}^{l}\right)\mathbf{v}^{\boldsymbol{l}} + \mathbf{g}^{\boldsymbol{l}} \tag{2-20}$$

In the above equation, C_b^l is the transformation matrix from the body frame to the LLF, $\mathbf{f}^{\boldsymbol{b}}$ is the specific force measured by the accelerometers in body frame, Ω_{ie}^{l} is the skew-symmetric matrix of angular velocity for the Earth rotation expressed in LLF, Ω_{el}^{l} is the skew-symmetric matrix of the angular velocity vector of LLF with respect to ECEF frame expressed in LLF, and $\mathbf{g}^{\boldsymbol{l}}$ is the gravity vector in LLF.

2.2.5.3 Position Equations

As described in (Titterton and Weston 2004), the time rate of change of the position components has the following relation to the velocity components:

$$\begin{bmatrix} \dot{\varphi} \\ \dot{\lambda} \\ \dot{h} \end{bmatrix} = \begin{bmatrix} \frac{1}{R_M + h} & 0 & 0 \\ 0 & \frac{1}{(R_N + h)\cos\varphi} & 0 \\ 0 & 0 & -1 \end{bmatrix} \begin{bmatrix} v_n \\ v_e \\ v_d \end{bmatrix}^{l} \tag{2-21}$$

$$\dot{\boldsymbol{r}}^{l} = D^{-1}\mathbf{v}^{l} \tag{2-22}$$

In the above equation, D^{-1} transforms the velocity from LLF to the rate of change of ECEF coordinates whereas R_M and R_N are the meridian and normal radii of the Earth's reference ellipsoid respectively.

As a summary, a block diagram of the mechanization in the LLF can be seen in Figure 2-2.

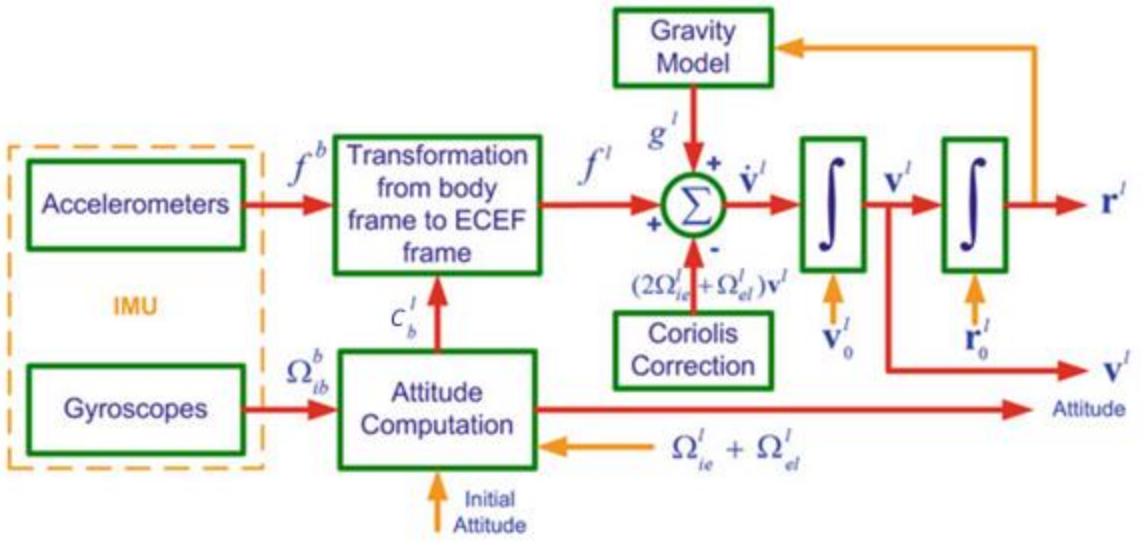


Figure 2-2: A block diagram depicting the mechanization of an INS in the LLF, reprinted with permission from (Noureldin, Karamat et al. 2013)

2.2.6 INS/GPS Integration

The integration operations involved in the mechanization cause unbounded accumulation of errors. Any small uncompensated sensor error will increase without bound after integration. So the mechanization output has short-term accuracy but does not have long-term accuracy because it drifts with time.

Stand-alone GPS provides accurate long term navigation information but may suffer from outages because of signal blockage, interference, or jamming (Grewal, Weill et al. 2007, Noureldin, Karamat et al. 2013). In contrast to GPS, INS has good short-term accuracy and operates independently, without the need for any external signals to provide navigation solution. In order to overcome the disadvantages associated with the stand-alone operation of GPS and INS, the two systems are often paired together in a complimentary fashion so that their drawbacks are minimized or eliminated.

By integrating the GPS and INS signals, a complementary solution can be obtained that is often more accurate than that of each independent system (Grewal, Weill et al. 2007). Integration of navigation systems can be accomplished by various methodologies. The architecture of an INS/GPS integrated navigation system varies in three respects: how corrections are applied to the inertial navigation solution, what types of GNSS measurements are used, and what is the technique and algorithm used for data fusion of GNSS and INS (Noureldin, Karamat et al. 2009).

2.2.7 Kalman Filter

Kalman filter (KF) is one of many estimation techniques used to integrate INS and GNSS. It is an optimal estimation tool that provides a sequential recursive algorithm for the estimation of a system states when the system model is linear (Minkler and Minkler 1993). In addition to its benefits as an optimal estimator, the KF provides real-time statistical data related to the estimation accuracy of the system states, which is very useful for quantitative error analysis (Grewal and Andrews 2011). The filter generates its own error analysis with the computation of the error covariance matrix, which gives an indication of the estimation accuracy.

The KF uses a form of feedback control by which the filter estimates the system state at some time and then obtains feedback in the form of noisy measurements. As such, the equations for the KF fall into two groups: time update or prediction equations and measurement update or correction equations.

The time update equations are responsible for projecting forward in time the current state and error covariance estimates to obtain the a priori estimates for the next step. Equations (2-23) – (2-25) shows the time update equations, where equation (2-23) is called system model, and equation (2-24) shows the state estimate extrapolation (or prediction in the future), while equation (2-25) shows the error covariance extrapolation.

$$\mathrm{x_k} = \Phi_k x_{k-1} + w_{k-1}, \qquad w_k \sim N(0, Q_k) \tag{2-23}$$

$$\hat{x}_{\mathrm{k|k-1}} = \Phi_{\mathrm{k-1}} \hat{\mathrm{x}}_{\mathrm{k-1|k-1}} \tag{2-24}$$

$$\mathrm{P_{k|k-1}} = \Phi_{\mathrm{k-1}} \mathrm{P_{k-1|k-1}} \Phi_{\mathrm{k-1}}^{\mathrm{T}} + Q_{k-1} \tag{2-25}$$

On the other hand, the measurement update equations are responsible for the feedback, i.e. for incorporating a new measurement into the a priori estimate to obtain an improved a posteriori estimate (Welch and Bishop 1995). Equations (2-26) – (2-29) shows a summary of the measurement update equations. Equation (2-26) is called the measurement model, whereas equation (2-27) represents the state estimate update, and equation (2-28) represents corresponding error covariance update. Finally, equation (2-29) shows Kalman gain matrix equation.

$$\mathrm{z_k} = H_k x_k + v_k, \qquad v_k \sim N(0, R_k) \tag{2-26}$$

$$\hat{x}_{\mathrm{k|k}} = \hat{\mathrm{x}}_{\mathrm{k|k-1}} + \mathrm{K_k}[\mathrm{z_k} - \mathrm{H_k}\hat{\mathrm{x}}_{\mathrm{k|k-1}}] \tag{2-27}$$

$$\mathrm{P_{k|k}} = [\mathrm{I} - \mathrm{K_k} H_k]\mathrm{P_{k|k-1}} \tag{2-28}$$

$$K_{k+1} = \mathrm{P_k}(-) H_k^T \left[H_k P_k(-) H_k^T + R_k\right]^{-1} \tag{2-29}$$

Figure 2-3 shows the relation among the different variables involved in KF algorithm.

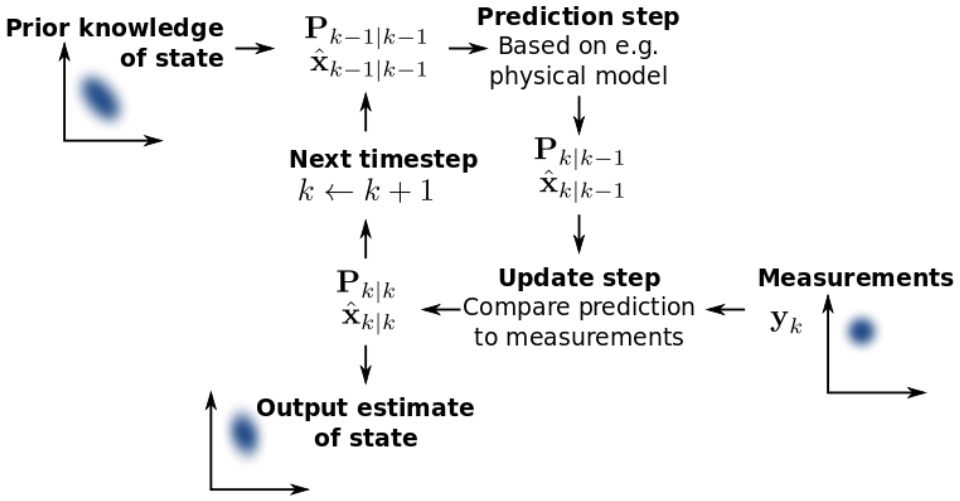


Figure 2-3: Illustration of recursive process of 'prediction' and 'correction' (Aimonen 2014)

The problem at hand (i.e. INS/GPS integration) has nonlinear models. In order to use KF in this integration problem, this nonlinear model has to be linearized around a nominal trajectory. In this problem the linearization can be either around the unaided INS mechanization or around the current estimated solution. The former is used in open loop integration and the KF is called Linearized KF (LKF), while the latter is used in closed loop and the KF is called Extended KF (EKF) (Maybeck 1982). With the assumption of Gaussian distributed noise sources, the minimum mean square error (MMSE) solution to the linear problem is then provided by the KF (Kailath, Sayed et al. 2000). For non-Gaussian-distributed noise sources, the Kalman filter provides the linear MMSE solution to the filtering problem.

2.2.8 Example: Loosely Coupled INS\GPS Integration

In this example, a 15-state KF for integrating INS/GPS is presented. The INS and GPS are loosely coupled, meaning that both operate independently and provide separate navigation solutions. Figure 2-4 shows a block diagram of a loosely coupled INS/GPS integration.

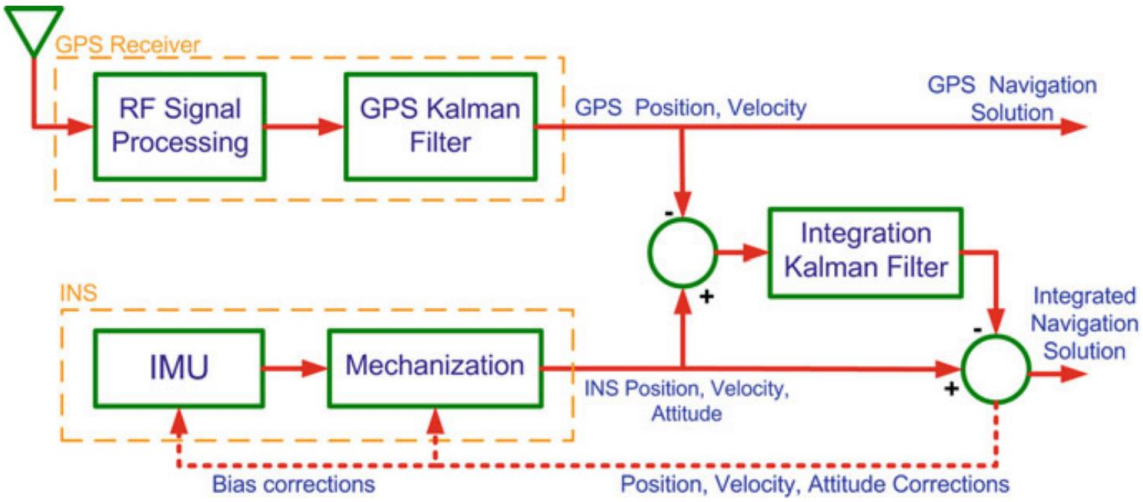


Figure 2-4: A block diagram of a loosely coupled INS/GPS integration, reprinted with permeation from (Noureldin, Karamat et al. 2013)

The KF state vector include error components of position, velocity, and attitude as well as accelerometer biases and gyroscope drifts

$$\delta\mathbf{x}^{l}_{15\text{x}1} = [\delta\mathbf{r}^{l}_{3\text{x}1}, \delta\mathbf{v}^{l}_{3\text{x}1}, \delta\boldsymbol{\varepsilon}^{l}_{3\text{x}1}, \delta\boldsymbol{\omega}_{3\text{x}1}, \delta\mathbf{f}_{3\text{x}1}] \tag{2-30}$$

Where

$\delta\mathbf{r}^{l}_{3\text{x}1} = [\delta\varphi, \delta\lambda, \delta h]^{T}$ Is the position error vector

$\delta\mathbf{v}^{l}_{3\text{x}1} = [\delta v_n, \delta v_e, \delta v_d]^{T}$ Is the Earth-referenced velocity error vector

$\delta\boldsymbol{\varepsilon}^{l}_{3\text{x}1} = [\delta p, \delta r, \delta A]^{T}$ Is the attitude error vector

$\delta\boldsymbol{\omega}_{3\text{x}1} = \left[\delta\omega_x, \delta\omega_y, \delta\omega_z\right]^{T}$ Is the gyroscope error vector (consisting of drifts)

$\delta\mathbf{f}_{3\text{x}1} = \left[\delta f_x, \delta f_y, \delta f_z\right]^{T}$ Is accelerometer error vector (consisting of biases)

The system model equation is:

$$\delta\mathbf{x}_k = (I + F\Delta t)\delta\mathbf{x}_{k-1} + G\Delta t w_{k-1} \tag{2-31}$$

where

$$F = \begin{bmatrix} 0_{3\text{x}3} & F_r & 0_{3\text{x}3} & 0_{3\text{x}3} & 0_{3\text{x}3} \\ 0_{3\text{x}3} & 0_{3\text{x}3} & F_v & 0_{3\text{x}3} & R^{l}_{b} \\ 0_{3\text{x}3} & F_\varepsilon & 0_{3\text{x}3} & R^{l}_{b} & 0_{3\text{x}3} \\ 0_{3\text{x}3} & 0_{3\text{x}3} & 0_{3\text{x}3} & F_\omega & 0_{3\text{x}3} \\ 0_{3\text{x}3} & 0_{3\text{x}3} & 0_{3\text{x}3} & 0_{3\text{x}3} & F_f \end{bmatrix} \tag{2-32}$$

and

$$G = \begin{bmatrix} \sigma_{r,1\text{x}3} \\ \sigma_{v,1\text{x}3} \\ \sigma_{\varepsilon,1\text{x}3} \\ \sigma_{\omega,1\text{x}3} \\ \sigma_{f,1\text{x}3} \end{bmatrix} \tag{2-33}$$

An expanded version of equation (2-31) is:

$$\begin{bmatrix} \delta\varphi \\ \delta\lambda \\ \delta h \\ \delta v_n \\ \delta v_e \\ \delta v_d \\ \delta p \\ \delta r \\ \delta A \\ \delta\omega_x \\ \delta\omega_y \\ \delta\omega_z \\ \delta f_x \\ \delta f_y \\ \delta f_z \end{bmatrix}_k = \begin{bmatrix} 1 & 0 & 0 & \frac{\Delta t}{R_M+h} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & \frac{\Delta t}{(R_N+h)\cos\varphi} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & -\Delta t & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & -f_d\Delta t & -f_e\Delta t & 0 & 0 & 0 & C_{11}\Delta t & C_{12}\Delta t & C_{13}\Delta t \\ 0 & 0 & 0 & 0 & 1 & 0 & f_d\Delta t & 0 & f_n\Delta t & 0 & 0 & 0 & C_{21}\Delta t & C_{22}\Delta t & C_{23}\Delta t \\ 0 & 0 & 0 & 0 & 0 & 1 & f_e\Delta t & -f_n\Delta t & 0 & 0 & 0 & 0 & C_{31}\Delta t & C_{32}\Delta t & C_{33}\Delta t \\ 0 & 0 & 0 & \frac{\Delta t}{R_M+h} & 0 & 0 & 1 & 0 & 0 & C_{11}\Delta t & C_{12}\Delta t & C_{13}\Delta t & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -\frac{\Delta t}{R_N+h} & 0 & 0 & 1 & 0 & C_{21}\Delta t & C_{22}\Delta t & C_{23}\Delta t & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -\frac{\Delta t\tan\varphi}{R_N+h} & 0 & 0 & 0 & 1 & C_{31}\Delta t & C_{32}\Delta t & C_{33}\Delta t & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1-\beta_{\omega x}\Delta t & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1-\beta_{\omega y}\Delta t & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1-\beta_{\omega z}\Delta t & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1-\beta_{fx}\Delta t & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1-\beta_{fy}\Delta t & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1-\beta_{fz}\Delta t \end{bmatrix} \begin{bmatrix} \delta\varphi \\ \delta\lambda \\ \delta h \\ \delta v_n \\ \delta v_e \\ \delta v_d \\ \delta p \\ \delta r \\ \delta A \\ \delta\omega_x \\ \delta\omega_y \\ \delta\omega_z \\ \delta f_x \\ \delta f_y \\ \delta f_z \end{bmatrix}_{k-1} + \begin{bmatrix} \sigma_\varphi \\ \sigma_\lambda \\ \sigma_h \\ \sigma_{vn} \\ \sigma_{ve} \\ \sigma_{vd} \\ \sigma_p \\ \sigma_r \\ \sigma_A \\ \sqrt{2\beta_{\omega x}\sigma_{\omega x}^2} \\ \sqrt{2\beta_{\omega y}\sigma_{\omega y}^2} \\ \sqrt{2\beta_{\omega z}\sigma_{\omega z}^2} \\ \sqrt{2\beta_{fx}\sigma_{fx}^2} \\ \sqrt{2\beta_{fy}\sigma_{fy}^2} \\ \sqrt{2\beta_{fz}\sigma_{fz}^2} \end{bmatrix} \Delta t\, w_{k-1} \tag{2-34}$$

The measurement model is:

$$\delta \mathbf{z}_k^{\cdot} = \begin{bmatrix} \boldsymbol{r}_{INS}^{l} - \boldsymbol{r}_{GPS}^{l} \\ \mathbf{v}_{INS}^{l} - \mathbf{v}_{GPS}^{l} \end{bmatrix} = \begin{bmatrix} \varphi_{INS} - \varphi_{GPS} \\ \lambda_{INS} - \lambda_{GPS} \\ h_{INS} - h_{GPS} \\ v_{n,INS} - v_{n,GPS} \\ v_{e,INS} - v_{e,GPS} \\ v_{d,INS} - v_{d,GPS} \end{bmatrix} \tag{2-34}$$

Since both INS and GPS can provide direct measurements for the first 6 states (the 3 position and three velocity components of the KF state vector), the H matrix is:

$$\mathrm{H_k} = [I_{6\mathrm{x}6} \quad 0_{6\mathrm{x}9}] \tag{2-35}$$

Therefore,

$$\begin{bmatrix} \boldsymbol{r}_{INS}^{l} - \boldsymbol{r}_{GPS}^{l} \\ \mathbf{v}_{INS}^{l} - \mathbf{v}_{GPS}^{l} \end{bmatrix} = \begin{bmatrix} I_{3\mathrm{x}3} & 0_{3\mathrm{x}3} & 0_{3\mathrm{x}3} \\ 0_{3\mathrm{x}3} & I_{3\mathrm{x}3} & 0_{3\mathrm{x}3} \end{bmatrix} + \begin{bmatrix} \boldsymbol{\eta}_r \\ \boldsymbol{\eta}_v \end{bmatrix} \tag{2-36}$$

And the expanded version is

$$\begin{bmatrix} \varphi_{INS} - \varphi_{GPS} \\ \lambda_{INS} - \lambda_{GPS} \\ h_{INS} - h_{GPS} \\ v_{n,INS} - v_{n,GPS} \\ v_{e,INS} - v_{e,GPS} \\ v_{d,INS} - v_{d,GPS} \end{bmatrix}_k = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}_k + \begin{bmatrix} \eta_\varphi \\ \eta_\lambda \\ \eta_h \\ \eta_{vn} \\ \eta_{ve} \\ \eta_{vd} \end{bmatrix}_k \tag{2-37}$$

Also, the measurements covariance matrix *R* (assuming measurements are independent on each other) is

$$R_k = \begin{bmatrix} \sigma_\varphi^2 & 0 & 0 & 0 & 0 & 0 \\ 0 & \sigma_\lambda^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & \sigma_h^2 & 0 & 0 & 0 \\ 0 & 0 & 0 & \sigma_{vn}^2 & 0 & 0 \\ 0 & 0 & 0 & 0 & \sigma_{ve}^2 & 0 \\ 0 & 0 & 0 & 0 & 0 & \sigma_{vd}^2 \end{bmatrix} \tag{2-38}$$

Finally, the state covariance matrix $\boldsymbol{P}_k$ has the variance of predicted states along its diagonal. Cross diagonal elements are cross correlations between various states. In our example, ignoring the cross diagonal elements, the 15 x 15 element square matrix is:

$$P_k = \begin{bmatrix} \sigma^2_{r,3x3} & . & . & . & . \\ . & \sigma^2_{v,3x3} & . & . & . \\ . & . & \sigma^2_{\varepsilon,3x3} & . & . \\ . & . & . & \sigma^2_{\omega,3x3} & .0 \\ . & . & . & . & \sigma^2_{f,3x3} \end{bmatrix} \quad (2\text{-}39)$$

2.3 Portable Navigation

The current time represents a new age of technology that is approaching very rapidly. A new age of wearable computing devices, such as smart glasses which is capable of overlaying a lot of information on user's current view of the world, one example can be overlaying maps, street names, and directions to some destination while the user is driving or walking in streets; another example is smart watches with processing capabilities that enable them to run various number of applications, such as those computing the distance walked or covered during a running drill.

Besides the general purpose computing unit most of these wearable devices include nowadays, one or more of MEMS inertial sensors triads, such as accelerometers, gyroscopes, and magnetometers triads. Some advantages of using MEMS technology is being small in size, light in weight, with low power consumption which facilitates including it in most of today's portable devices where battery life is of big concern, and finally and most importantly being at low cost which helped it spread in most of recent smart phones and tablets. These sensors are mainly included for games or enabling a free-hand interaction between the user and the device, for example, when the user performs certain gesture, one or more of these sensors is used to capture this gesture and a corresponding command is executed in response. These sensors can be used also for navigation purposes, but due to the low accuracy offered by such grade of low-cost commercial MEMS sensors, the error drifts rapidly, and therefore, further enhancements by innovative algorithms are needed than using these sensors readings directly.

In the following subsections, a brief overview on key technologies used in outdoor positioning and indoor positioning, following that, a detailed background about Pedestrian Dead-Reckoning (PDR) is given, then an overview about hybrid indoor positioning, and at the end, related work about employing multiple IMUs to enhance the portable navigation solution is presented.

2.3.1 Outdoor Positioning

The main technologies used in outdoor navigation systems include geo-positioning and wireless communication. Examples of users' needs include routes, which contain roads with a certain speed limit and roads with two or more lanes. Examples of users' preferences are routes with shortest distance, fastest travel time, simple directions, no congestion, scenic sites, and no tolls. The two most common technologies used in todays' outdoor positioning systems are GNSS and wireless communication systems as (i) most of todays' portable devices possess a GNSS receiver chip and/or a wireless communication chip like GSM, Wi-Fi, or Bluetooth, (ii) GNSS is characterized by higher accuracy than other navigation techniques when the environment is appropriate.

For more details about GNSS systems please refer to section 2.2.3, while for wireless communication systems see the next section.

2.3.2 Indoor Positioning

Many techniques have been proposed to enable indoor positioning, using wireless positioning systems in particular, comprehensive coverage can be found at (Hightower and Borriello 2001, Sun, Chen et al. 2005, Liu, Darabi et al. 2007, Gu, Lo et al. 2009). Using existing infrastructure in navigation, such as already deployed communication systems like GSM, Wi-Fi and Bluetooth, cut off expenses and time to install. However, this lead to a partially optimal positioning solution. For example, for Wi-Fi communication networks, it's desired to minimize the

overlap between the access points coverage and each other so that interference between these access points be minimized as well (Ergin, Ramachandran et al. 2007). On the other hand, for having an accurate positioning solution using Wi-Fi networks, it's more desirable that there are larger overlap region between coverage of Wi-Fi access points and each other.

In the following section, techniques relying on inertial sensors and require no physical infrastructure are discussed.

2.3.3 Pedestrian Dead-Reckoning (PDR)

Inertial sensors (e.g. accelerometers and gyroscopes) are commonly used for tracking human movements especially the "on foot" activities (such as for example walking or running) and in this section Pedestrian Dead-Reckoning techniques which depend on inertial sensors are presented.

Pedestrian Dead Reckoning (PDR) is the main technique that employs inertial sensors in indoor positioning or in pedestrian navigation in general, where PDR is mainly consists of three main steps: Step detection, Step length estimation, and Pedestrian heading estimation as seen in Figure 2-5.

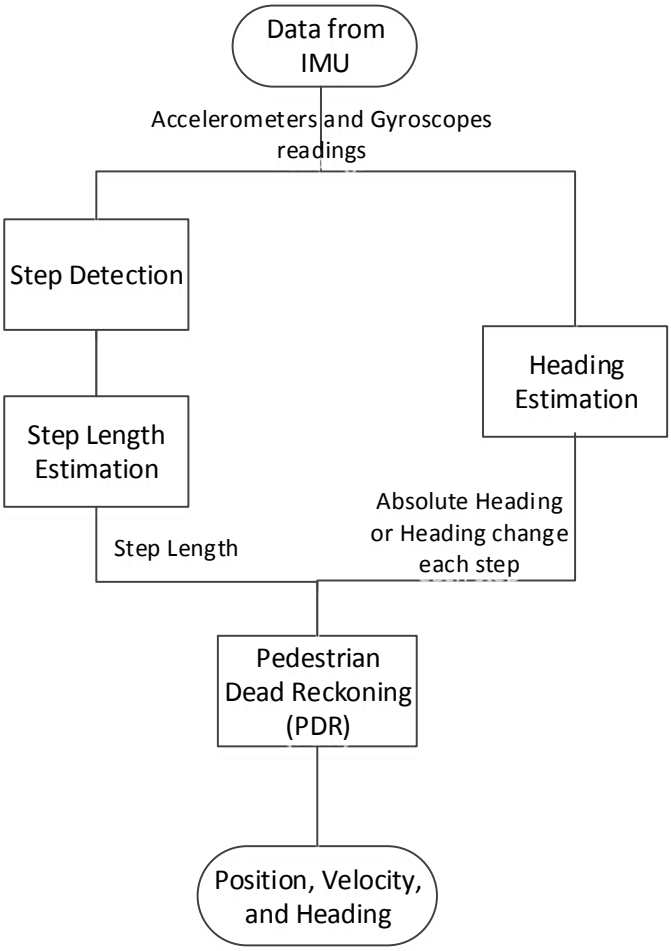


Figure 2-5: PDR

There are many methods known in literature for step detection, and peak detection algorithm is one of the most popular methods. When the heel strikes, sharp changes take place at the vertical acceleration signal. Simple peak detection algorithms can be used to detect these strikes, hence detect the steps. It should be noted that each foot impact might generate multiple local peaks, which requires a more complex peak detection algorithm to avoid such unwanted local peaks. Figures 2-6 and 2-7 show the definition of step vs. stride and an example algorithm on step detection using accelerometer respectively.

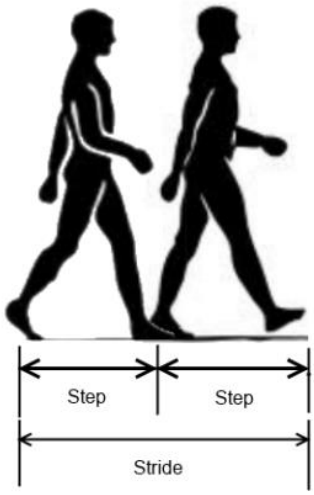


Figure 2-6: Step length definition

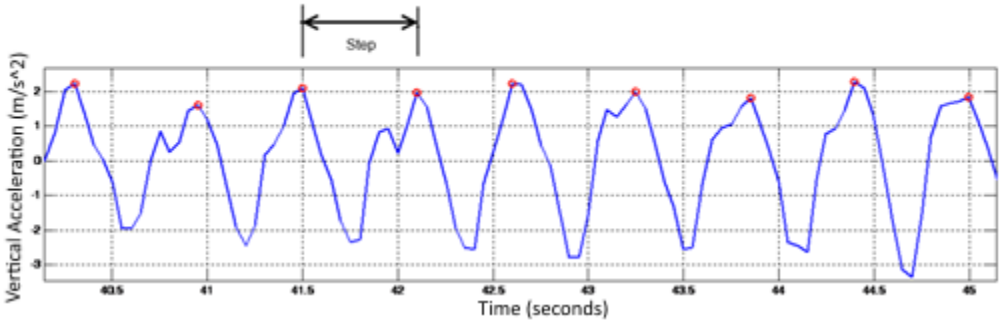


Figure 2-7: Example step detection algorithm

Zero crossings approach is another popular step detection method (Cho and Park 2006), where it makes use of the cyclic property of pedestrian on foot motion pattern, and detects the acceleration value on zero crossings events. This is a popular choice for pedometers or activity monitors due to its simplicity. Also, to avoid false zero crossing events in the signal due to noise, a low pass filter may be applied to the accelerometer signal before applying the zero crossing step detection algorithm. There are several methods known in the literature for estimating step length as well (Shin, Park et al. 2007, Jahn, Batzer et al. 2010, Köse, Cereatti et al. 2012). A common assumption (but not accurate) is that the step length is a constant value regardless the pedestrian

characteristics like height, weight, and gender, or motion dynamics (such as walking or running speed and acceleration).

There are many methods known in literature for pedestrian heading estimation (Titterton and Weston 2004, Noureldin, Karamat et al. 2013). Heading estimation within a PDR is typically not different from an INS since there are so few sensors available as inputs. Single integration of gyroscope signals provides estimates of heading change. In addition, some systems use only a single gyroscope mounted parallel to the torso, making the assumption that its axis remains (near) vertical during walking. Magnetometers may also be used directly or fused with the gyroscope outputs to estimate heading (Groves 2013). 2D-magnetometers or 3D-magnetometers can be used. Heading estimated using magnetometers is known in literature to be accurate but only when magnetometer used is well calibrated. The main problem of using magnetometers is it being disturbed by magnetic objects and electro-magnetic field in the surrounding environment, causing some bias and scale factor errors due to different error sources such as hard iron effect and soft iron effect of these magnetic objects and electro-magnetic field on the magnetometer used in the PDR solution.

2.3.4 Hybrid Indoor Positioning Solutions

Some work in PDR looks similar to research performed in the robotics community. However, the problem in PDR is significantly more challenging, since with pedestrian, there is no longer control on how the sensors move and change their alignment with respect to the moving body. Nonetheless, a number of techniques used in robotics research have proved useful to PDR problem domain. For example, incorporating environment information, such as building maps in particular, introduces some more useful constraints on the PDR solution, like pedestrians cannot pass walls and only doors can be passed through. Such type of constraints can be used to update the PDR solution, and enhance the medium-term drift of the overall solution.

K. Abdulrahim et el. (Abdulrahim, Hide et al. 2012) introduced a different type of orientation constraint, constraining pedestrian's movements to lie along paths parallel to the external building walls. For a typical rectangular building, this constraint allows movement in one of two perpendicular directions. Over a series of eight walking loops that varied in duration from 12–40 minutes and in length from 0.5–3 km, they achieved an average return-to-start error of 4.62 m using zero velocity updates (ZUPTs) and orientation constraints, versus an error of 153.62 m when only ZUPTs were applied.

In case no map is available, Simultaneous Localization and Mapping (SLAM) techniques can be used. These techniques were developed to allow a robot to locate itself within an environment it has not seen before (Thrun, Burgard et al. 2005). The typical sensors used for SLAM are laser rangefinders and cameras. These types of sensors are not popular in pedestrians' applications yet, but what is really good about these techniques is that it enables location systems that improve over time as they learn about the environment.

The integration between the different techniques mentioned above is performed traditionally using example, Kalman filter, which can be used, for example, to integrate information from different environments (such as in the case of maps and SLAM) with PDR. Particle filtering is another possible approach for system integration that allows the incorporation of complex constraints that can help limit drift (Maskell and Gordon 2001, Rekleitis 2004, Georgy and Noureldin 2011). Particle filter has the advantage of dealing nonlinear dynamics and arbitrary noise characteristics, which are usually the case in portable navigation.

2.4 Multiple IMU's Usage

APPlication acCESSORY or *appcessory* for short is a new term commonly used nowadays to describe an accessory for a portable device that is specific for certain application, and which can use Wi-Fi or Bluetooth for communication with other portable devices such as smartphones.

Appcessories represents another shift in portable devices technology. Many appcessories are available for most of today's portable devices, such as, belt clips, and digital wristwatches connected to the portable device such as smartphone or tablet to stream social networks' status updates, or show the reviews of nearby points of interest on its screens, or monitoring user physical activities and calculate how many calories user burnt out for certain time. Such appcessories include also one or more inertial sensors triads such as those found in wearable computing devices mentioned previously. Current electronic gadgets consumer can carry (or wear) multiple devices each has one or more low-cost MEMS sensors triads, such as smart glasses, smartwatch, smartphone, and a belt-clip appcessory, just to name a few, see Figure 2-8.

Figure 2-8: Some Examples on Today's Wearable and Portable Devices (Wikipedia 2014)

GPS and IMUs have been successfully integrated since the formal introduction of GPS (El-Sheimy, Kai-wei et al. 2006, Grewal, Weill et al. 2007, Groves 2013). More recently, attention has been placed on integration with MEMS IMUs to reduce cost, but still provide reliable navigation solutions (Noureldin, Karamat et al. 2009, Georgy, Karamat et al. 2011, Shen, Georgy et al. 2011, Iqbal, Georgy et al. 2013). A natural progression is to use multiple IMU sensors, and thus capitalize on the decreasing cost of MEMS sensors, in order to improve overall accuracy.

Y. Jin et al. (Jin 2011, Jin, Toh et al. 2011) presented a pedestrian tracking scheme using two sets of low cost DR sensors (MEMS IMUs). This pedestrian tracking scheme exploits the stable relative displacements between two sensor modules carried by the same pedestrian. The positioning problem was defined as a maximum a posteriori sensor fusion problem and a derivation for the optimal solution for this problem was proposed. The optimal solution proposed was shown to be the weighted average of the two sensor modules' solution, and which can be illustrated as shown in Figure 2-9. For the rest of this document, this architecture will be referred to by the Weighted Average (WA) method. The proposed solution was experimentally evaluated by using, (i) two mobile devices, each containing a single orientation sensor, mounted with arbitrary device orientations, (ii) one mobile device, containing two different orientation sensors, mounted with fixed device orientation. The proposed scheme had exhibited good tracking performance with good error reductions, compared to traditional DR, in both scenarios. The largest error reduction rate was reported at 73.7%. However, in some other runs, the proposed algorithm delivered intermediate tracking performance, with average errors in between those of the two phones' individual DR systems.

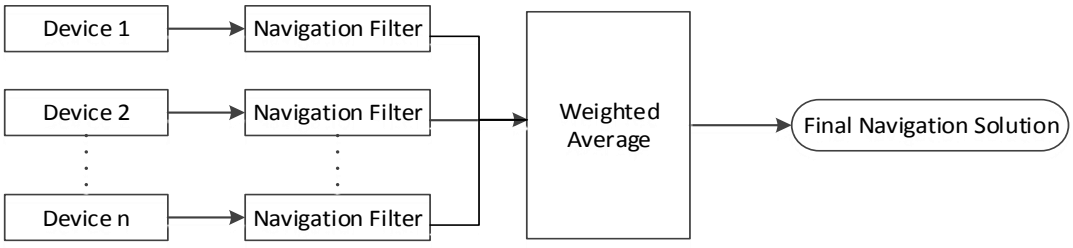


Figure 2-9: The Weighted Average (WA) method architecture

In another work (Yuksel 2011, Yuksel and El-Sheimy 2011), a solution of the optimum inertial sensor fusion problem was derived for skew redundant inertial measurement units

(SRIMU). In the SRIMU, the redundancy of the sensors generates additional observations which can be used to estimate the sensor errors without requiring any external aid. In this study, it was shown that the projection of raw sensor outputs to the left null space of the sensor configuration matrix can be used to define these redundancy observations. Based on these observations, the computation of best acceleration and rotation rate (which can be used to execute the standard INS equations in an SRIMU based navigation system) was determined.

J. B. Bancroft (Bancroft 2010) also presented three methods for fusing multiple IMUs at two different layers or domains. First layer was defined to be in the observation domain. The second layer was higher than the first one and was defined to be in the estimation domain. Two methods were presented under the first layer, and one method under the second layer. The first method is simply mapping all raw IMU measurements into a common frame (i.e. a virtual frame) and processed in a typical combined GNSS-IMU Kalman filter as shown in Figure 2-10. The second method is similar to the first one except a large stacked filter was constructed of several IMUs as illustrated in Figure 2-11. This filter construction allowed for relative information between the IMUs to be used as updates. Finally, the third method was a federated filter used to process each IMU as a local filter. The output of each local filter is shared with a master filter, which in turn, shares information back with the local filters.

Results indicated that the stacked filter provides a linear increase in accuracy, while other architectures typically have less improvement with the addition of more than three IMUs. Areas where GPS is sufficient show little improvement with additional IMUs. Only the stacked filter decreases the minimal detectable blunder of GPS observations by a significant amount.

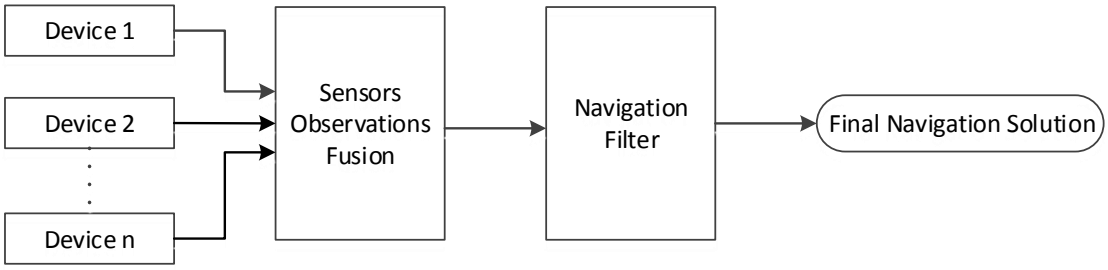


Figure 2-10: The sensors observations fusion method architecture

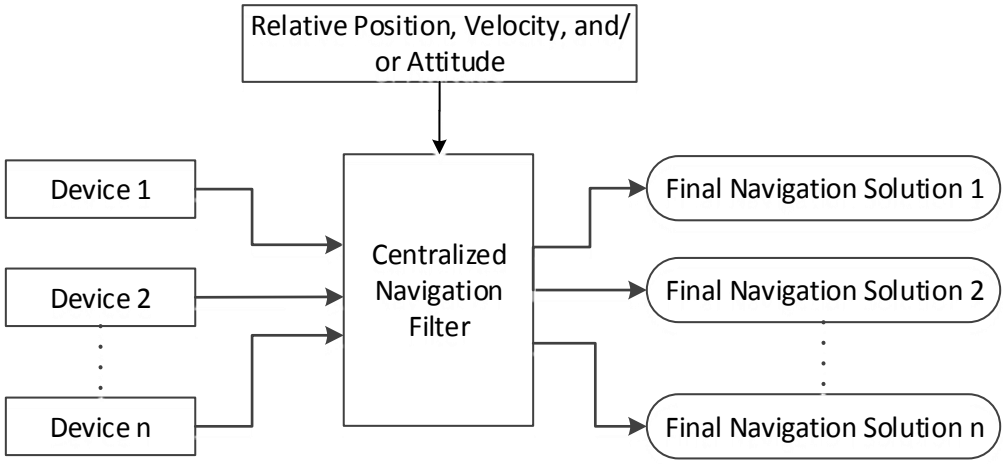


Figure 2-11: The centralized filter method architecture

J. B. Bancroft again introduced Relative Position Updates (RPUPT) in another work (Bancroft 2009). Assuming the IMUs are rigidly mounted, and the vector between them (in the body frame) remains constant, it was possible to use this information as an update to the system. This update does not provide an absolute position correction (as a GPS update does), but provides a method of constraining the rate of divergence between two IMUs. Relying on the same principle idea discussed before in the RPUPT, Relative Velocity Updates (RVUPT) and Relative Attitude Updates (RAUPT) were introduced in the same work (Bancroft 2009). It was shown that relative

attitude updates provide pessimistic position variances, while the relative position and velocity updates provide more realistic position variances. The combination of position, velocity, and attitude updates provides the most significant improvement, where in seven 30-seconds GPS-outages, the position improvement using five inertial units was 34 %, while in urban canyons there was an improvement of 55 % on the RMS error during 400 s of data.

2.5 Nonlinear system identification

System identification is different than system modeling. System modeling can be defined as constructing a mathematical model from physical laws such as Newton's laws. On the other hand, system identification can be defined as constructing a mathematical model also, but by assuming a general form, and then identify the parameters of this general form to represent the system to be identified.

There are some general forms to represent linear systems, and others to represent nonlinear systems. In general, the nonlinear system representations can be thought of as a generalization in one way or another of the linear representations. For example, Volterra and Wiener series models for nonlinear systems can be thought of the generalization of impulse response function of a linear system. Therefore, the nonlinear system identification problem can be defined as identifying the parameters (or kernels) of some nonlinear general form of nonlinear system such as Volterra or Wiener series. Equation (2-39) shows the convolution of the infinite impulse response (IIR) of a linear system and some input u(t), while equations (2-40) and (2-41) shows Volterra and Wiener series respectively.

$$\mathrm{y(t)} = \int_{-\infty}^{\infty} h(\tau)u(t-\tau)d\tau \tag{2-39}$$

$$\mathrm{y(t)} = \sum_{q=0}^{\infty} \int_{-\infty}^{\infty} \dots \int_{-\infty}^{\infty} h^{(q)}\left(\tau_1, \dots, \tau_q\right)u(t-\tau_1)\dots u\left(t-\tau_q\right)d\tau_1 \dots d\tau_q \tag{2-40}$$

$$y(t) = \sum_{q=0}^{\infty} G_q[K^{(q)}(\tau_1, \dots, \tau_q); u(t'), t' \leq t)] \quad (2\text{-}41)$$

There are many identification methods for identifying a model's parameters. Some methods are based on correlation functions, others are based on solving explicit least-squares regression problems. Finally, some additional methods are based on iterative least-squares algorithms.

Accuracy of the resulting models represents a key metric when judging which method is most suitable for which application. However, it is not the only metric, computational requirements and storage considerations can be key players too. Typically, these are the three key metrics that are first considered when choosing an identification method for identifying the parameters of some model.

2.5.1 Artificial Neural Networks (ANN)

One of the most popular nonlinear system identification methods is Artificial Neural Networks (ANN). ANNs are based on brain-like models for processing information, and these models are characterized by being massively parallel in architecture. That is why ANNs are commonly implemented on a parallel-computing based hardware such as neurocomputers or at least take advantage of graphical processing units (GPUs) on conventional computers.

ANN are most common in solving problems characterized by (i) Nonlinearities, (ii) High dimensionality, and (iii) a lack of a clearly stated mathematical solution or algorithm. However, ANNs are usually encouraged when (i) An ANN provides the only practical solution, or (ii) other solutions exist but an ANN gives an easier or more efficient solution according to some metric.

2.5.2 Fast Orthogonal Search (FOS)

FOS is another nonlinear system identification method. FOS main idea is based on finding functional expansions using an arbitrary set of non-orthogonal candidate functions (Korenberg 1989, Korenberg and Paarmann 1989). The algorithm can be applied to a variety of practical problems. Applications of FOS in different research fields have shown successful detection of signals buried in short- and long-term errors.

One important advantage of FOS is its ability to choose both the candidate functions representing the system together with obtaining their coefficients, as opposed to some other methods that just obtains coefficients for known model structures. FOS is an efficient and effective method of system identification that is able to find accurate models of systems even from very large numbers of candidate basis functions. FOS can choose candidates from an over-complete set since it has the ability to choose the most significant candidates.

More details on FOS method and its implementation, in addition to why use FOS vs. ANN is presented in Chapter 4, where FOS were chosen to model the relation between step length and some motion parameters such as step frequency and vertical acceleration variance.

Chapter 3

Experimental Setup

3.1 Experimental work strategy

The experimental work strategy followed can be divided into the subsequent steps. First, collecting initial data set enough for developing proposed algorithms using a prototype unit; more details about this prototype unit in the hardware section. The preferred language for developing the proposed algorithms and preliminary testing and analysis was MATLAB®. Afterwards, the algorithms were written in C language and integrated into Trusted Portable Navigator (T-PN) and T-PN ME, depending whether the algorithm is enhancing a navigation solution using single portable/wearable device or multiple of those devices. The T-PN ME is the multiple device extension of the T-PN. T-PN and T-PN ME are navigation solutions developed by Trusted Positioning Inc., a fully owned subsidiary of Invensense Inc.. More information about T-PN and T-PN ME are in the software section. Finally, the integrated version of T-PN or T-PN ME is deployed on one or more of the different consumer devices available currently in the market for assessing and verifying algorithm efficiency. Figure 3-1 shows a summary of the experimental work strategy followed during this research work.

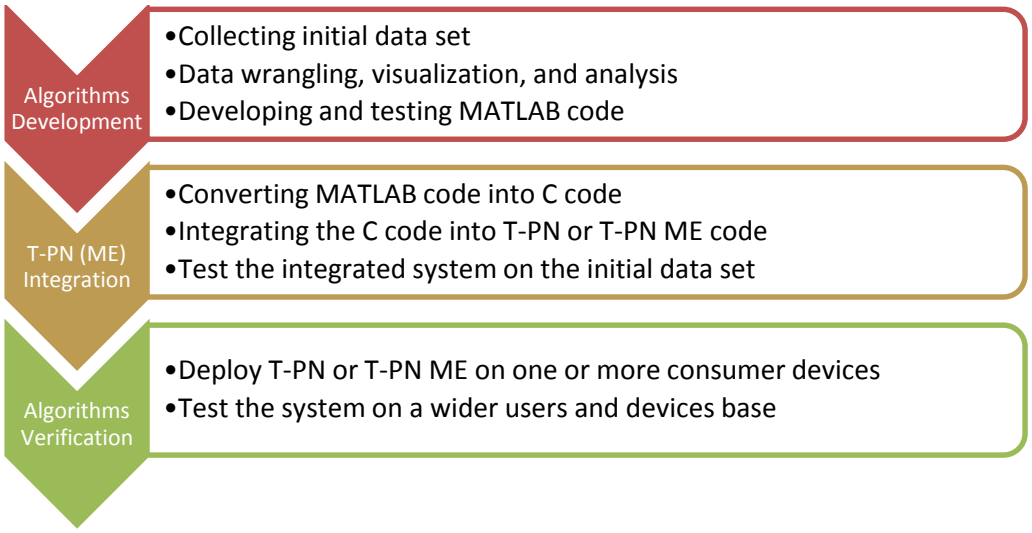


Figure 3-1: Experimental work strategy

3.2 Hardware platform

One prototype was used to emulate smartphone, smart watch, smart glasses, and smart belt clip. A few units were used while collecting the initial data set required for development of algorithms. The prototype unit consists of a six degrees of freedom inertial unit from Invensense (i.e. tri-axial gyroscopes and tri-axial accelerometer) (MPU-6050), tri-axial magnetometers from Honeywell (HMC5883L), barometer from Measurement Specialties (MS5803), and a GPS receiver from u-blox (LEA-5T).

Many other devices were used afterwards for verifying results obtained from the initial data set. All of these devices were android based; thanks for its openness. Some examples on the devices were used for verification are shown in Figures 3-2 – 3-4. In addition, Table 3-1 summarizes the sensors of interest found in each consumer device.

Figure 3-2: Example smartphones, Samsung S3, Samsung S4, and Google Nexus 4 respectively

Figure 3-3: Example smart watches, Samsung Galaxy Gear Watch 1 & 2 respectively

Figure 3-4: Example smart glasses, Oakley AirWave™ goggles (left), and Vuzix M100 Smart glasses (right)

Table 3-1: Model names of portable devices used in data collection and the sensors they contain. Only a subset of the sensors listed for each device was used

Device Type	Model Name	Sensors
Smartphone	Samsung SGH-I747 SIII	InvenSense MPU-6050 6-Axis IMU: - Tri-axial - Accelerometer - Tri-axial Gyroscope AsahiKasei AKM8975 3-Axis Electronic Compass Bosch BMP180 Pressure sensor Sharp GP2A Light Sensor Sharp GP2A Proximity Sensor Qualcomm GNSS Receiver
	Samsung SM-N900 Galaxy Note 3	InvenSense MPU-6500 6-Axis IMU: - Tri-axial Accelerometer - Tri-axial Gyroscope Yamaha YAS532 Tri-axial Magnetometer Bosch Barometer Maxim MAX88921 RGB Light Sensor Samsung Orientation Sensor Maxim MAX88921 Proximity Sensor Sensirion SHTC1 Ambient Temperature Sensor GPS Receiver
	Google Nexus Phone	InvenSense MPU 9150 9-axis IMU: - Tri-axial Accelerometer - Tri-axial Gyroscope - Tri-axial Magnetometer Bosch BMP180 Pressure sensor Sharp GP2A Light sensor Sharp GP2A Proximity sensor SiRF SiRFstarIV GSD4t GPS Receiver
Tablet	Google Nexus 7 Tablet	InvenSense MPU-6050 6-Axis IMU: - Tri-axial Accelerometer - Tri-axial Gyroscope AKM Tri-axial Magnetometer LSC Lite-On Ambient Light Sensor Broadcom BCM4751 GPS Receiver

Tablet	Google Nexus 10 Tablet	InvenSense MPU-6500 6-Axis IMU: - Tri-axial Accelerometer - Tri-axial Gyroscope InvenSense Tri-axial Magnetometer Bosch BMP182 Barometer Rohm BH1721fvc Light sensor GPS Receiver
Smartwatch	Samsung SM-V700 Galaxy Gear Digital Smartwatch	InvenSense MPU-6500 6-Axis IMU: - Tri-axial Accelerometer - Tri-axial Gyroscope
	Samsung Gear Live Watch	InvenSense MPU 9250 9-axis IMU: - Tri-axial Accelerometer - Tri-axial Gyroscope - Tri-axial Magnetometer InvenSense ICS-43430 Microphone Heart Rate Sensor
Smart glasses	VUZIX M100 Smart glasses	InvenSense MPU-6500 6-Axis IMU: - Tri-axial Accelerometer - Tri-axial Gyroscope Tri-axial Magnetometer Light Sensor Proximity Sensor Gesture Sensor Rotation Vector Sensor Gravity Sensor
	RECON Snow Goggle	STMicroelectronics Tri-axial Gyroscope STMicroelectronics Tri-axial Accelerometer STMicroelectronics Tri-axial Magnetometer GPS Receiver

3.3 Software platform

As mentioned in the beginning of this chapter, two main languages that were used during this research, MATLAB® and C. MATLAB® was chosen for developing most of the data processing scripts, and automating tasks such as data wrangling, visualization, and analysis. Besides, rapidly developing and testing algorithms. However, to test these developed algorithms in

real world scenarios using commercial portable and wearable devices, conversion to C was inevitable.

Navigation solutions can evolve very easily and rapidly to a very complex piece of software. Figure 3-5 shows the building blocks for an example navigation solution. In order to develop a complete navigation solution which can work seamlessly whenever there is a GNSS signal available or not, or while changing motion mode from walking to driving to walking again or may be running. A complete team of researcher and developers is needed. Here comes the role of T-PN and T-PN ME. T-PN represents the complete navigation solution using a single portable/wearable device, this solution was developed by one of the top navigation researchers and developers team in the field of navigation. Instead of developing a limited navigation simulation program, the objective of this research was to use T-PN as the underlying navigation solution and extend/enhance it, which led to T-PN ME version later on after the cooperation with the T-PN team. The work in this thesis just represent the seed for T-PN ME. T-PN ME is more advanced than what is shown here and it is the result of cooperation of whole team of researchers and developers and not the work of one person.

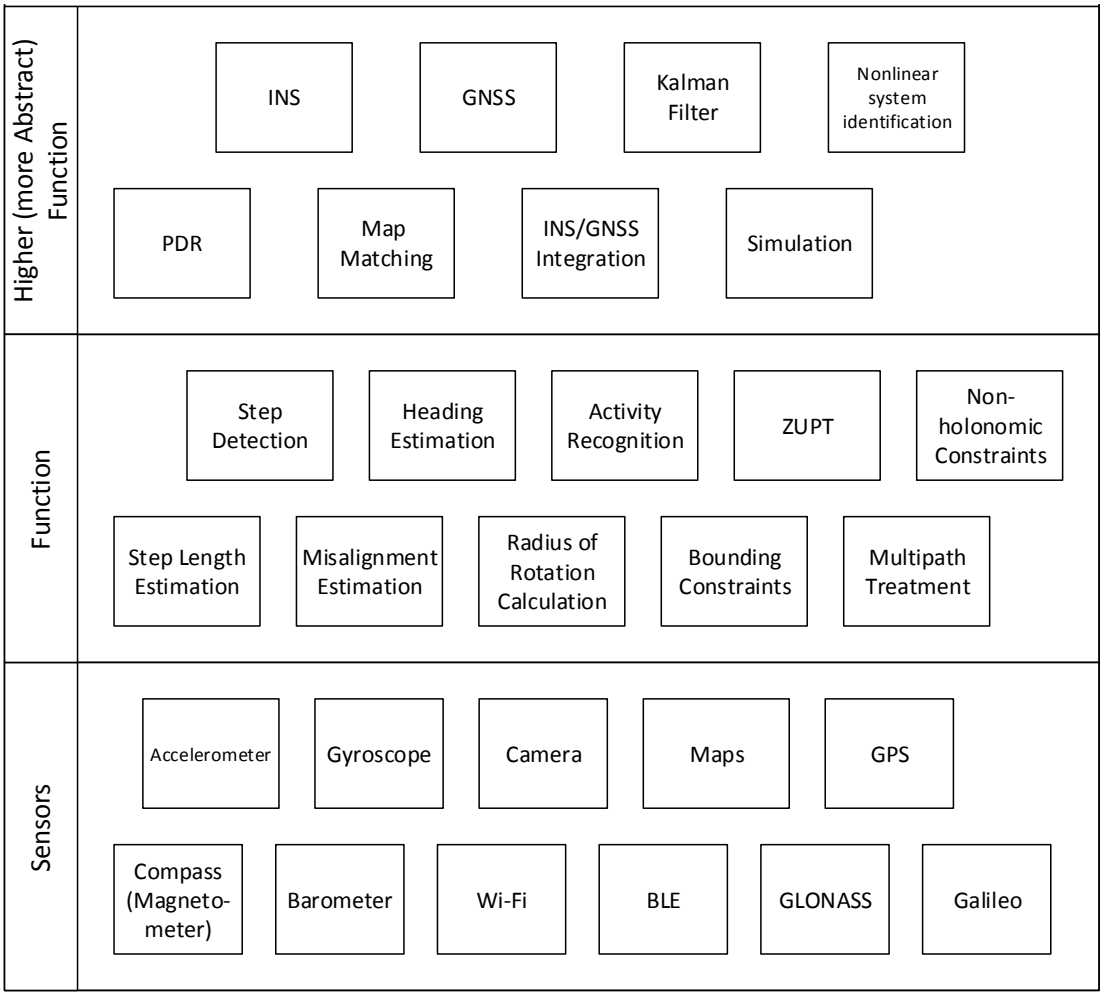


Figure 3-5: Example Navigation System Building Blocks

Testing on real world scenarios using commercial portable and wearable devices has two main advantages than testing using simulation.

First is avoiding being over optimistic. This can happen when simulation unintentionally isolate one or more sources of errors to test the method under investigation, these errors later on after integrating the tested method with a full functional navigation solution may exceed the improvement due to the tested method, which make the result obtained from simulation result over optimistic to the result of the fully integrated solution.

Second is avoiding being over pessimistic. This can happen when a coupling relationship is missing in the simulation environment. One popular example in the field of navigation is the coupling relationship between accelerometers and gyroscopes biases and the position and velocity states of the navigation solution (Noureldin, Karamat et al. 2013). A missing coupling relationship can lead to an over pessimistic performance due to the fact that improving one quantity in the simulation environment will not lead to the improvement of another due to the lack or the missing of this coupling relationship between these two quantities.

Enhancing T-PN is a true challenge. T-PN accuracy is approximately 4-8% of distance travelled when operating without any wireless updates. Yet, its computation demand is less than many other competing portable navigation solutions. Therefore, given the above metrics, any further enhancement not only shall be significant but also worth it; in terms of CPU cost. The proposed work in this research succeeded in improving T-PN accuracy by a factor of 1.5-3 times, (this performance enhancement varies based on the number of devices integrated in the solution), and represents the seed for T-PN ME.

3.4 Statistics about personnel involved and data collected

In addition to testing the proposed algorithms on different consumer devices, a number of users with different physical characteristics were asked to use and test the solution. Up to 25 users in some cases have been asked to test the final solution. Each user at least collected two complete data sets. A data set represent a number of devices which are connected together and used in predefined use cases (or a sequence of use cases). The final database size is approximately 50 GB, consisting of more than 2000 walking and driving trajectories, and spanning more than 150 hours' worth of data.

Motion modes covered during the data collection:

- ✓ Walking
- ✓ Driving
- ✓ Running

Phone Use cases covered while walking,

- ✓ Handheld
- ✓ Dangling (or swinging)
- ✓ On arm
- ✓ On chest
- ✓ On leg
- ✓ In pocket
- ✓ On belt
- ✓ On ear
- ✓ Texting (both portrait and landscape).

Use cases covered while running:

- ✓ Handheld
- ✓ Dangling (or swinging)
- ✓ On arm
- ✓ On chest
- ✓ On leg
- ✓ In pocket
- ✓ On belt

Use cases covered while driving:

- ✓ In pocket
- ✓ On belt
- ✓ On seat
- ✓ In a cradle
- ✓ In a cup holder

Other devices used besides smartphone:

- ✓ Smart watch
- ✓ Smart glasses
- ✓ Smart belt-clip

In the case of step length estimation, a reference step length is collected from the GNSS solution besides the inertial data from accelerometers and gyroscopes. On the other hand, there were no reference for the misalignment work. However, there was a record present for each device use case per each data set. Knowing the device use case plus the final position and velocity solutions are enough to judge on the misalignment solution.

Chapter 4

Step Length Estimation Using Multiple Portable and Wearable Devices

Integrating acceleration and heading rate signals leads to fast drift. However, using step and heading system (SHS) instead avoids this fast drift. That is why step length estimation is one of the most important aspects of pedestrian navigation solutions, especially in pedestrian dead reckoning (PDR). In this chapter, novel methods for step length estimation using both single and multiple sensors assemblies are presented.

4.1 Current State of the art

Inertial sensors (e.g. accelerometers and gyroscopes) are commonly used for tracking human movements especially the "on foot" activities (such as walking or running). Accelerometers are used for measuring user's acceleration from which steps can be detected and existing step length estimation techniques are based upon. Figures 4-1 and 4-2 show the definition of step vs. stride, and an example algorithm on step detection using accelerometer respectively. Step length estimation of a pedestrian is used in several applications such as PDR or measuring the distance travelled by the user.

The step length of pedestrians was estimated in many former research works with noticeable approximation. Some methods require the placement of inertial sensors in certain locations on the user's body (for example, on foot, waist or chest … etc.) (Cho and Park 2006, Sun, Mao et al. 2009, Köse, Cereatti et al. 2012), which again make it hard to apply these methods for a variety of applications except those supporting the location for which the method was built. Some other methods defines the step length as be a function of the pedestrian characteristics like height, weight, and gender, or motion dynamics (for example walking or running speed and acceleration).

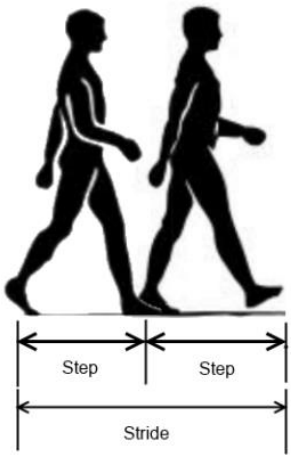


Figure 4-1: Step length definition

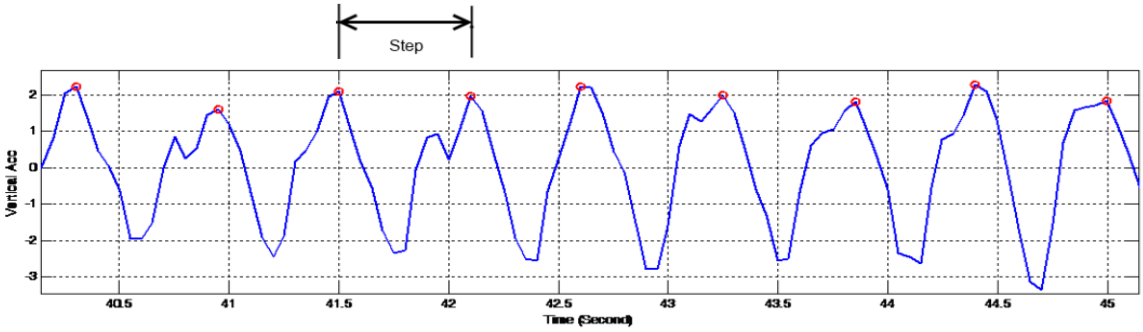


Figure 4-2: Example step detection algorithm

One method for step length calculation is the double integration of acceleration readings to obtain pedestrian displacement (Zijlstra and Hof 1997, Alvarez, González et al. 2006). This method, if used alone, results in lower accuracy due to the drift increasing over time and the high noise and errors in commercial accelerometer readings especially for the cases with low acceleration values. The common technique to enable using this method is to restart the integration with each detected step because the velocity is zero at each footfall. This is why this technique prefers foot mounting or lower leg mounting, to avoid the body sway as this technique needs zero velocity update at each step detection.

Other known methods for step length estimation may be categorized roughly into two main groups, namely methods based on biomechanical models and methods based on empirical relationships (Jahn, Batzer et al. 2010). As an example on biomechanical model methods is kneeless biped model for step length estimation, where in this method, the kneeless biped is modeled as an inverted pendulum and the final estimation is scaled with some constant dependent on the user (Zijlstra and Hof 1997, Alvarez, González et al. 2006). One main drawback of this method is that it is user-dependent since for this method to work properly, one must find the best scale constant for each specific user.

Another example on biomechanical model methods can assume a more complex model by considering the vertical displacement of the center of mass of the user's body through double integration of the acceleration and ruled by two pendulums, namely, an inverted pendulum with the leg's length during the swing phase, and a second pendulum, during the double-stance phase of the step, the stride length (where stride length is defined as the length of two consecutive steps) (Alvarez, González et al. 2006). In this case, the method can be computed as the summation of the displacements in both stages. One main drawback of this method is that it requires the knowledge of the foot length (from first metatarsal head to the calcaneal tuberosity) of the user, which makes it again, user-dependent/specific, and certain configurations may work with one user but not with another user because of differences in their physical characteristics like height and/or leg-to-body ratio.

The same problem again appears in the empirical relationships, where there is one or more parameter which requires calibration and customization for each user. As such, these methods are practically not appealing. One example of these methods is one that develops a relationship between the stride length, the step counter, the first harmonic of the vertical acceleration of the center of mass of the person, and a constant that needs calibration (Weinberg 2002, Alvarez, González et al.

2006). There are several other empirical methods, in some of which, the equations are obtained experimentally and need calibration of parameters.

Some prior work that may fall under the empirical category noticed that the speed of a moving pedestrian can affect his step length, for example when a person walks faster he tends to increase the length of his step, also his motion style and dynamics are affected by his increased speed. A related parameter is that the step frequency affects the step length (Alvarez, González et al. 2006).

This type of methods is in general more suitable than other methods for various applications even without need for special mounting of the inertial sensors on the body or special mounting in a certain location. In some prior work it was assumed that the step length has a linear relation with the step frequency, where step frequency indicates how many steps are detected per second. The results of this method as is are not satisfactory for applications that requires accurate varying step length estimation as the frequency of step is not the only parameter that can affect the step length. Therefore another technique was used assuming the step length has a linear relation with both the step frequency and the acceleration variance in a step, where acceleration variance is the variance of the acceleration measured by the accelerometers during one step period (Shin, Park et al. 2007).

The parameters of the linear model are obtained either by online training when GNSS signal is available or by offline training before the model is used by the user. The main drawback of online training is that the model requires to be trained with different walking or running speeds, which is not guaranteed and most probably will not happen in a natural real-life trajectory during GNSS availability for online training. This issue is solved by using offline training and the step length estimation is better than online training implementations.

The main drawback in the last mentioned approach is the assumption of linear relation, which neglects the effect of some motion dynamics and speeds that differ among users and can

cause the relation to be nonlinear. This problem is also more severe when the step length estimation is intended for both walking and running, not just walking with different speeds.

4.2 Step Length Estimation based on Nonlinear System Identification Methods

In this chapter a new method is introduced for varying step length estimation for "on foot" activities, such as walking and running, using nonlinear system identification. This work is patent pending. The method proposed assumes a nonlinear relation between step length and different parameters that represent human motion dynamics such as step frequency, acceleration variance during step, acceleration peak value during step, peak to peak acceleration value during step, among others. Two phases are required for accurate estimation of varying step length using the proposed method.

The first phase is a model building phase where a nonlinear model is built offline by a nonlinear system identification technique. The nonlinear system identification technique is fed by different parameters that represent human motion dynamics during on foot motion with different speeds, collected by different people with different characteristics such as weights, heights, genders, etc. The second phase is the usage phase in which the nonlinear model is used directly in real-time with any user to estimate a varying step length that can vary nonlinearly with human motion dynamics.

Fast Orthogonal Search (FOS) is chosen as the nonlinear system identification technique for implementing the proposed method. That is why a brief background about FOS is presented in the following subsection.

4.2.1 Fast Orthogonal Search (FOS)

FOS is a general-purpose nonlinear system identification technique that finds functional expansions using an arbitrary set of non-orthogonal candidate functions (Korenberg 1989, Korenberg and Paarmann 1989). The algorithm can be applied to a variety of practical problems.

Applications of FOS in different research fields have shown successful detection of signals buried in short- and long-term errors. The following is a background explanation about FOS algorithm.

FOS is used in the system identification of a nonlinear system; the system is given by the difference equation model (Korenberg 1988)

$$y(n) = F[y(n-1), \ldots \ldots y(n-K), \quad x(n), \ldots . x(n-L)\ ] + e(n) \tag{4-1}$$

Where x(n) is the system input, y(n) is the system output while the system error is given by e(n) and n is given as n = 0,…..,N where N is the number of samples .

The above equation can be represented as follows:

$$y(n) = \sum_{m=0}^{M} a_m\, P_m(n) + e(n) \tag{4-2}$$

Where a_m represent the weights of the non-orthogonal candidate functions, P_m is the non-orthogonal candidate functions, while M is the number of candidate functions selected by the FOS algorithm.

Fast orthogonal Search begins by building a functional expansion of y(n) using a group of orthogonal functions as follows:

$$y(n) = \sum_{m=0}^{M} g_m w_m(n) + e(n) \tag{4-3}$$

Where g_m is the weight of the orthogonal functions and w_m is a set of orthogonal functions derived from P_m by using Gram-Schmidt (GS) Ortho-normalization process. The GS weights $\alpha_{m,r}$ are produced by using the GS algorithms; these weights are used to calculate the orthogonal function w_m according to the following equation:

$$\omega_m(n) = P_m(n) - \sum_{r=0}^{m-1} \alpha_{m,r} w_r(n) \tag{4-4}$$

The mechanism of using the FOS to identify the nonlinear system can be described in few steps; first the calculation of the $\alpha_{m,r}$ by using the GS method, second step is the calculation of the

$g_m$where it is used with the output $y(n)$ to find the weight of the non-orthogonal candidate functions a_m.

The first step is the calculating of the Gram Schmidt weights α_{mr} by using the following equation:

$$\alpha_{mr}(n) = \frac{\overline{P_m(n) * \omega_r(n)}}{\omega_r^2(n)} \tag{4-5}$$

Where m = 1, 2, 3 … M is the index of the current term while r = 0, 1, 2 … m-1 is the index of the previously term. In order to calculate the GS weight without calculating the orthogonal functions ω_m, the correlation of the candidate functions and the orthogonal functions $D(m,r)$is calculated according to equation (4-6).

$$D(m,r) = \overline{P_m(n) * \omega_r(n)} \tag{4-6}$$

The previous equation can be expressed by:

$$D(m,r) = \overline{P_m(n) * P_r(n)} - \sum_{i=0}^{r-1} \alpha_{ri} D(m,i) \tag{4-7}$$

One great advantage of FOS is that it does not require explicit creation of the orthogonal functions. This greatly reduces computing time. In addition, storage requirements are considerably diminished. All of this without any degradation of the results over the basic technique to explicitly create the orthogonal functions (Korenberg 1988).

From the above equations, the weights of GS can be expressed as follows:

$$\alpha_{mr}(n) = \frac{D(m,r)}{D(r,r)} \tag{4-8}$$

While the orthogonal functions ω_m is given as in equation (4-9)

$$D(m,m) = \overline{\omega_m(n)} \tag{4-9}$$

After the calculation of the Gram Schmidt weights α_{mr}, the orthogonal functions are used to calculate and compute the orthogonal weights g_m. The main target of using the GS weights in the computation of the orthogonal weights is to minimize the Mean Squared Error (MSE) of the orthogonal functional expansion. The MSE is given as follows:

$$\mathrm{MSE} = \overline{\varepsilon^2(n)} = \overline{y^2(n)} - \sum_{m=0}^{M} {g_m}^2 * \overline{\omega_m^2(n)} \tag{4-10}$$

By deriving the above equation with respect to the g_m and setting its value to zero, the value of the orthogonal weights (gm) which minimize the value of the MSE is given as in equation (4-11)

$$g_m = \frac{\overline{y(n) * \omega_m(n)}}{\overline{\omega_m^2(n)}} \tag{4-11}$$

The calculation of the mean average value of the output and the orthogonal functions required to calculate the orthogonal weights g_m is given in the following equation:

$$C(m) = \overline{y(n) * \omega_m(n)} = \overline{P_m(n) * y(n)} - \sum_{r=0}^{m-1} \alpha_{mr} * C(r) \tag{4-12}$$

Where $C(m)$ could be solved recursively from $C(0)$ which is equal to the mean square average of the output $y(n)$ and equation (4-12) where m = 1, 2, 3 … M. Then the value of the orthogonal weights g_m is given as follows:

$$g_m = \frac{C(m)}{D(m,m)} \tag{4-13}$$

The MSE reduction from the addition of the m^{th} term can be represented by the equation (4-14) where the candidate with the largest value for Q is selected as the model term, but optionally its addition to the model may be subject to its Q value exceeding a threshold level.

$$Q(M) = {g_m}^2 * \overline{{\omega_m}^2(n)} = {g_m}^2 * D(M,M) \tag{4-14}$$

By calculating the orthogonal weights g_m and the GS weights α_{mr}, the coefficients a_m in equation (4-2) can be calculated by using the following formula:

$$a_m = \sum_{i=m}^{M} g_i * v_i \tag{4-15}$$

$$v_i = -\sum_{r=m}^{i-1} \alpha_{ir} * v_r \qquad \text{and } v_m = 1 \tag{4-16}$$

In the above equation the index i = m + 1, 2 … M.

By calculating the coefficients a_m and having the model terms $P_m(n)$ selected, then the system in equation (4-2) can be represented.

4.2.2 Why FOS?

One important advantage of FOS is its ability to choose both the candidate functions representing the system together with obtaining their coefficients, as opposed to some other methods that just obtains coefficients for known model structures. FOS is an efficient and effective method of system identification that is able to find accurate models of systems even from very large numbers of candidate basis functions. FOS can choose candidates from an over-complete set since it has the ability to choose the most significant candidates.

Although identifying candidate functions and their corresponding coefficient is an important advantage for FOS and represents a good motivation to use FOS in step length estimation. The “Fast” part of FOS abbreviation was completely ignored. Being a fast nonlinear system identification algorithm can be of great interest.

Online learning and personal customization can both be enabled using FOS owing to its speed compared to other nonlinear system identification techniques. User can choose to fine tune his/her own step length model by enabling online training while GNSS is available, which can further enhance the overall navigation solution by customizing the step length module for the current user.

In addition, online training and personal identification can be enabled automatically using step parameters for user identification as well. When same user is identified more than predefined threshold of number of times, an online learning is enabled, and whenever another user is detected (like a guest) using the same device, a default model is used instead.

Last but not least, due to FOS's low computation demand compared to ANN, and since ANN neither provide the only practical solution nor gives an easier or more efficient solution as mentioned in Chapter2. FOS is preferred over ANN in this research.

4.2.3 How FOS practically works?

The first step in FOS algorithm is the generation of a candidates' pool, this candidates' pool is the place where all possible model terms (candidates) are found and FOS algorithm search among them for the most relevant and effective terms to pick and build the final model.

The FOS algorithm requires a number of inputs from the user. The first set of inputs are for the generation of the candidates' pool, like the maximum delay allowed in the input signal(s), the maximum delay allowed in the output signal(s), and the maximum number of cross products allowed in a candidate (or a term in the final model).

For example, letting n be sample time, X be the input signal to a given system, Y be the corresponding output from the system, L be the maximum delay in input equals two, K be the maximum delay in output equals one, and M be the maximum number of cross products is two, the following set of candidates will represent the candidates' pool generated by the FOS algorithm [X(n), X(n-1), X(n-2), Y(n-1), X(n)X(n), X(n)X(n-1), …, X(n-1)Y(n-1), X(n-2)Y(n-1)]. Noting that the previous set of candidates does not include X(n-3) or X(n)X(n)X(n-1), because in the first term the delay is three while L (the max delay allowed in input) is set to two by user in this example, while the second term is not a legitimate candidate because M (the max number of cross products) is set to two while the number of cross products in this term is three.

The other set of inputs for the FOS algorithm are for defining the stopping condition, a maximum number of terms per model, a minimum mean square error (MSE) improvement per each additional term to the model, and finally, a minimum MSE for the overall model be reached combined represent the stopping condition for FOS, where whichever the first condition becomes true the algorithm stops.

Although the rest of the implementation of the FOS algorithm is a direct interpretation for the equations in section 4.2.1 and as explained in (Korenberg 1989, Korenberg and Paarmann 1989) into the programming language of choice. A verification step is important due to the nature of the problems FOS is trying to solve, where FOS is trying to identify nonlinear models which its user does not know beforehand.

FOS implementation can be verified against predefined nonlinear models. Two predefined models were used to test FOS implementation. The first model was represented in the form of difference equation (4-17). This model will be referred to by the first model from here onwards.

$$\begin{aligned} Y(n) = {} & 1 + 0.7\,X(n) + 0.8\,X(n-1) - 0.4\,X(n)\,X(n-1) + 0.1\,Y(n-2) \\ & + 0.2\,X(n-1)\,Y(n-1) \end{aligned} \tag{4-17}$$

The second model is an Linear Nonlinear (LN) Parallel Cascade as shown in the following figure, where g1 = (0.9, 0.7, 0.4), g2 = (0.8, 0.6, 0.5), a1 = (1, 0.3, 0.1, 0.02, 0.01), and a2 = (1, 0.4, 0.2, 0.1, 0.01, 0.01), and will be also referred to by the second model from here onwards.

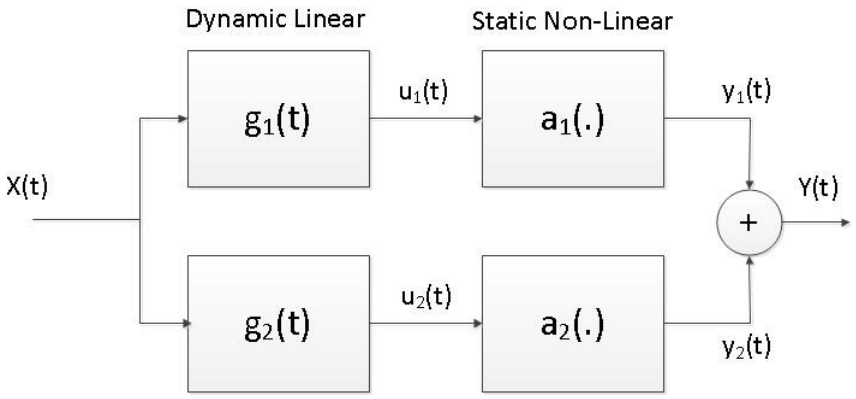


Figure 4-3: LN Parallel Cascade

One advantage of the first model over the second one is that it enable direct comparison of candidates picked by FOS and corresponding coefficient to the original model terms and corresponding coefficients. On the other hand, the second model represents a more challenging problem than the first model, and which is more close to the practical problems we are trying to solve, and where we will depend solely on MSE measure to pick the best model.

For the first model, Figure 4-4 shows a sample of a randomly generated input to the system and the corresponding output, and Figure 4-5 shows the probability distribution function for both the input and the output. The length of the input signal is 3000, where the first 1000 samples are used to build (train) different models with different L and K values (but M is constant equals two), the second 1000 samples were used to select a winner model (by cross validation), and the third and final 1000 samples were used to verify the result of the winner model is consistent.

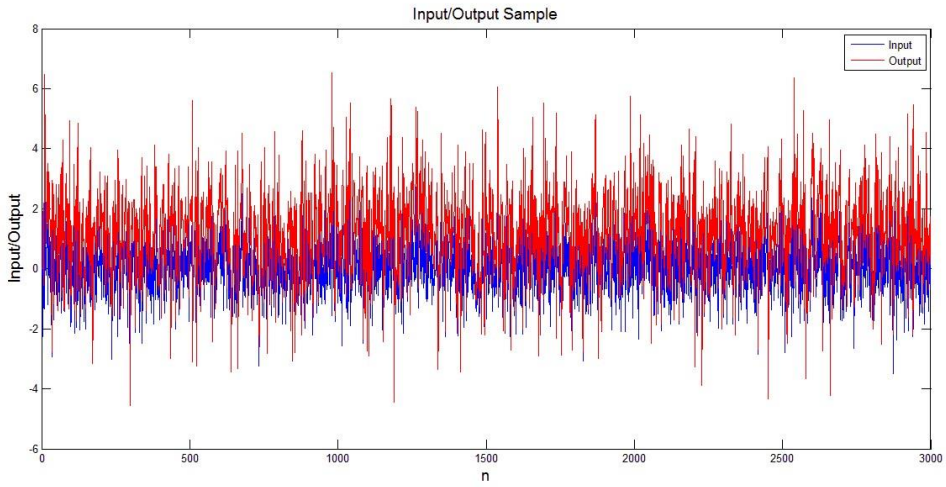


Figure 4-4: First Model - Input/Output Sample

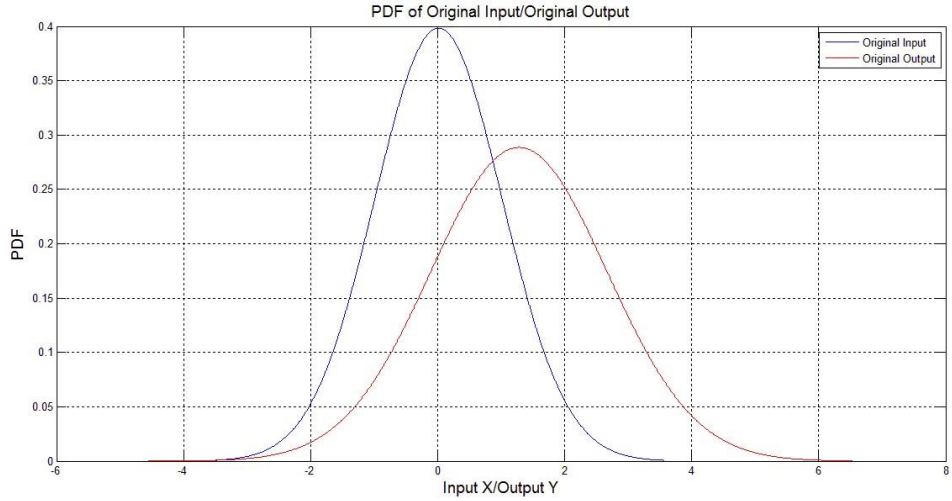


Figure 4-5: First Model - Input/Output Probability Distribution Functions

After running the FOS algorithm on the input and output signals shown in Figure 4-5 (the noise free signals), and for L and K values range from 5 to 30 with step 5. The output from the FOS algorithm is shown in Listing 1.

Listing 1: FOS output for the first model

```
%MSE of Model( i=1        L=5       K=5 ) = 0.0000 %
%MSE of Model( i=2        L=10      K=10) = 0.0000 %
```

```
%MSE of Model( i=3          L=10        K=15) = 0.0000 %
%MSE of Model( i=4          L=20        K=20) = 0.0000 %
%MSE of Model( i=5          L=20        K=25) = 0.0000 %
%MSE of Model( i=6          L=30        K=30) = 0.0000 %
%MSE of Model( i=7          L=30        K=20) = 0.0000 %
%MSE of Model( i=8          L=10        K=20) = 0.0000 %
%MSE of Model( i=9          L=10        K=30) = 0.0000 %
%MSE of Model( i=10         L=5         K=10) = 0.0000 %

The Winner Model is :       i=1

Number of Terms = 6

Best %MSE = 0.0000 %    (Second         interval )

Best %MSE = 0.0000 %    ( last          interval )

Best Model terms :

  1.0000
  0.8000 * X(n−1)
  0.7000 * X(n)
- 0.4000 * X(n) * X(n-1)
  0.2000 * X(n−1) *Y(n−1)
  0.1000 * Y(n−2)
```

Figure 4-6 shows the result of the winner model vs the original output, while Figure 4-7 shows a zoom in of the complete results shown in the previous figure. Finally, Figure 4-8 shows the change in the MSE of the generated model while adding each term to the model, and as demonstrated in the figure, FOS selected the best term first (which decreases the MSE most), then the second best one and so on. This figure was vital while the development of the code, as it can be interpreted as whether the calculation of the intermediate values ($\alpha_{mr}, D_{mm}, \ldots, etc.$) are correct or not.

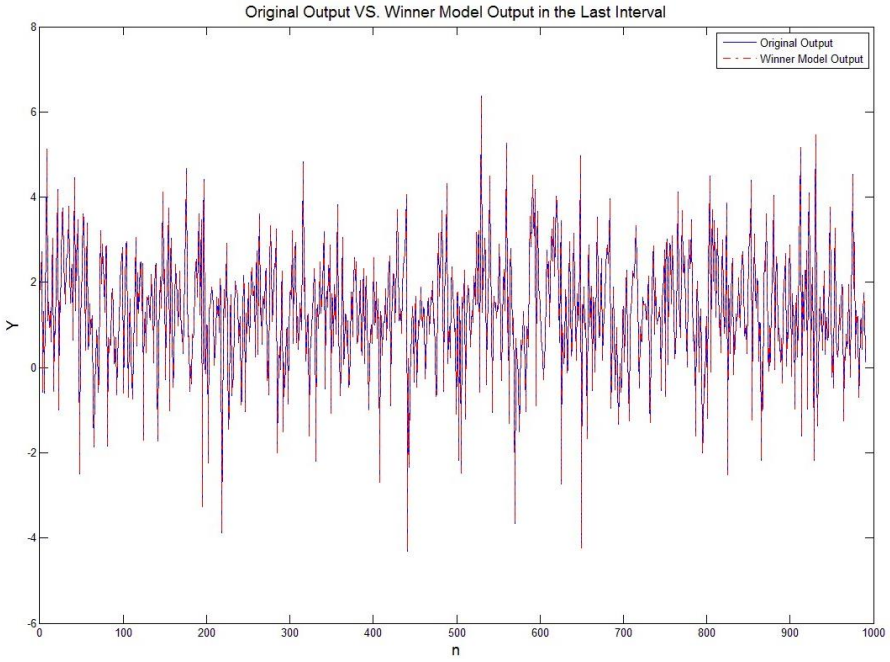


Figure 4-6: First Model - Output from the Winner Model in the Last Interval

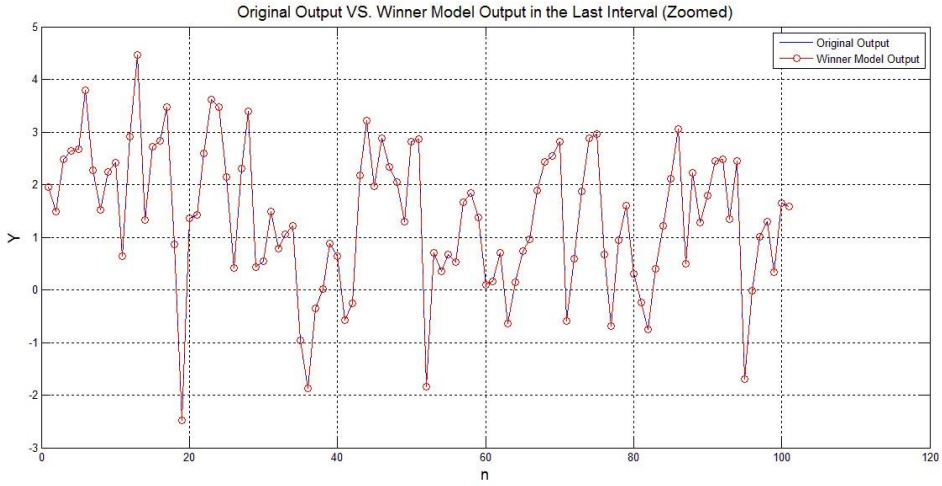


Figure 4-7: First Model - Output from the Winner Model in the Last Interval (Zoomed)

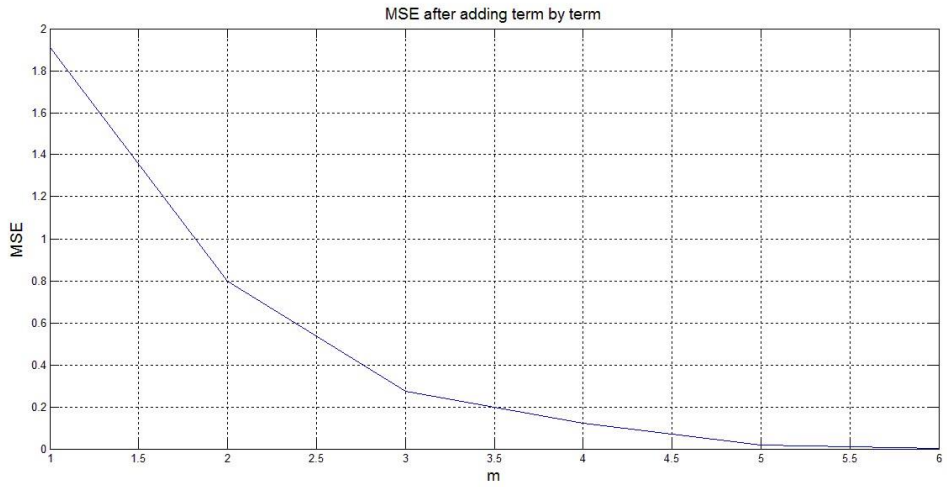


Figure 4-8: First Model - First Model MSE while adding term by term

For the second model, the output from the FOS algorithm is shown in Listing 2, and Figures 4-9 to 4-11 shows similar results as Figures 4-6 to 4-8. Figure 4-9 shows the result of the winner model vs the original output, while Figure 4-10 shows a zoom in of the complete results shown in the previous figure. Finally, Figure 4-11 shows the change in the MSE of the generated model while adding each term to the model.

Listing 2: FOS output for the second model

```
%MSE of Model( i=1          L=5         K=5) = 0.0881 %
%MSE of Model( i=2          L=10        K=10) = 0.0882 %
%MSE of Model( i=3          L=10        K=15) = 0.0884 %
%MSE of Model( i=4          L=20        K=20) = 0.0886 %
%MSE of Model( i=5          L=20        K=25) = 0.0889 %
%MSE of Model( i=6          L=30        K=30) = 0.0894 %
%MSE of Model( i=7          L=30        K=20) = 0.0894 %
%MSE of Model( i=8          L=10        K=20) = 0.0886 %
%MSE of Model( i=9          L=10        K=30) = 0.0894 %
%MSE of Model( i=10         L=5         K=10) = 0.0882 %

The Winner Model is :       i=1
Number of Terms = 11
Best %MSE = 0.0881 %        (Second     interval )
Best %MSE = 0.1080 %        ( last      interval )
Best Model terms
```

2.0302
0.5334 * X(n) * Y(n − 1)
0.4085 * X(n − 2) * X(n − 1)
0.3463 * X(n − 1) * X(n − 1)
- 0.3572 * X(n) * X(n - 3)
0.2672 * X(n − 2)
0.5868 * X(n − 1)
0.5354 * X(n) * X(n)
- 0.5975 * X(n)
- 0.1053 * X(n - 1) * Y(n - 1)
0.1133 * X(n − 2) * X(n − 2)

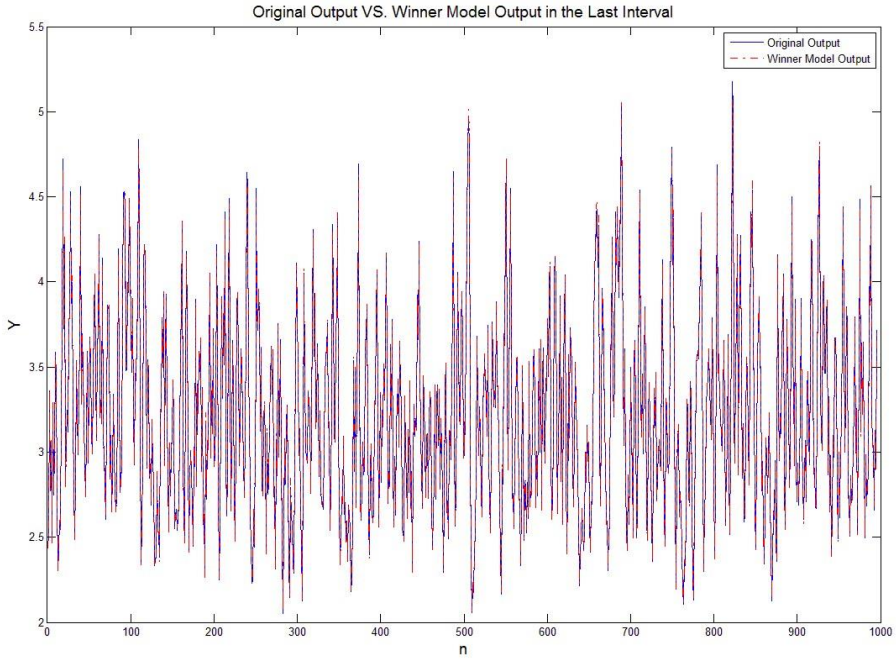


Figure 4-9: Second Model - Output from Model in the Last Interval

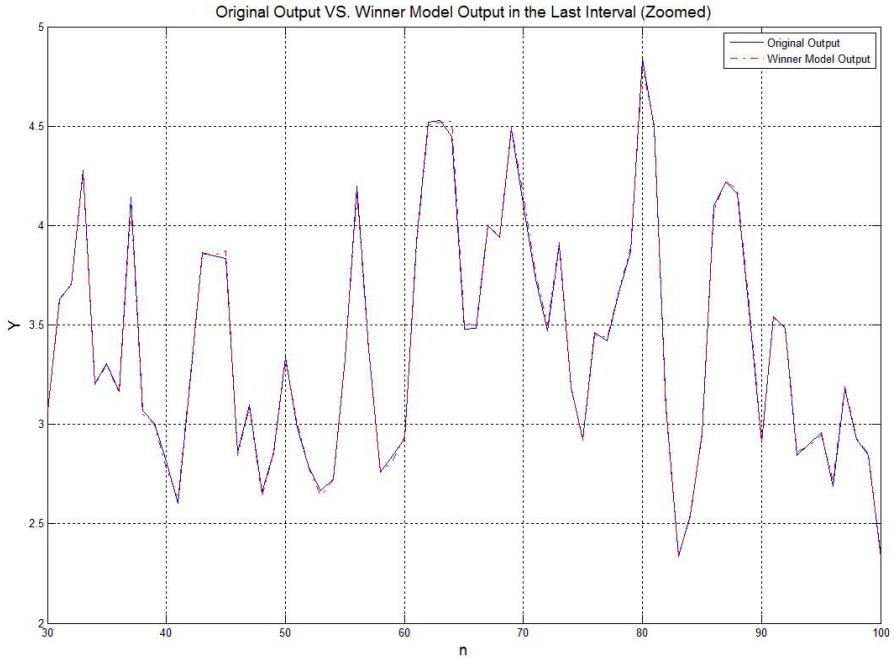


Figure 4-10: Second Model - Output from the Winner Model in the Last Interval (Zoomed)

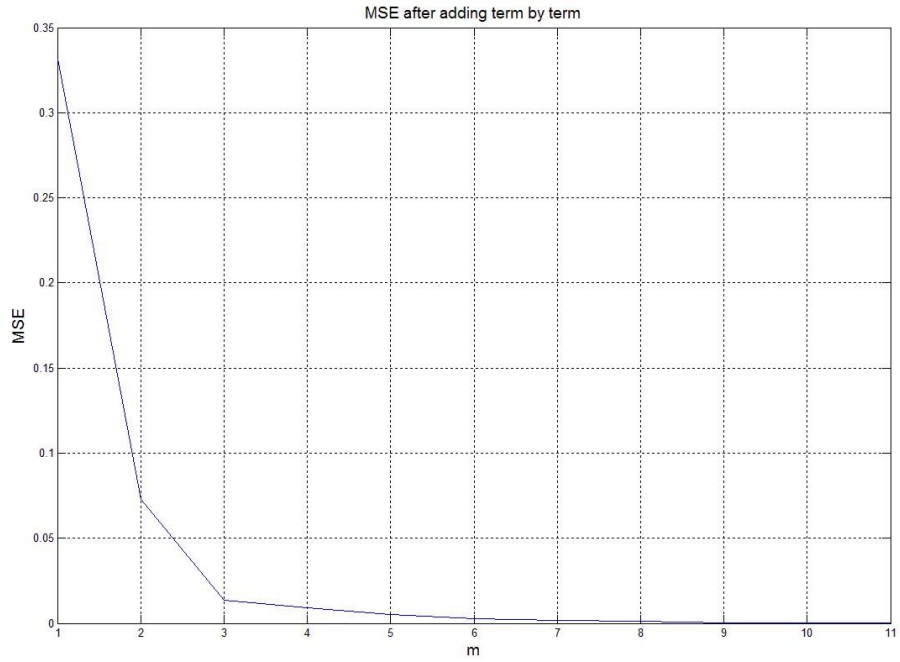


Figure 4-11: Second Model - MSE while adding term by term

4.2.4 Model Building Phase

In the step length estimation problem, the inputs to the model whether in the model-building phase (first phase) or the usage phase (second phase) are the different parameters that represent human motion dynamics together with other inputs derived from some or all of these parameters:

1. Different time-delayed versions of these parameters; and/or
2. Terms obtained by multiplying: (a) any/some/all of these parameters or their time-delayed versions by (b) any/some/all of these parameters or their time-delayed versions.

Either the vertical acceleration or the magnitude of the accelerometer readings can be used to detect steps and in calculating the parameters from the acceleration readings during the detected step. In the proposed method vertical acceleration is used.

Different parameters that represent human motion dynamics can be used in model building phase. They can be any combination of the following: step frequency, acceleration variance during step, acceleration peak value during step, peak to peak acceleration value during step, among others. During model-building, with each step detected, a reference step length using GPS and the used parameters that represent human motion dynamics are collected, stored and then fed to FOS. The nonlinear model proposed is:

$$\text{Step Length} = h(f, v) \tag{4-18}$$

where h is a nonlinear function of step frequency f and vertical acceleration variance v, and which is found using FOS as shown in sections 4.2.3. and can be expressed in its expansion form as:

$$\begin{aligned}\text{Step Length} = {} & \alpha_1 + \alpha_2 f + \alpha_3 v + \alpha_4 f_{prev} + \alpha_5 v_{prev} + \alpha_6 f.f + \alpha_7 f.v \\ & + \alpha_8 f.f_{prev} + \alpha_9 f.v_{prev} + \alpha_{10} f_{prev}.v + \alpha_{11} f_{prev}.f_{prev} \\ & + \alpha_{12} f_{prev}.v_{prev} + \alpha_{13} v.v + \alpha_{14} v.v_{prev} + \alpha_{15} v_{prev}.v_{prev}\end{aligned} \tag{4-19}$$

4.2.5 Model Usage Phase

After building the model, in the model usage phase, the same motion parameters used in building the model are fed to the final model except for the reference step length. In fact, the output of the model in the model usage phase is the estimated step length.

Figure 4-12 summarizes both, the model building and model usage phases.

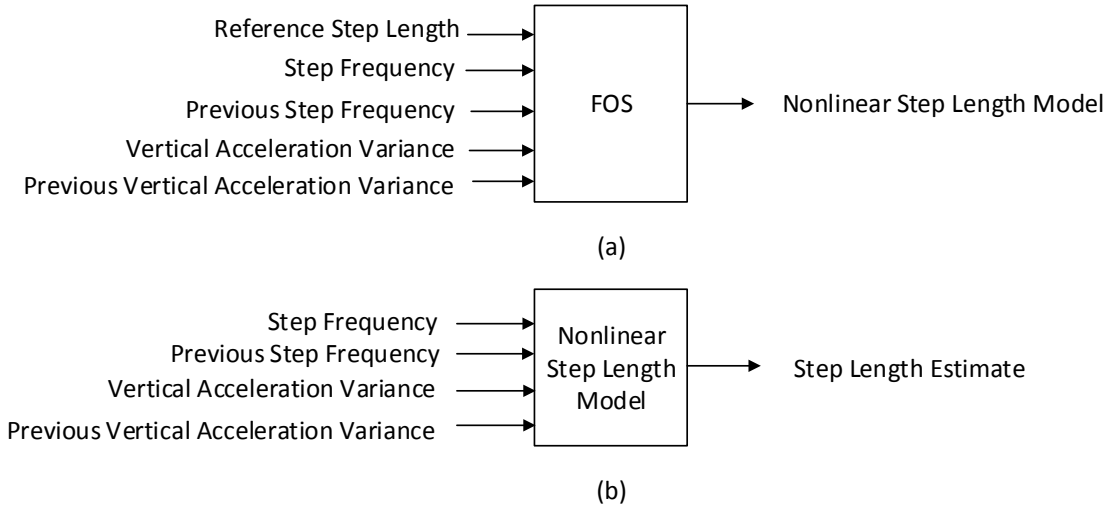


Figure 4-12: Summary of model building phase and model usage phase (a) model building phase (b) model usage phase

4.3 Using multiple FOS models

One further step which can enhance more the robustness of our approach is by following the divide and conquer paradigm, where the data can be divided according to users' speed during the model building phase, and instead of fitting one single model for all walking or running speeds, a model can be fit for each walking/running speed class instead. Since all experiments are done following a constant distance course or trajectory. Instead of using walking speed in clustering and classifying the data, the step length will be used for its measurement convenience than the walking speed measurement. In Figures 4-4 to 4-9, the relation between different parameters and the step length is plotted, both the 2D and 3D of each relation are illustrated.

Figure 4-13 shows the relation among current step frequency vs. previous step frequency vs. step length. The figure shows how difficult it is to define definite boundaries for any specific number of step length clusters or classes. Figure 4-14 furthers illustrates why no definite boundaries can be drawn to efficiently divide the data according to these previously mention parameters and the step length. Figure 4-14 is not showing only how the data is in-separable as shown in Figure 4-13 but also data is overlaid on each other, meaning that for certain current step frequency and previous step frequency values, there are more than one corresponding step length value, which make it even harder to use both current and previous step length frequency parameters as the only parameters to divide the training data according to step length values.

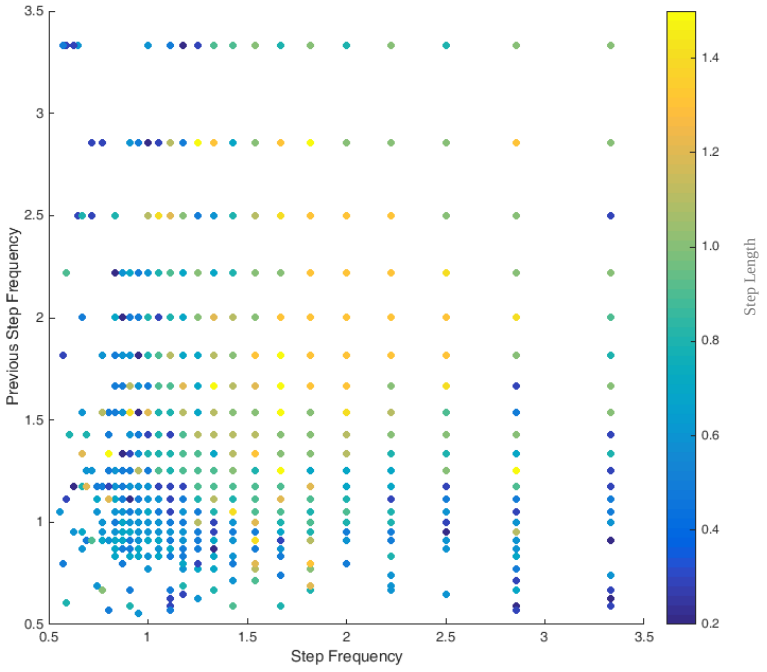


Figure 4-13: Step Frequency Vs. Previous Step Frequency

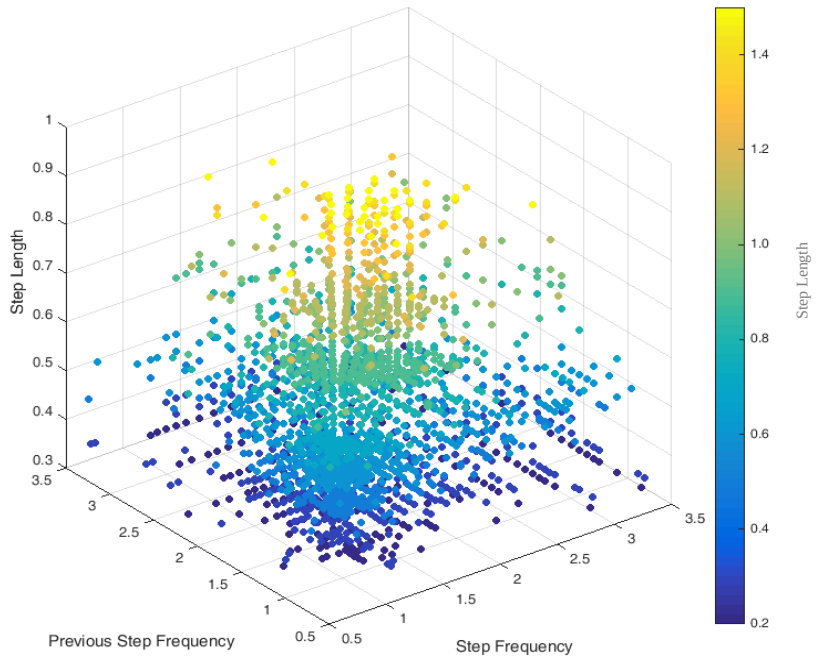


Figure 4-14: Step Frequency Vs. Previous Step Frequency Vs. Step Length

Figures 4-6 – 4-9 show similar results to Figures 4-4 and 4-5, but illustrating different pairs of motion parameters vs. step length. For example, Figures 4-6 and 4-7 are showing the relation among current vertical acceleration variance vs. previous vertical acceleration variance vs. step length, while Figures 4-8 and 4-9 are showing the relation among current step frequency vs. current vertical acceleration vs step length. The relation among previous step frequency vs. previous vertical acceleration vs. step length is ignored among the other permutations mixing between current step frequency and previous vertical acceleration variance and vice versa due to the similarity expected between these all permutations and the current step frequency vs current vertical acceleration variance shown in Figures 4-8 and 4-9.

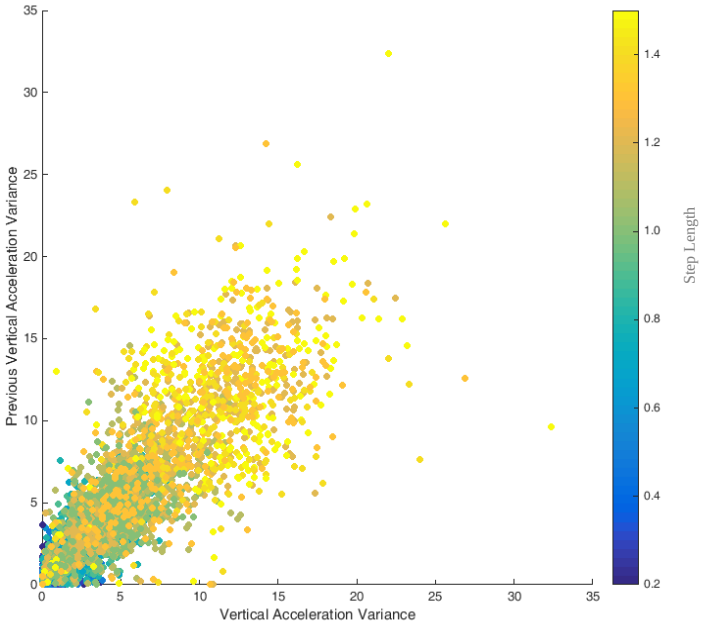


Figure 4-15: Vertical Acceleration Variance Vs. Previous Vertical Acceleration Variance

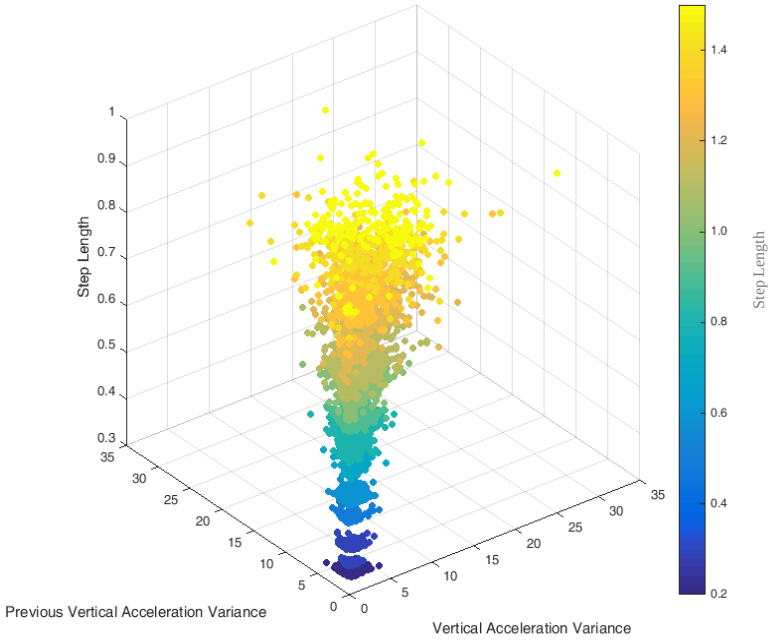


Figure 4-16: Vertical Acceleration Var. Vs. Previous Vertical Acceleration Var. Vs. Step Length

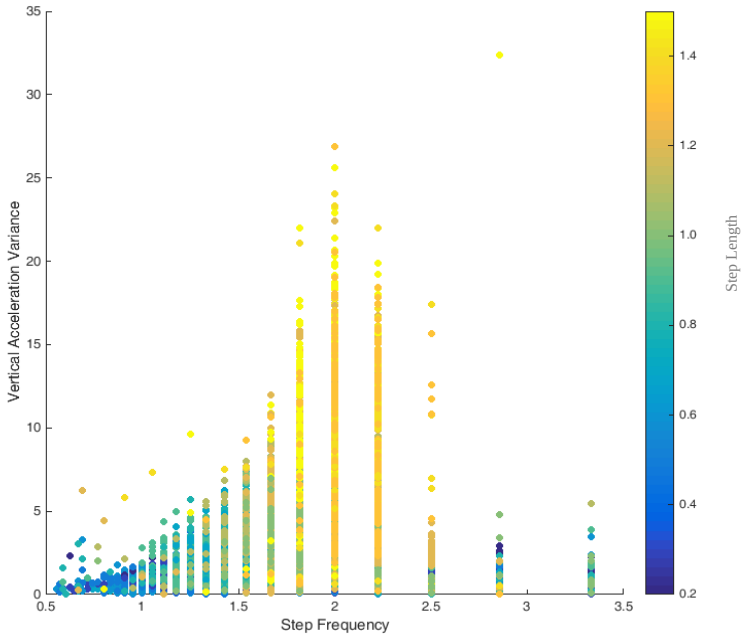


Figure 4-17: Step Frequency Vs. Vertical Acceleration Variance

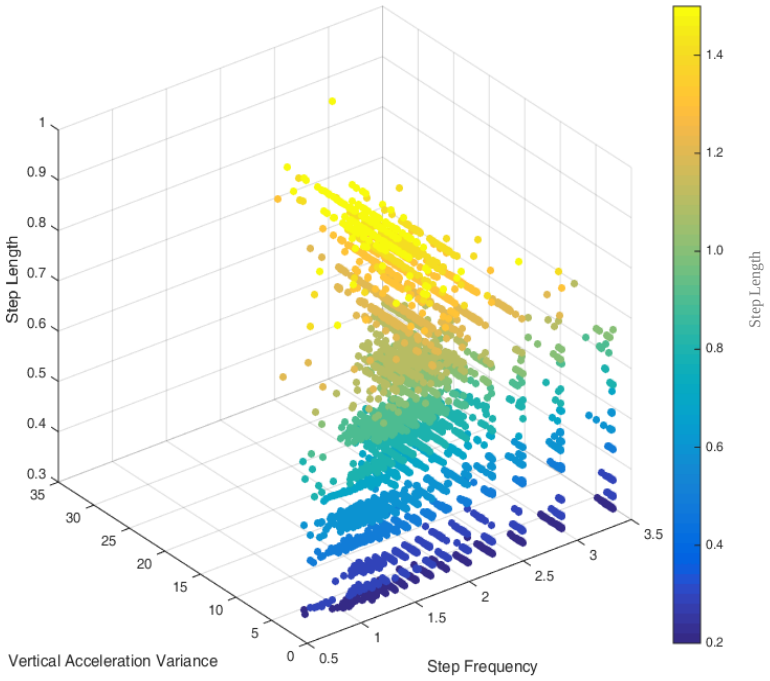


Figure 4-18: Step Frequency Vs. Vertical Acceleration Vs. Step Length

Figure 4-19 is different than the previous figures because it is plotting more than two motion parameters (in this case current step frequency, previous step frequency, and current vertical acceleration) vs step length. The other positive difference about Figure 4-19 is that it shows that our problem of dividing the data according to step length is feasible. Figure 4-19 shows that a radial basis function (RBF) neural network can solve this problem efficiently. However, due to the drawbacks of the neural networks, specifically computational power requirement, another approach is followed which is explained in more details in the following subsections.

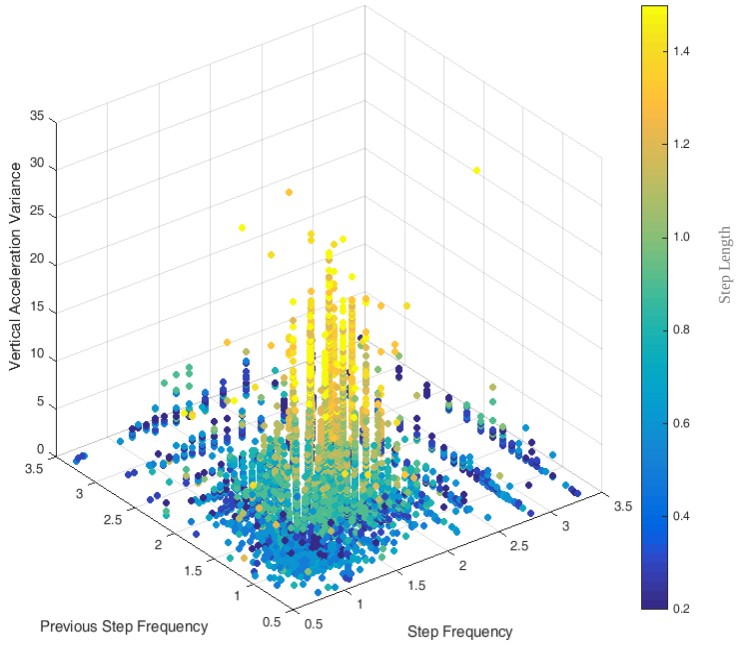


Figure 4-19: Step Frequency Vs. Previous Step Frequency Vs. Vertical Acceleration Variance

4.3.1 Using Machine Learning to cluster and classify the training data

From Figures 4-4 – 4-9, one can conclude that the problem of properly dividing the data according to step length is very difficult, if not impossible, if two motion parameters are involved.

In addition, this data division problem gets slightly easier when more than two parameters were involved as shown in Figure 4-19. Instead of continuing manually investigating the relation among three and plus motion parameters vs the step length, here comes the rule of machine learning, since adding more motion parameters means adding more dimensions to the problem. A set of experiments were done examining first classifying the data using three and more motion parameters. Using decision tree algorithm to solve the classification problem instead of using Neural Networks due to its computation power requirement. The problem remains unsolvable due to the limited capabilities of Decision Tree classifiers compared to Neural Networks.

One more step that can be taken before giving up solving this problem using decision tree classifying algorithm is by involving the step length itself in the classification process. However, since step length will not be available at run time, i.e. when the classification decision is needed. In fact, it quite the opposite, the classification decision is needed first in order to choose the most appropriate step length model to calculate the step length. That is why using a clustering algorithm (k-means clustering algorithm in our case) before building the classifier is proposed.

In both steps of clustering and classification, the MATLAB® functions kmeans and ClassificationTree.fit were used respectively. The default values for all optional inputs for these two functions were used, i.e. using the squared Euclidean distance as distance measure and 100 maximum number of iterations with the kmeans algorithm described in (Lloyd 1982). In addition, the classification algorithm used is explained in more detail in (Breiman, Friedman et al. 1984).

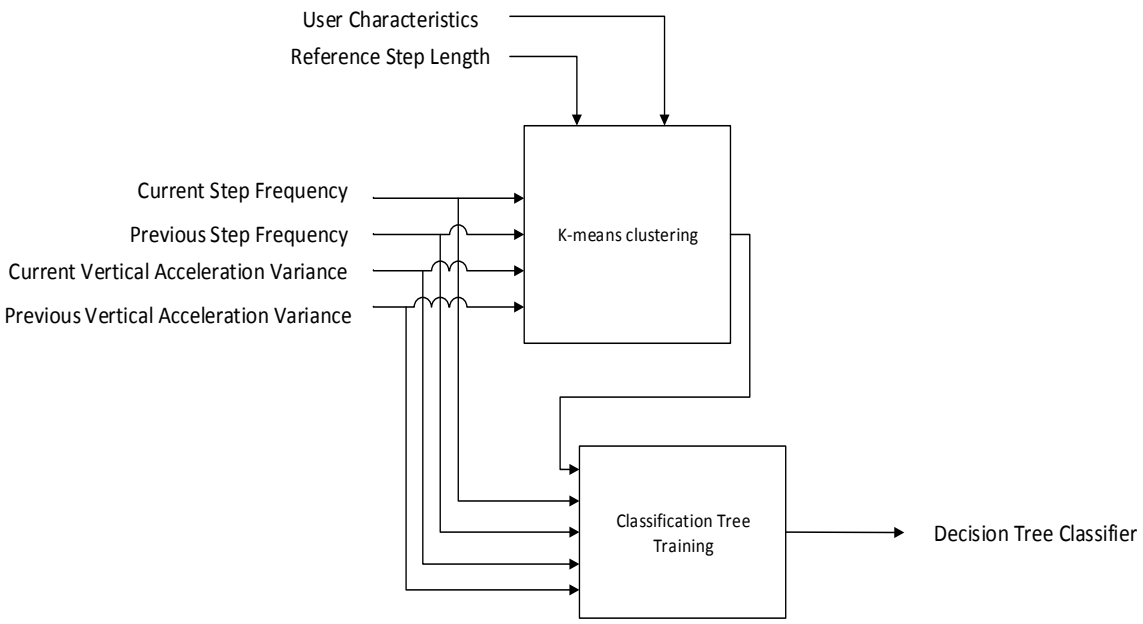


Figure 4-20: Decision Tree Classifier Training Process

Using the clustering algorithm before the classifier training phase is like compressing the information found in the input(s) to the clustering algorithm, and training the classifier to decompress this information using a subset of the input(s) used in the clustering phase. In fact, after applying this approach, the most successful classifier will be after using current and previous step frequency, current and previous vertical acceleration variance, step length, and one last input represent a collective characteristics for users collected the training data. These inputs were fed to a k-means clustering algorithm, which its output was used as the label (class) for the decision tree classification algorithm, while inputting just both current and previous step frequency and vertical acceleration variance. Figure 4-20 summarizes how a classification tree was built to divide the training data in the model building phase, and used later on to select the most appropriate model in the model usage phase at run time, while Figure 4-21 shows an example decision tree, where f , f_{prev} and v are the step frequency, previous step frequency, and vertical acceleration variance

respectively and which represents the inputs to the decision tree, whereas $\omega_1, \omega_2,$ and ω_3 are the parameters to be found using the decision tree classification algorithm.

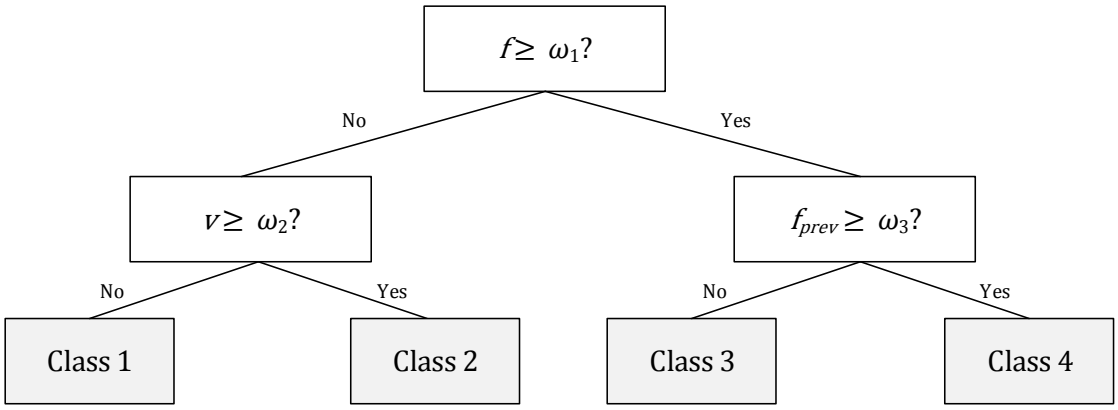


Figure 4-21: Example decision tree

An example multiple FOS model from this step is shown in Listing 3.

Listing 3: FOS output for the PDR solution

if(user_class == Class 1){
Step Length $= \alpha_{1,1} + \alpha_{1,2}f + \alpha_{1,3}v + \alpha_{1,4}f_{prev} + \alpha_{1,5}v_{prev} + \alpha_{1,6}f.f + \alpha_{1,7}f.v + \alpha_{1,8}f.f_{prev} + \alpha_{1,9}f.v_{prev} + \alpha_{1,10}f_{prev}.v + \alpha_{1,11}f_{prev}.f_{prev} + \alpha_{1,12}f_{prev}.v_{prev} + \alpha_{1,13}v.v + \alpha_{1,14}v.v_{prev} + \alpha_{1,15}v_{prev}.v_{prev}$;

} else if (user_class == Class 2){
Step Length $= \alpha_{2,1} + \alpha_{2,2}f + \alpha_{2,3}v + \alpha_{2,4}f_{prev} + \alpha_{2,5}v_{prev} + \alpha_{2,6}f.f + \alpha_{2,7}f.v + \alpha_{2,8}f.f_{prev} + \alpha_{2,9}f.v_{prev} + \alpha_{2,10}f_{prev}.v + \alpha_{2,11}f_{prev}.f_{prev} + \alpha_{2,12}f_{prev}.v_{prev} + \alpha_{2,13}v.v + \alpha_{2,14}v.v_{prev} + \alpha_{2,15}v_{prev}.v_{prev}$;

} else if (user_class == Class 3){
Step Length $= \alpha_{3,1} + \alpha_{3,2}f + \alpha_{3,3}v + \alpha_{3,4}f_{prev} + \alpha_{3,5}v_{prev} + \alpha_{3,6}f.f + \alpha_{3,7}f.v + \alpha_{3,8}f.f_{prev} + \alpha_{3,9}f.v_{prev} + \alpha_{3,10}f_{prev}.v + \alpha_{3,11}f_{prev}.f_{prev} + \alpha_{3,12}f_{prev}.v_{prev} + \alpha_{3,13}v.v + \alpha_{3,14}v.v_{prev} + \alpha_{3,15}v_{prev}.v_{prev}$;

} else if (user_class == Class 4){
Step Length $= \alpha_{4,1} + \alpha_{4,2}f + \alpha_{4,3}v + \alpha_{4,4}f_{prev} + \alpha_{4,5}v_{prev} + \alpha_{4,6}f.f + \alpha_{4,7}f.v + \alpha_{4,8}f.f_{prev} + \alpha_{4,9}f.v_{prev} + \alpha_{4,10}f_{prev}.v + \alpha_{4,11}f_{prev}.f_{prev} + \alpha_{4,12}f_{prev}.v_{prev} + \alpha_{4,13}v.v + \alpha_{4,14}v.v_{prev} + \alpha_{4,15}v_{prev}.v_{prev}$;
}

Figure 4-22 shows both building and usage phases of the nonlinear step length estimation model after adding the decision tree classifier generated in the previous step.

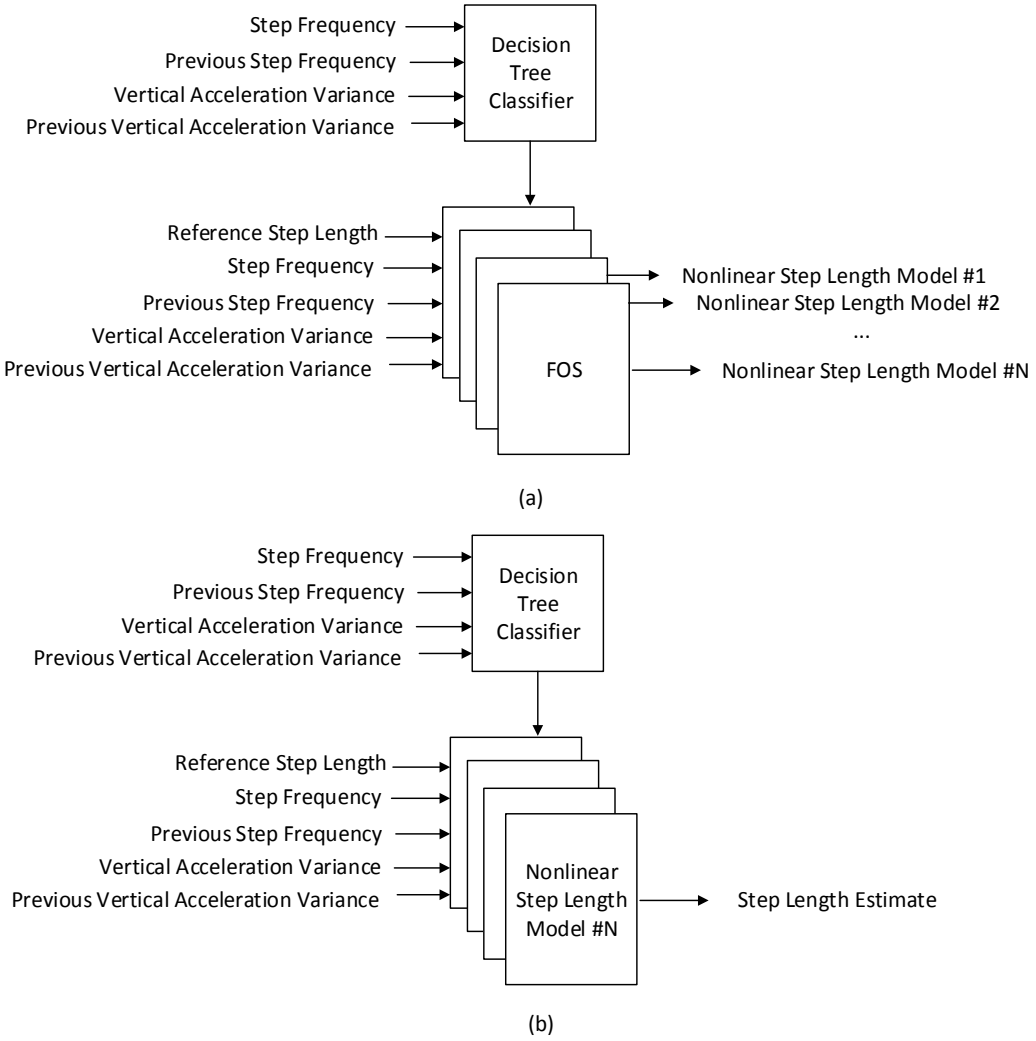


Figure 4-22: Summary of new model building phase and model usage phase after using the previously generated decision tree classifier (a) model building phase (b) model usage phase

In order to avoid redundancy, the results of the multiple FOS models enhancement are shown with the results of the multiple sensor assemblies' enhancement in the next section.

4.4 Using multiple sensor assemblies

The aim of this section is to show how using multiple sensor assemblies can additionally enhance varying step length estimation. The multiple sensors assemblies are not necessarily all on the same device but within the same user range and connected to the same communication network. The device including any of the participating sensor assemblies can be tethered as it is the case of smart belt clips or smart watches, or non-tethered as it is the case of smartphones.

In the following example architectures, the enhancement to the varying step length estimation are achieved through fusing sensor assemblies' data at different levels and/or using different estimators or filters. The following examples can be seen as special cases of the most general concept illustrated in Figure 4-23.

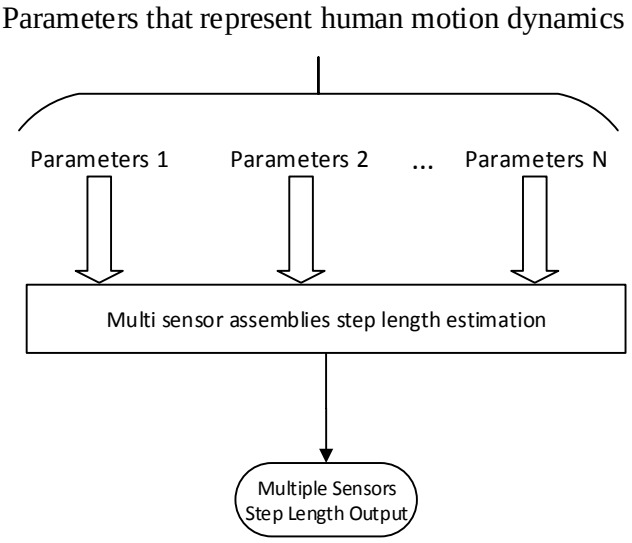


Figure 4-23: Multi-sensor assemblies step length estimation concept idea

Four example architectures which enables enhancing the varying step length estimation using multiple sensor assemblies are discussed here. In the first architecture, the enhancement to the varying step length estimation is achieved through fusing different outputs coming from different step length estimators, each step length estimator was fed in by a set of inputs from a

sensor assembly found in same device or found in different devices, this enhanced step length estimate represents the final step length estimate for all sensor assemblies.

In the second architecture, the enhancement to the varying step length estimation is achieved through fusing different inputs coming from different sensor assemblies found in same device or found in different devices but having similar measurements from sensors, then input the enhanced result as a single input to a step length estimator, this step length estimator outputs the final step length estimate for all sensor assemblies.

In the third architecture, the enhancement to the varying step length estimation is achieved through multi-level fusion architecture, where the first level of fusion takes place at the level of inputs, where the similar inputs coming from different sensor assemblies are grouped and fused together, then feed the output of the first fusion level in addition to the rest of the (different) inputs to corresponding step length estimator, then finally, at the output level, fuse all the outputs coming from each step length estimator. This final enhanced step length estimate represents the final step length estimate for all sensor assemblies.

In the fourth architecture and the final one, the enhancement to the varying step length estimation is achieved through a centralized filter, where all inputs coming from different sensor assemblies plus the relation between these sensor assemblies measurements plus any other statistic about these inputs (such as mean or variance) are fed to the centralized filter to estimate the final enhanced step length estimate.

There are two more ways to further enhance the varying step length estimation using multiple sensor assemblies; first is through enhancing the step detection, and second through enhancing the step frequency. Enhancing step detection can be done for example through a voting process, where for example in a set of three devices, two devices detected a step while the third one did not detect a step, according to the proposed method, a step detection event is forced in the third device, so that this device not to miss this step.

In addition, enhancing step frequency can be done for example through using an inverse function which calculates the step frequency given the enhanced step length calculated using one of the previous architectures (or method(s)). This more accurate step frequency can be used later on in the calculation of the next step length estimate.

4.4.1 Output Level Weighted Average Fusion Architecture

In this architecture, the enhancement to the varying step length estimation is achieved through fusing at the output level, i.e. fusing different outputs coming from different step length estimators, each step length estimator was feed in by a set of inputs from a sensor assembly found in same device or found in different devices as shown in Figure 4-24.

One advantage for this architecture is that it does not require sensors to be on same device, or having similar measurements. One example where sensors can be attached to the same platform (the user in this case) and not reading the same measurements is a set of sensor assemblies where the first sensor assembly is in a smartwatch on the user's arm or a smartphone in his/her pocket while the second sensor assembly is in a smart glasses on face or in a fitness appcessory on chest. If the inputs to the step length estimator are motion frequency and acceleration variance. In this case each sensor assembly will have different measurement depending on where the sensor assembly is attached to the user body, the motion frequency measured at arm or leg will be different than that measured at head or chest.

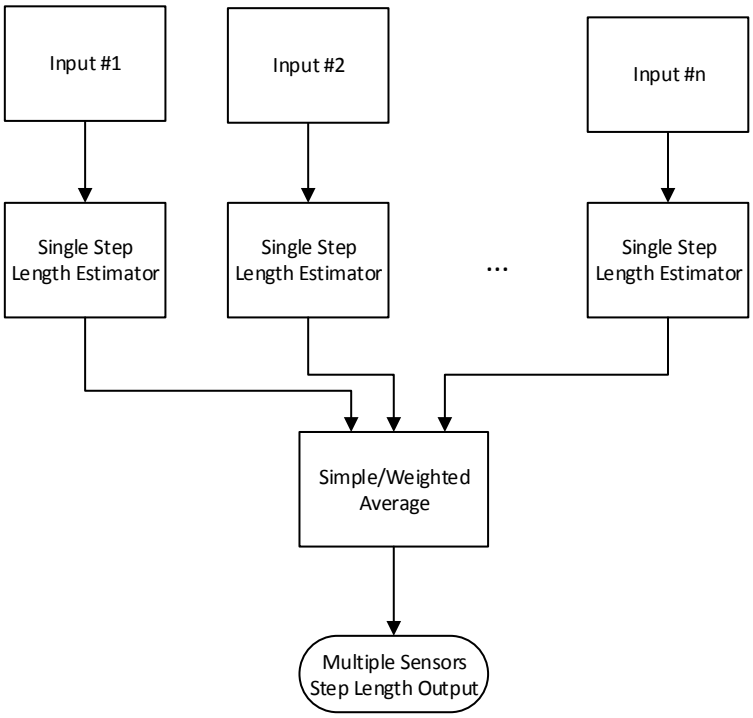


Figure 4-24: Outputs-level fusion architecture

Simple average can be applied at the output level when there are no available information about the accuracy of each step length estimator. Whereas, when there are some information about the accuracy of each step length estimator (such as a priori or predefined weight depending on sensor/device type and/or position and orientation of device relative to the platform), the weighted average is more preferable as shown in the results section.

The simple/weighted average can be replaced by any other machine learning or system identification driven model such as artificial neural network or using some optimization algorithm to find sensors/devices weights to better infer the best inputs for the step length estimator. However, for seeking lower computation resources, the simple/weighted average are used here.

4.4.2 Input Level Weighted Average Fusion Architecture

In this architecture, the enhancement to the varying step length estimation is achieved through fusing sensor assemblies' data at the input level. A simple or weighted average is applied to the different inputs coming from different sensor assemblies found in same device or found in different devices but having similar measurements from sensors. The weighted average input is fed to a step length estimator, this step length estimator outputs the final step length estimate for all sensor assemblies as illustrated in Figure 4-25.

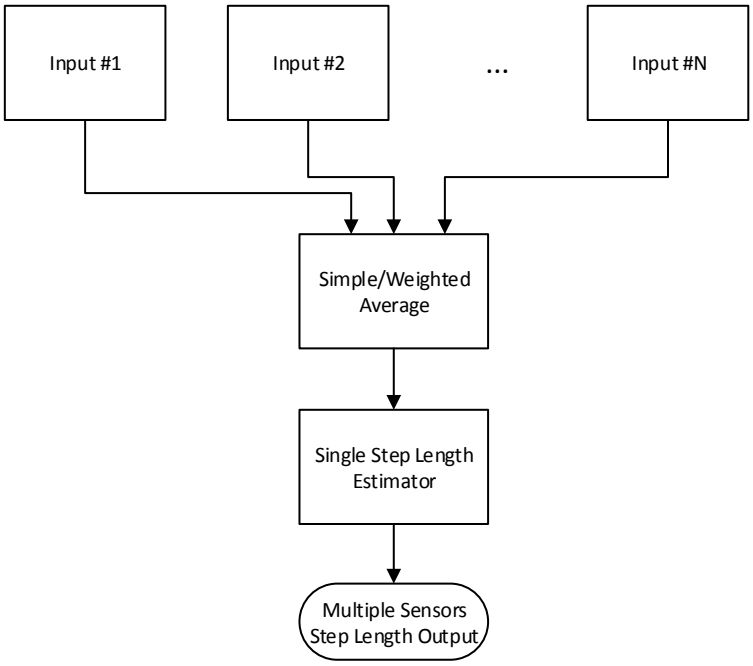


Figure 4-25: Inputs-level fusion architecture

It is worth to mention here that, in order to use this fusion architecture, the inputs must be similar. This can be achieved either when the sensors are fixed on the same device, or on different devices but measuring similar quantities such as step frequency.

Simple average can be applied when there are no available information about the accuracy of sensor assemblies used in the solution. Whereas, when there are some information about the

accuracy of sensor assemblies (such as variance or sensor/device type and/or position and orientation of the device relative to the platform), the weighted average is more preferable.

The simple/weighted average again can be replaced by any other machine learning or system identification driven model such as artificial neural network or using some optimization algorithm to find sensors/devices weights to better infer the best inputs for the step length estimator. However, for seeking lower computation resources, the simple/weighted average is most recommended in portable navigation applications.

4.4.3 Multi-Level Weighted Average Fusion Architecture

In this architecture, the enhancement to the varying step length estimation is achieved through multi-level fusion architecture, where on the first level of fusion takes place at the level of inputs, where the similar inputs coming from different sensor assemblies are grouped and fused together, then feed the output of the first fusion level in addition to the rest (different) inputs to a corresponding step length estimator, then finally fuse the output coming from each step length estimator (fusion at the output level). This final enhanced (fused) step length estimate represents the final step length estimate for all sensor assemblies as illustrated in Figure 4-26.

An example on similar measurements (inputs) which can be grouped together and fused using either simple/weighted average or any other estimation technique is step frequency of the user, since the user will not have different step frequencies according to where the sensor assembly is attached to his/her body. Whereas, an example on different measurements can be acceleration variance, where an accelerometer sensor attached to the arm or the leg of the user will measure a higher acceleration variance signal than another sensor attached to the head or the chest of the user.

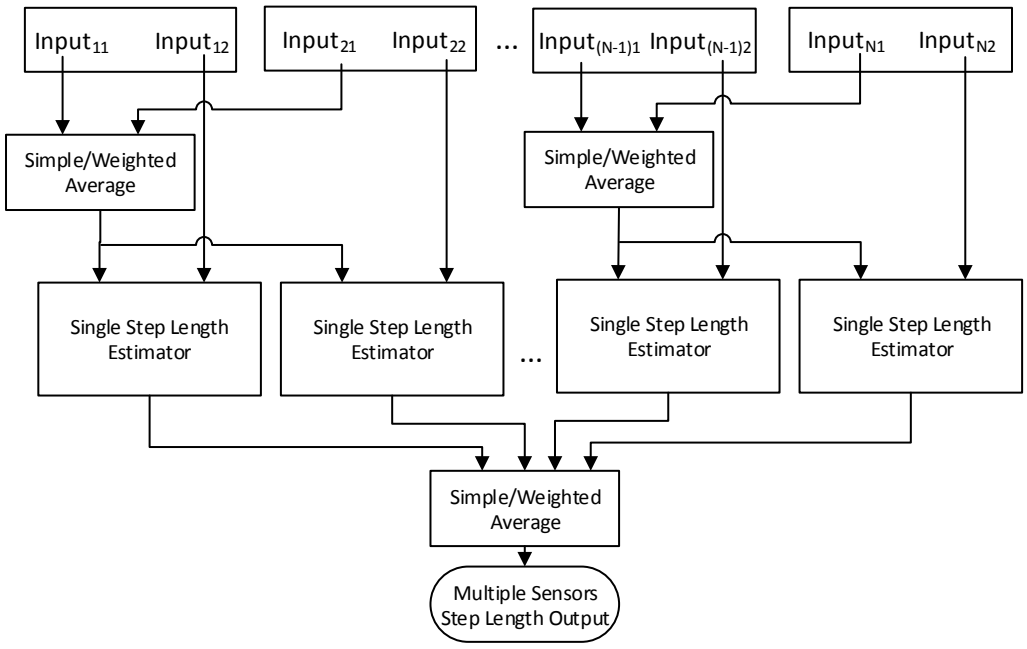


Figure 4-26: Multi-level fusion architecture

4.4.4 Centralized Filter Fusion Architecture

In this architecture, the enhancement to the varying step length estimation is achieved through a centralized filter, where all inputs coming from different sensor assemblies plus the relation between these sensor assemblies measurements (readings) plus any other statistic about these inputs (such as mean or variance) are fed to the centralized filter to estimate the final enhanced (fused) step length estimate as illustrated in Figure 4-27.

The centralized filter fusion architecture (or tightly coupled architecture) is the best architecture in terms of taking into consideration the inter relationships among the sensor assemblies and each other. However, it is the most complex one, which led to lower flexibility (scalability) than the other architectures and higher computation demand, and according to the computation and power consumption requirements by the portable and wearable devices, this architecture can be impractical.

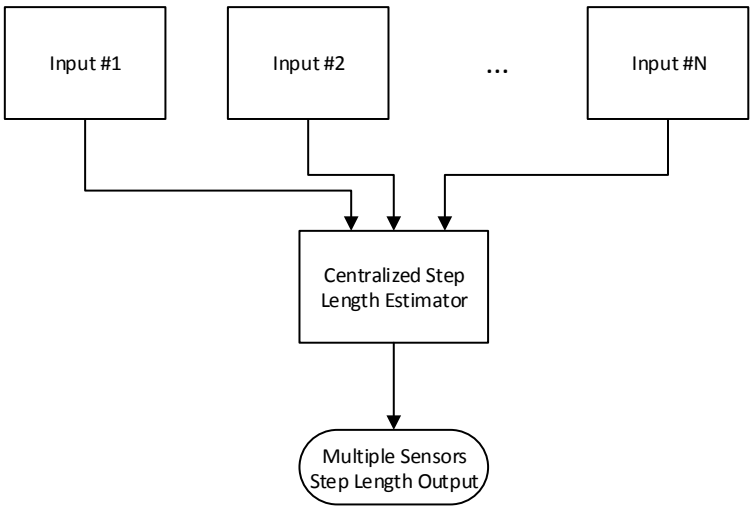


Figure 4-27: Centralized Filter Fusion Architecture

4.5 Further Enhancements

4.5.1 Using Scoring for Weighting

One implementation for weighting participating sensor assemblies' output is hard coding the weights in the real-time code using simple if-else statements, checking on hosting device type (and usage case or orientation) and select the proper set of weights for the current sensor assemblies. Another approach is through using a look up table, where different weights are set for devices with different types and usage cases and/or orientations.

Although, the two previously mentioned approaches are computationally efficient, they are not flexible enough to add or learn online new weighting sets for new devices types and/or usage case and orientation. One method that can be sufficiently flexible for this purpose is using scoring for weighting instead of constant weights hardcoded in the code or stored in a look table.

Scoring can be done individually for each device and usage case or orientation alone, and when multiple sensor assemblies (or their hosting devices) connect together, the scores are

summed, and each score is divided by the total sum becomes the weight of this sensor assembly in the final solution.

One advantage of scoring is that, there is no need to cover all possible combinations of all possible participating devices and their usage modes and/or orientation. It is just enough to cover each possible device and usage case or orientation, and set some default value just in case and newer device or usage case is used. More importantly, scoring can easily enable online training when there is a reference for step length such as GNSS for example, where in this case, the score of a given device increase when its step length estimate is close enough to the reference step length, and decreases otherwise.

4.5.2 Using Dynamic Weighting Scheme

There are two more ways to further enhance the multiple sensor assemblies' solution or the overall navigation solution. The simplest is through selecting the best device at a time. For example, when there is a smart glasses available just use it, otherwise, when phone in handheld view use case and smart watch in dangling use case use the weighted average solution.

The second method is by following similar approach to what was followed in multiple FOS models section, which is divide and conquer approach. On analyzing the results by user instead of by device or usage mode, it was noticed that for a given set of weights for each device and/or usage mode, the multiple sensor assemblies' solution was better than each device all solution 100% of the time. Therefore, by assigning different weighting sets for each device and/or usage case for each group of similar users, the result of the overall multiple sensors assemblies' performance can be significantly improved. Again, this can be done through training a classifier, inputting to the classifier the motion parameters of each group of users found to be similar in performance to each other and a class id for this group. Afterwards, in the usage phase in real time, the output classifier from the training phase can be used to classify the input motion parameters for a given user and use the most appropriate weighting set for the participating devices, and obtain an overall better

solution than each device alone. Again, if identifying similar groups of user is difficult, a clustering algorithm can be used before the classification training phase.

4.6 Results

4.6.1 Single Device Step Length Experimental results

4.6.1.1 Varying step length estimation for Walking with different speeds

In this experiment for varying speed walking, during the model-building phase the data set was collected by 17 different subjects. Different physical characteristics are considered like height, weight, gender, and age. Each subject collected data with four different walking speeds – very fast, fast, normal, and slow. These speeds are relative to each user, but the distance covered is constant, and the tester calculates the duration of each trajectory and make sure about labeling the trajectory with the right speed. Furthermore, each one of the subjects collected two datasets in each speed category and for each device use case (e.g. handheld, dangling, pocket), in order to have an abundance of datasets for verification. The datasets used for model-building included four datasets for each subject (each dataset with one of the aforementioned speeds, this means one very fast, one fast, one normal, and one slow). One third of subjects were not included in the model building and were left solely for validation.

Three rectangular walking trajectories among those collected are presented below where GPS is used as a reference as shown in Figure 4-28, Figure 4-29, and Figure 4-30.

The FOS model results were compared against a linear regression model generated using the MATLAB® function regress and which is explained in more details in (Shin, Park et al. 2007).The goal of this comparison is to test whether a nonlinear model is needed or the linear model proposed in (Shin, Park et al. 2007) and which is commonly used in literature can provide an accurate model.

The linear model used and which is explained in more details in (Shin, Park et al. 2007) is:

$$\text{Step Length} = \alpha.f + \beta.v + \gamma \quad (4\text{-}20)$$

where f and v are the step frequency and vertical acceleration variance respectively, whereas $\alpha, \beta, \text{and } \gamma$ are the parameters to be found using the regression algorithm. The nonlinear model proposed is:

$$\text{Step Length} = h(f, v) \quad (4\text{-}21)$$

where h is a nonlinear function of step frequency f and vertical acceleration variance v, and which is found using FOS as shown in sections 4.2.3 and 4.2.4.

As mentioned before in section 4.2.4., during the model building phase, with each step detected, a reference step length using GPS, step frequency, and vertical acceleration variance are collected. Same data used in training FOS model were also used to train the linear regression model for fairness. The results shown demonstrate how the step length estimated from the nonlinear model generated by FOS is more accurate than the one estimated from linear regression model. The difference in performance of the FOS model vs the linear regression model can be explained due to the false assumption of linear relationship between the motion parameters and step length as assumed in the literature.

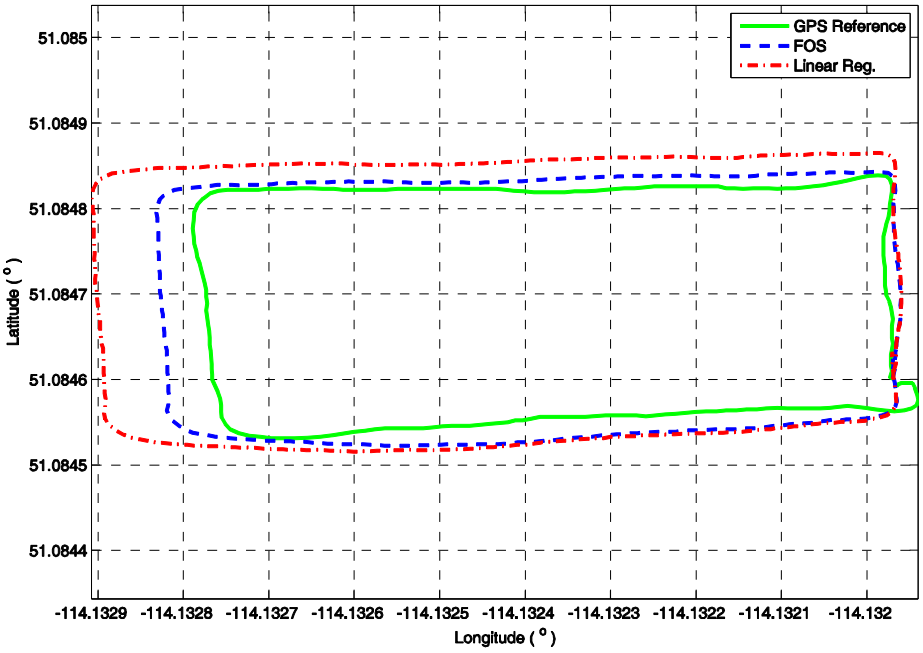


Figure 4-28: Slow walking speed trajectory

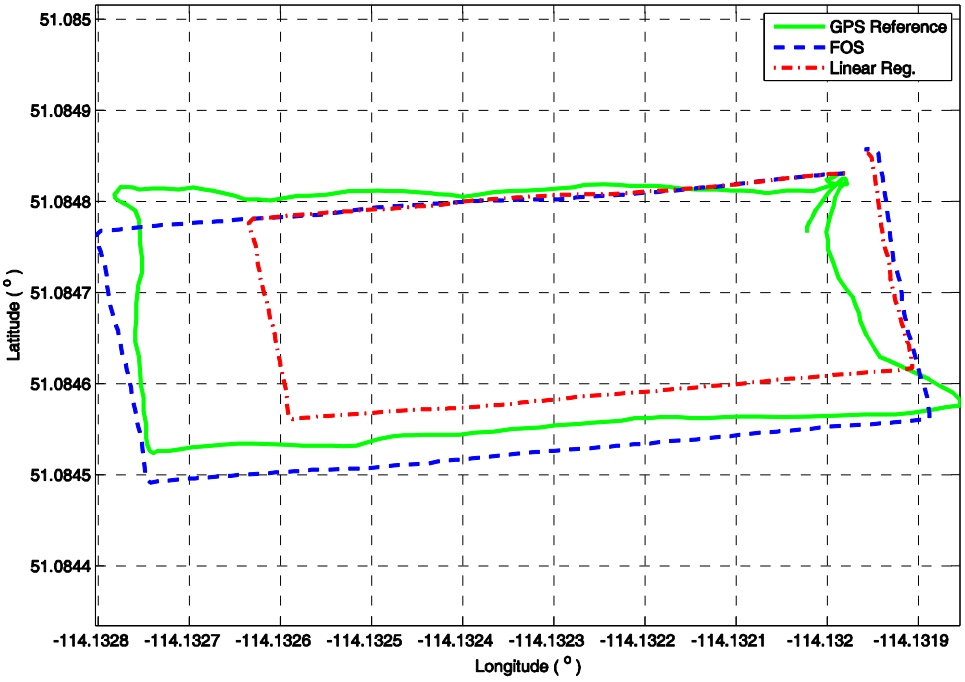


Figure 4-29: Normal walking speed trajectory

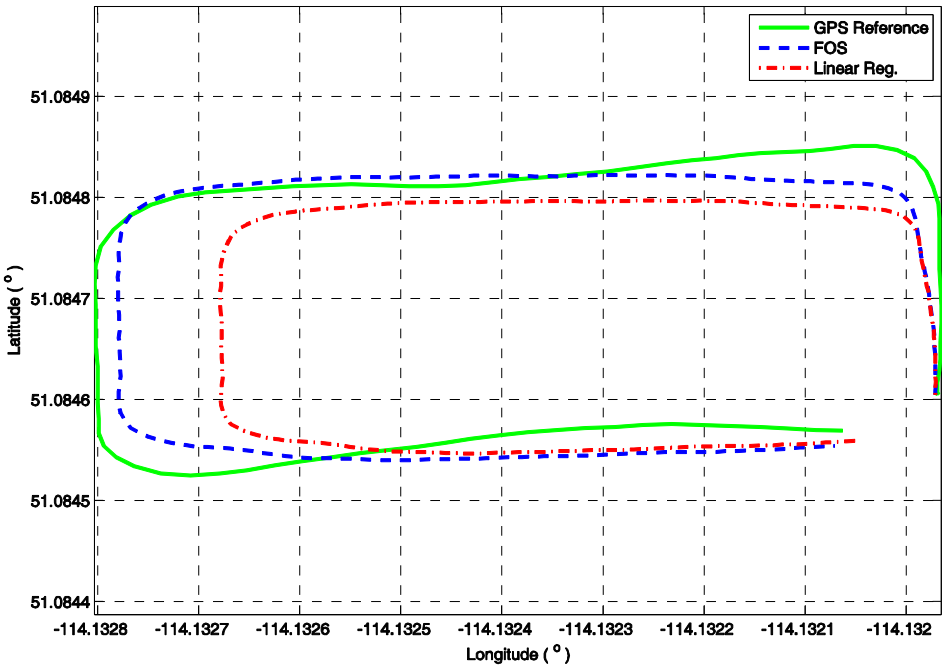


Figure 4-30: Fast walking speed trajectory

4.6.1.2 Varying step length estimation for Running with different speeds

In this experiment for varying speed running, during the model-building phase the data set was collected by the above mentioned subjects. Six running speeds were collected: extreme fast, very fast, fast, normal, slow, and very slow. Similar to walking experiment, each one of the subjects collected several datasets in each speed category, in order to have an abundance of datasets for verification. The datasets used for model-building included six datasets for each subject (each dataset with one of the aforementioned speeds). One third of subjects were not included in the model building and were left solely for validation.

Three straight line running trajectories among those collected are presented below where GPS is used as a reference as shown in Figure 4-31, Figure 4-32, and Figure 4-33.

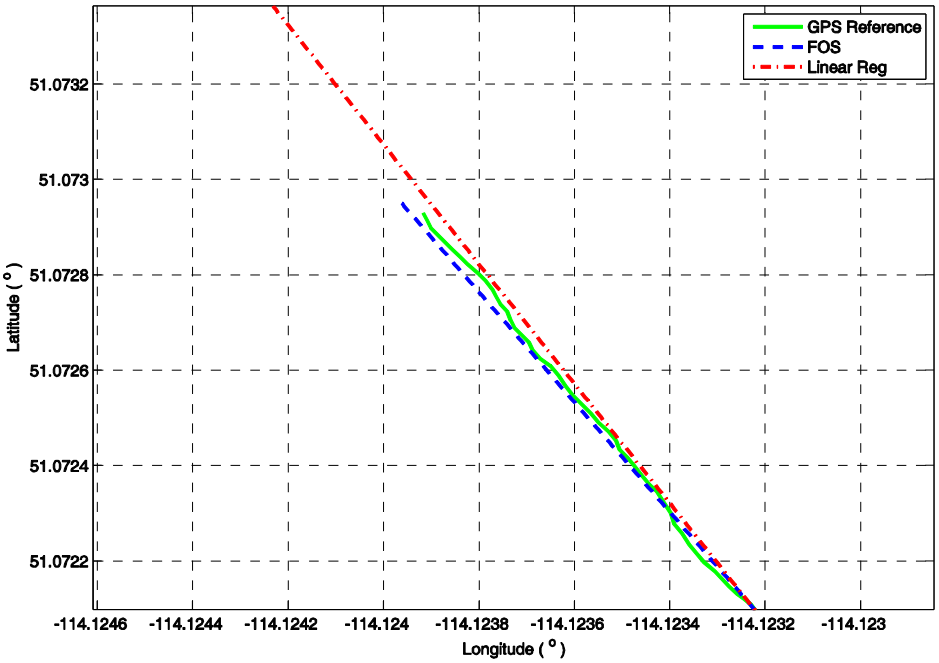


Figure 4-31: Slow running speed trajectory

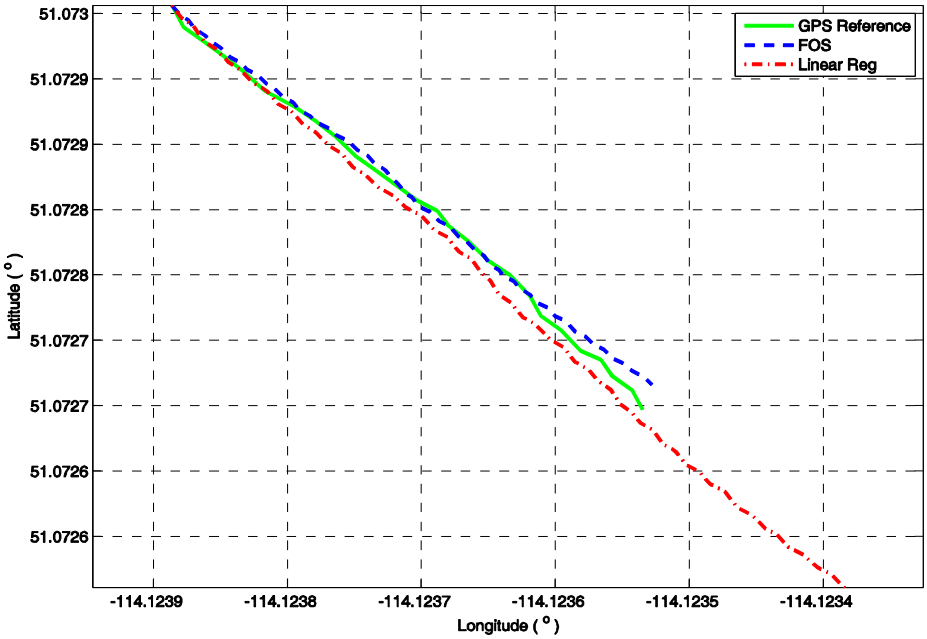


Figure 4-32: Normal running speed trajectory

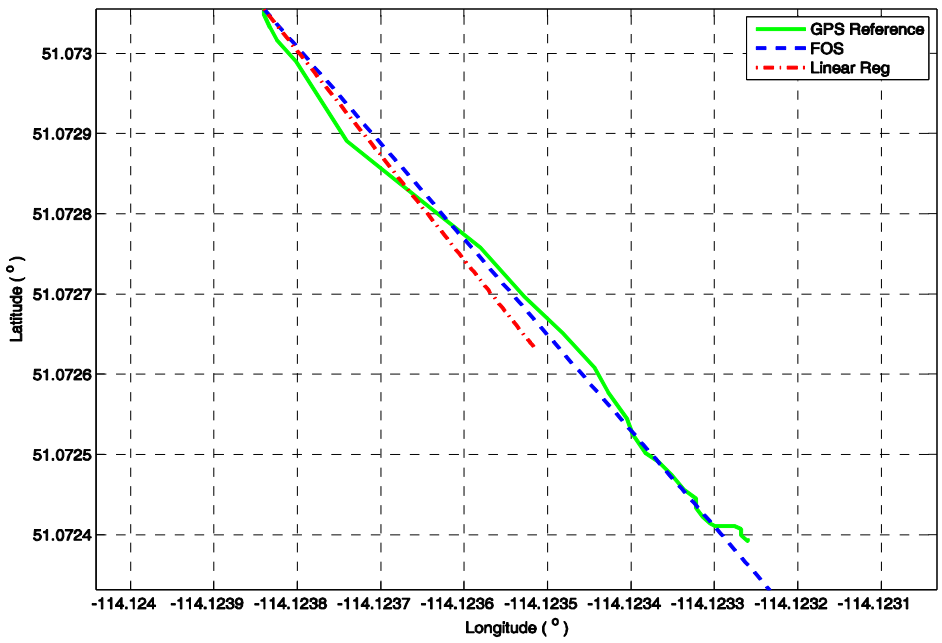


Figure 4-33: Extreme Fast Running speed trajectory

4.6.2 Multiple FOS Multiple Devices Step Length results

In the following set of experiments for varying speed walking using multiple sensors assemblies (in different devices in this case), the data set was collected by 16 different subjects. Different physical characteristics are considered like height, weight, gender, and age. Each subject collected data with three different walking speeds – slow, normal, and fast. Furthermore, each one of the subjects collected two datasets in each speed category and for each device use case (e.g. arm band, pocket, torso, belt, hand dangling, and wrist), in order to have an abundance of datasets for verification.

Three different devices were carried by the user, for the rest of the following figures, the smart glasses are shown in orange, the smart watch in green, and smartphone in in blue. In addition to each result of each single device, the multiple sensor assemblies' solution is shown in red, plus

the reference in black. Three different orientations were tested for both the smartphone and the smart watch: handheld/viewing use case, dangling, and pocket.

Since there were no available device having more than one sensor assembly on it nor the devices were fixed or tethered on the same position on the user's body, i.e. failing to maintain a fixed inter relationships to each other or measuring similar measurements. Therefore, the results shown here are using the output level weighted average architecture which proved to be the most suitable architecture in this application domain for enhancing the step length estimation using portable and wearable devices.

4.6.2.1 Three devices results: Smartphone, Smartwatch, and Smart Glasses

In Figure 4-34, three trajectories for three subjects walking with three different speeds are shown. The slow speed, normal speed, and the fast speed are shown in Figure 4-34-a, Figure 4-34-b, and Figure 4-34-c respectively. Each subject carrying three devices; smart glasses, smart watch in dangling use case, and smartphone in the handheld use case.

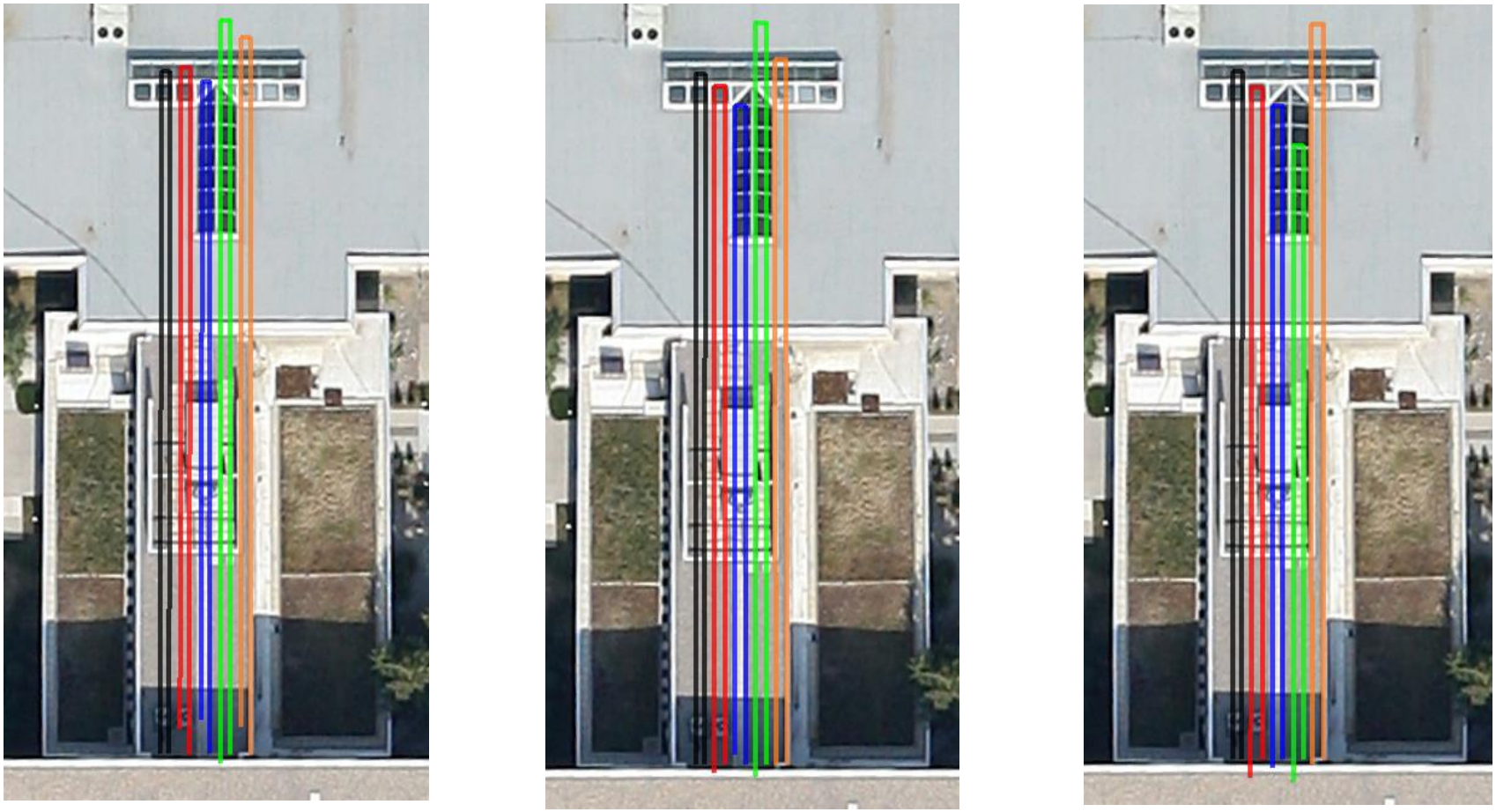
In Figure 4-34-a, the error in the single sensor assembly in the smartphone device alone is 0.26 m in the upward direction (i.e. towards the North of the map) while 1.48 m in the downward direction, and total of 1.74 m in the whole trajectory. The error in the smart watch 3.48 m upward, 4.74 m downward, and total of 8.22 m. Finally, in the smart glasses case, the error were 2.28 m upward, 1 m downward, and total 3.28 m. On the other hand, in the multiple sensor assemblies' case, the error were 0.87 m in the upward direction, while 0.23 m in the downward direction, and total of 1.1 m error in the whole trajectory. The percentage of enhancement of using multiple sensor assemblies over using single ones is 36.78% in case of smartphone, 89.41% in case of smart watch, and 66.46% in the case of smart glasses.

In Figure 4-34-b, the error in the single sensor assembly in the smartphone device alone is 2.32 m in the upward direction (i.e. towards the North of the map) while 2.23 m in the downward direction, and total of 4.55 m in the whole trajectory. The error in the smart watch 3.42 m upward,

5 m downward, and total of 8.42 m. Finally, in the smart glasses case, the error were 0.13 m upward, 1 m downward, and total 3.28 m. On the other hand, in the multiple sensor assemblies' case, the error were 0.87 m in the upward direction, while 0.23 m in the downward direction, and total of 1.1 m error in the whole trajectory. The percentage of enhancement of using multiple sensor assemblies over using single ones is 36.78% in case of smartphone, 89.41% in case of smart watch, and 66.46% in the case of smart glasses.

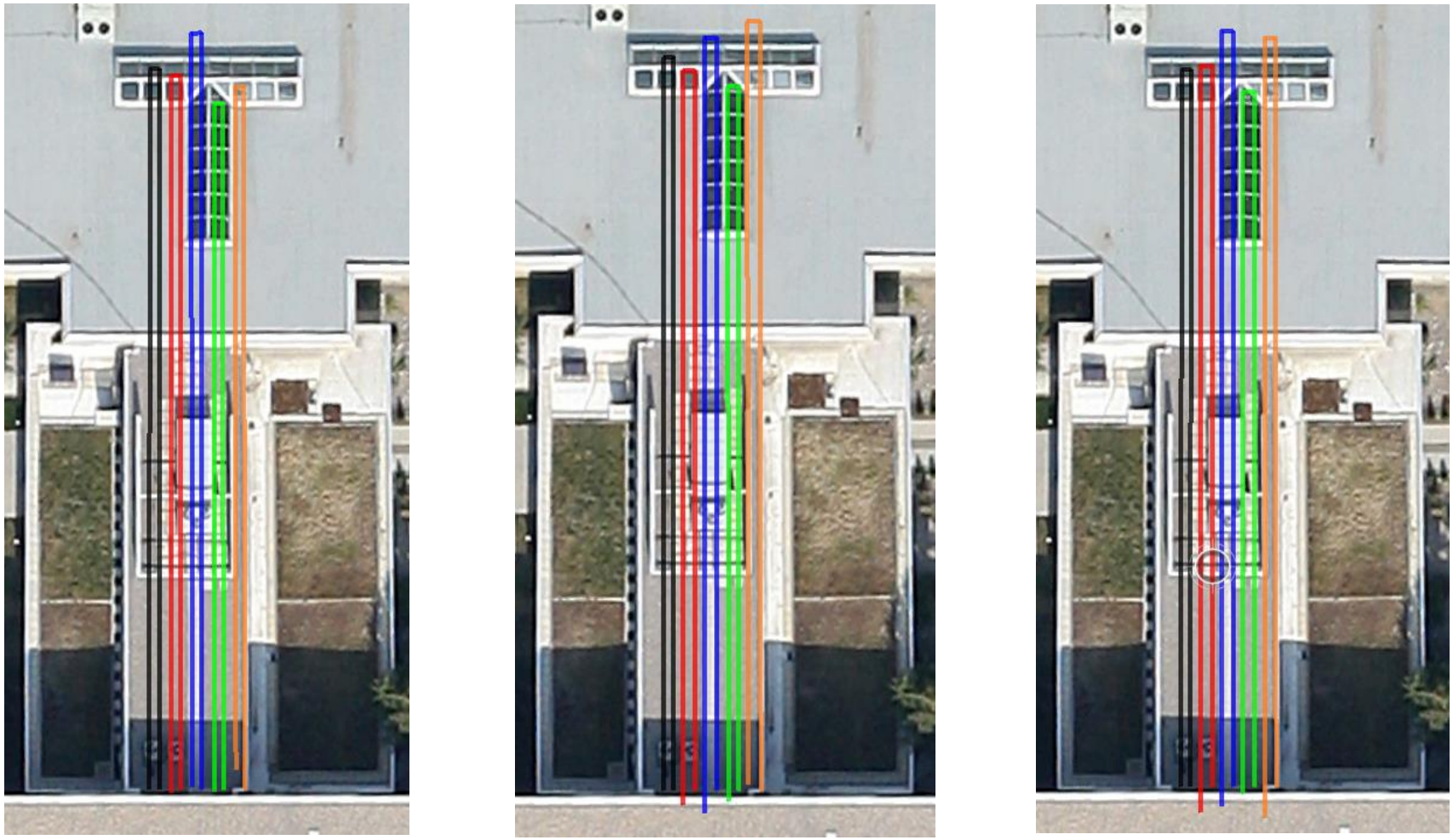
In Figure 4-34-c, the error in the single sensor assembly in the smartphone device alone is 2.32 m in the upward direction (i.e. towards the North of the map) while 2.23 m in the downward direction, and total of 4.55 m in the whole trajectory. The error in the smart watch 3.42 m upward, 5 m downward, and total of 8.42 m. Finally, in the smart glasses case, the error were 0.13 m upward, 1 m downward, and total 3.28 m. On the other hand, in the multiple sensor assemblies' case, the error were 0.87 m in the upward direction, while 0.23 m in the downward direction, and total of 1.1 m error in the whole trajectory. The percentage of enhancement of using multiple sensor assemblies over using single ones is 36.78% in case of smartphone, 89.41% in case of smart watch, and 66.46% in the case of smart glasses.

Instead of repeating stating similar statistics for the following figures, tables 1-4 summarizes the results of the 12 trajectories shown in figures 4-24 – 4-26, where in Figure 4-35, three trajectories are shown for different subjects walking with different speeds – slow, normal, and fast. However, they carry the devices in different use cases than what was shown in Figure 4-34. The use cases were used were; phone in dangling use case, while smart watch in viewing use case. Same case for Figure 4-36 except that both phone and smartwatch were in pocket.



(a) (b) (c)

Figure 4-34: Trajectories 1-3 for three subjects walking with three different speeds, each subject carrying three devices (smart glasses in orange, smart watch in dangling use case in green, and smartphone in the handheld orientation in blue). The multiple sensor solution is shown in red, and the reference in black. (a) Trajectory 1: Slow walking speed (b) Trajectory 2: Normal walking speed (c) and Trajectory 3: Fast walking speed



(a) (b) (c)

Figure 4-35: Trajectories 4-6 for three subjects walking with three different speeds, each subject carrying three devices (smart glasses in orange, smart watch in viewing use case/orientation in green, and smartphone in the dangling use case in blue). The multiple sensor solution is shown in red, and the reference in black. (a) Trajectory 4: Slow walking speed (b) Trajectory 5: Normal walking speed (c) and Trajectory 6: Fast walking speed

(a) (b) (c)

Figure 4-36: Trajectories 7-9 for three subjects walking with three different speeds, each subject carrying three devices (smart glasses in orange, smart watch in pocket in green, and smartphone in pocket in blue). The multiple sensor solution is shown in red, and the reference in black. (a) Trajectory 7: Slow walking speed (b) Trajectory 8: Normal walking speed (c) and Trajectory 9: Fast walking speed

Table 4-1: Phone Results

Use case	Traj Name	1st Half (m)	Error (m)	Percent %	2nd Half (m)	Error (m)	Percent %	Total Error (m)	Percent %	Mean/ Use Case %	Mean/device %
Handheld	Traj 1	46.67	0.5	1.06%	44.19	2.98	6.32%	3.48	3.69%	4.32%	4.79%
	Traj 2	45.09	2.08	4.41%	44.17	3	6.36%	5.08	5.38%		
	Traj 3	45.05	2.12	4.49%	45.63	1.54	3.26%	3.66	3.88%		
Dangling	Traj 4	49.67	2.5	5.30%	49.35	2.18	4.62%	4.68	4.96%	5.31%	
	Traj 5	49.9	2.73	5.79%	50.97	3.8	8.06%	6.53	6.92%		
	Traj 6	48.47	1.3	2.76%	49.68	2.51	5.32%	3.81	4.04%		
Pocket	Traj 7	49.34	2.17	4.60%	48.56	1.39	2.95%	3.56	3.77%	4.73%	
	Traj 8	43.01	4.16	8.82%	44.3	2.87	6.08%	7.03	7.45%		
	Traj 9	46.07	1.1	2.33%	45.46	1.71	3.63%	2.81	2.98%		

Table 4-2: Watch Results

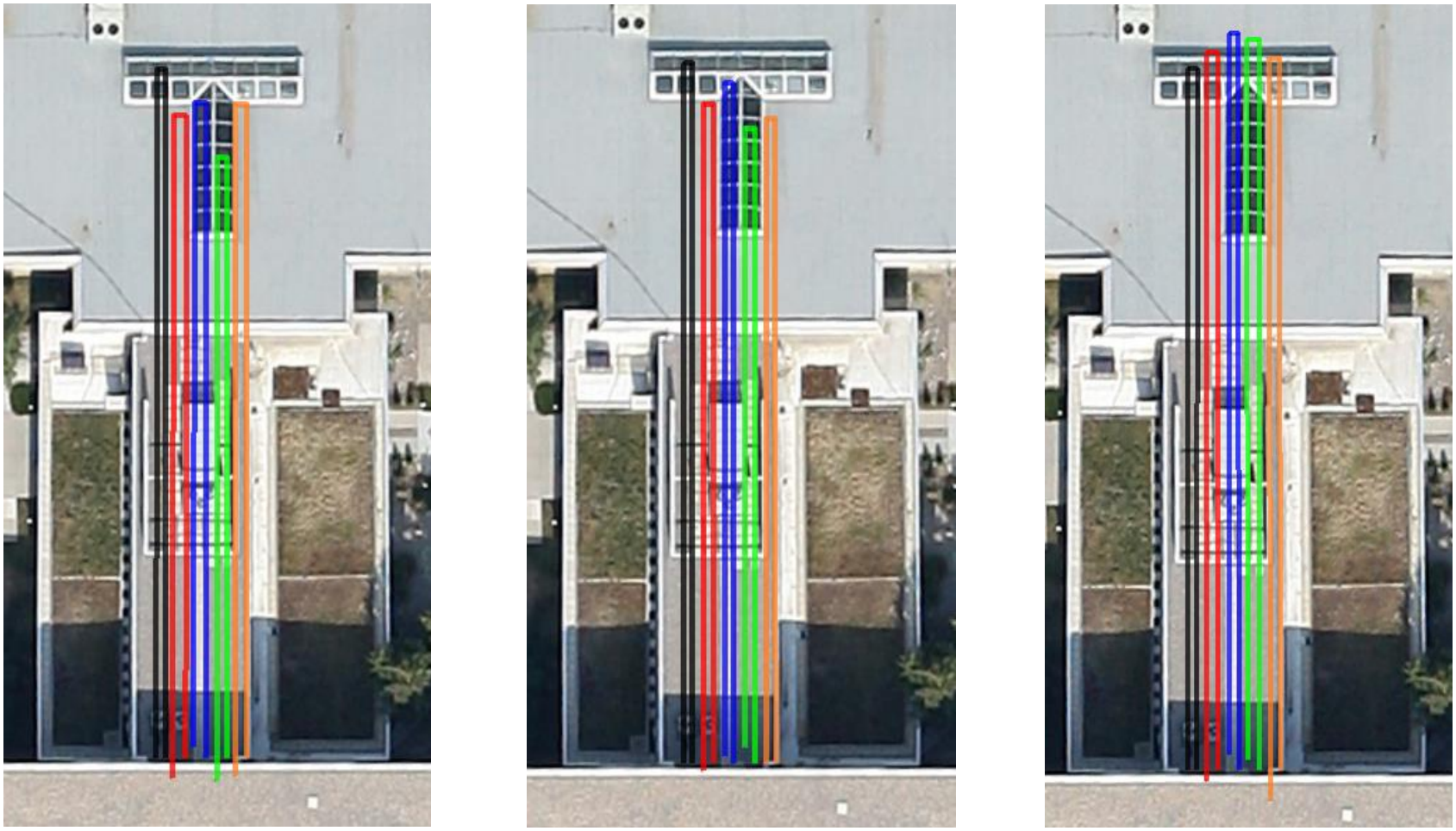
Use case	Traj Name	1st Half (m)	Error (m)	Percent %	2nd Half (m)	Error (m)	Percent %	Total Error (m)	Percent %	Mean/ Use Case %	Mean/device %
Handheld	Traj 1	50.94	3.77	7.99%	51.56	4.39	9.31%	8.16	8.65%	8.80%	6.12%
	Traj 2	50.76	3.59	7.61%	51.82	4.65	9.86%	8.24	8.73%		
	Traj 3	42.07	5.1	10.81%	43.77	3.4	7.21%	8.5	9.01%		
Dangling	Traj 4	50.02	2.85	6.04%	45.07	2.1	4.45%	4.95	5.25%	3.52%	
	Traj 5	45.99	1.18	2.50%	46.28	0.89	1.89%	2.07	2.19%		
	Traj 6	45.43	1.74	3.69%	45.96	1.21	2.57%	2.95	3.13%		
Pocket	Traj 7	44.25	2.92	6.19%	38.91	8.26	17.51%	11.18	11.85%	6.04%	
	Traj 8	46.21	0.96	2.04%	45.22	1.95	4.13%	2.91	3.08%		
	Traj 9	45.66	1.51	3.20%	45.68	1.49	3.16%	3	3.18%		

Table 4-3: Glasses Results

Traj Name	1st Half (m)	Error (m)	Percent %	2nd Half (m)	Error (m)	Percent %	Total Error (m)	Percent %	Mean/device %
Traj 1	49.58	2.41	5.11%	47.77	0.6	1.27%	3.01	3.19%	
Traj 2	48.15	0.98	2.08%	48.42	1.25	2.65%	2.23	2.36%	
Traj 3	50.63	3.46	7.34%	50.81	3.64	7.72%	7.1	7.53%	
Traj 4	46.09	1.08	2.29%	44.77	2.4	5.09%	3.48	3.69%	
Traj 5	49.45	2.28	4.83%	51.25	4.08	8.65%	6.36	6.74%	5.19%
Traj 6	49.43	2.26	4.79%	48.97	1.8	3.82%	4.06	4.30%	
Traj 7	49.24	2.07	4.39%	47.24	0.07	0.15%	2.14	2.27%	
Traj 8	52.18	5.01	10.62%	53.57	6.4	13.57%	11.41	12.09%	
Traj 9	48.85	1.68	3.56%	49.73	2.56	5.43%	4.24	4.49%	

Table 4-4: Multiple Sensors Assemblies' Results

Traj Name	1st Half (m)	Error (m)	Percent %	2nd Half (m)	Error (m)	Percent %	Total Error (m)	Percent %	Mean/Use Case %	Mean/device %
Traj 1	47.54	0.37	0.78%	45.71	1.46	3.10%	1.83	1.94%		
Traj 2	46.34	0.83	1.76%	46.99	0.18	0.38%	1.01	1.07%	1.43%	
Traj 3	46.47	0.7	1.48%	47.69	0.52	1.10%	1.22	1.29%		
Traj 4	46.76	0.41	0.87%	46.81	0.36	0.76%	0.77	0.82%		
Traj 5	47.41	0.24	0.51%	49.14	1.97	4.18%	2.21	2.34%	1.36%	1.30%
Traj 6	46.3	0.87	1.84%	47.18	0.01	0.02%	0.88	0.93%		
Traj 7	48.15	0.98	2.08%	47.68	0.51	1.08%	1.49	1.58%		
Traj 8	46.52	0.65	1.38%	47.55	0.38	0.81%	1.03	1.09%	1.11%	
Traj 9	46.8	0.37	0.78%	47.41	0.24	0.51%	0.61	0.65%		



(a) (b) (c)

Figure 4-37: Trajectories 10-12 for three subjects walking with three different speeds, each subject carrying three devices (a) Trajectory 10: slow walking speed, smart watch is in dangling use case while smartphone is in handheld viewing orientation (b) Trajectory 11: normal walking speed, smart watch is in viewing orientation while smartphone is in dangling use case (c) Trajectory 12: fast walking speed, both smart watch and phone in are pocket

The results shown above are really good, reducing the mean absolute error per device from 4.8% to 1.3% is a good achievement. However, unfortunately this is not the case when same statistics applied to all trajectories in the data set which consist of about 240 trajectories. Figure 4-37 shows some examples when all trajectories are either overestimating or underestimating the step length.

The following table summarizes the enhancement percentage of using the multiple sensor assemblies over each device alone. The whole dataset consist of 240 trajectories, divided into 80 trajectories per each device use case.

Table 4-5: Data set Summary Results

Device	Use Case	Enchantment per Use Case %	Enhancement Per Device %
Phone	Handheld	16.97%	22.81%
	Dangling	43.91%	
	Pocket	7.56%	
Watch	Handheld	3.02%	17.97%
	Dangling	37.77%	
	Pocket	13.13%	
Glasses		9.43%	

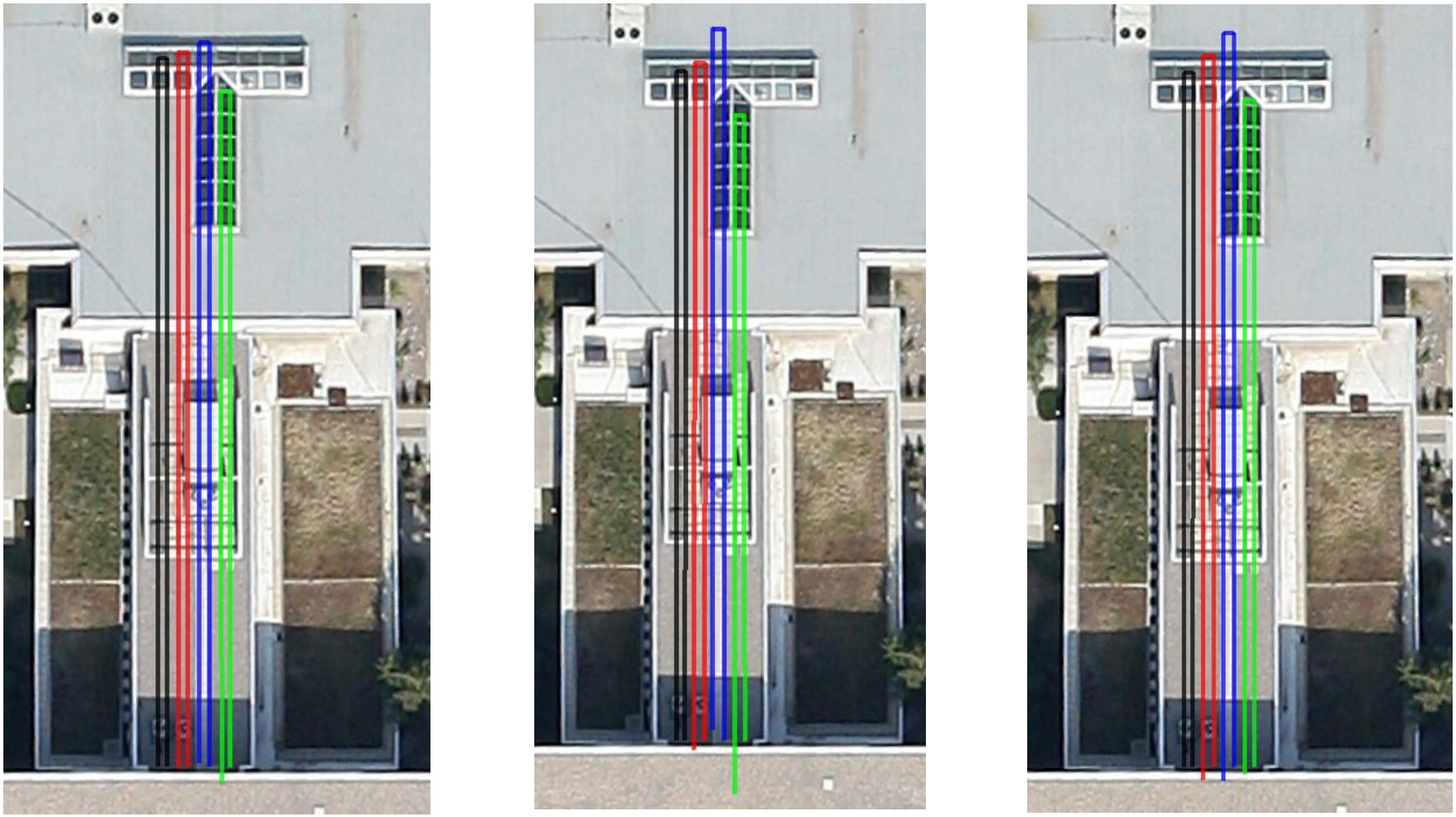
4.6.2.2 Two devices results

Figures 4-28 and 4-29 are showing similar results to that shown in Figures 4-24 – 4-26, the only difference here is that, it is showing the results of two participating devices only not three. Figure 4-38 shows the results of using a phone and smartwatch only, while Figure 4-39 shows the results of using a smart glasses and a smart watch. Of course, the multiple sensor assemblies' solution is not better than all the participating solution all the time as was the case in the previous three-devices examples, where sometimes both participating devices are either underestimating or overestimating the step length of the user. When only the phone and smart watch were used, the multiple sensor assemblies' solution were better than the phone solution 57.87% of the time, and better than the watch solution 49.62 % of the time, and

better than both 18.5% of the time. In addition, when both smart glasses and smart watch were the only participating devices, the multiple sensor assemblies' solution were better than the smart glasses solution 54.72% of the time, and better than the watch solution 63% of the time.

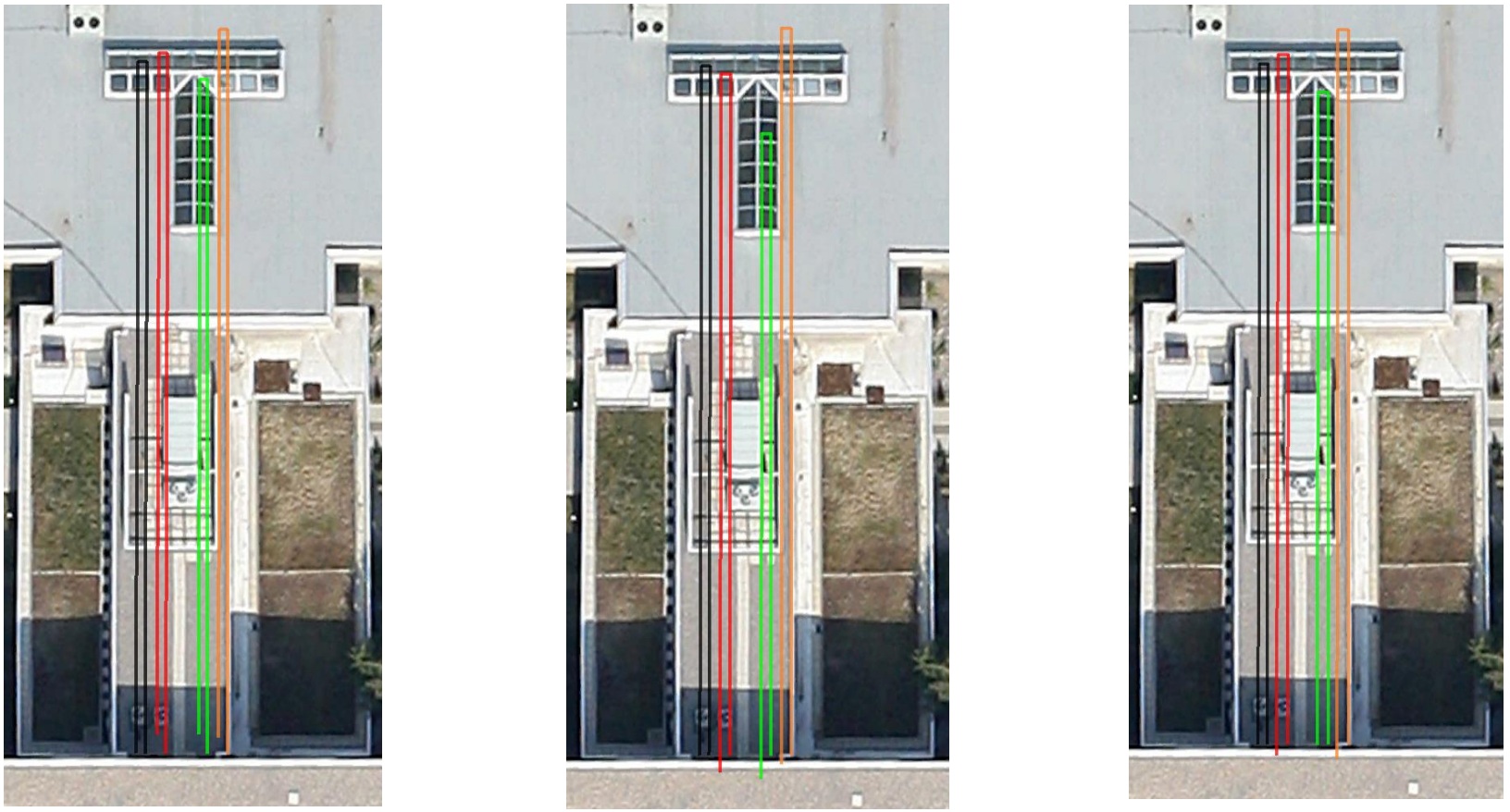
4.7 Conclusion

The presented results demonstrate that the step length estimated from the nonlinear model are more accurate than the one estimated from linear regression model in all the speeds which clearly indicates that the nonlinear model is more capable of solving the varying step length problem for on foot motion without the need of linear approximations either in walking or in running in different speeds.



(a) (b) (c)

Figure 4-38: Trajectories 13-15 for three subjects walking with three different speeds, each subject carrying only two devices (a smart watch in green, and smartphone in blue). The multiple sensor solution is shown in red, and the reference in black. (a) Slow walking speed, smart watch in viewing mode/orientation while smartphone in the dangling use case (b) Normal walking speed, both smart watch and smartphone are in pocket (c) Fast walking speed, smart watch in dangling mode, while smartphone in handheld viewing orientation



(a) (b) (c)

Figure 4-39: Trajectories 16-18 for three subjects walking with three different speeds, each subject carrying only two devices (a smart glasses in orange, and smart watch in pocket in green). The multiple sensor solution is shown in red, and the reference in black. (a) Slow walking speed (b) Normal walking speed (c) Fast walking speed

Chapter 5

Misalignment Angle Estimation

Portable navigation is a challenging problem. Because, in order to obtain an accurate navigation solution, an accurate heading estimation is required. However, portable devices are free to take any orientation. That is why resolving the misalignment angle between the portable navigation device and the moving platform (for example user walking indoors or user driving vehicle outdoors) is critical in order to obtain an accurate heading estimation.

In this chapter, a novel solution for resolving the misalignment angle problem is presented, which exploits multiple portable devices like smartphones or tablets and/or smart wearable devices; such as smart watches, smart glasses, and/or smart fitness and activity trackers.

5.1 Introduction

Smartphones and/or smart wearable devices (ex. smart glasses, smart watch, fitness and activity trackers) can be used in a variety of ways. For example, smartphones can be used in texting mode, on the ear, in a pocket, in a backpack, and on a belt. In order to obtain an accurate heading that represents the moving platform (i.e. the user's heading while walking or the vehicle's heading while driving as shown in Figure 5-1), sensors' sensitivity axes must be aligned with the platform's forward, traversal, and vertical axes. This is what is called the sensors alignment step or resolving the misalignment angle problem.

Sensor alignment is a critical step in order to achieve highly accurate heading estimation, as explained in (Grewal, Weill et al. 2007, Noureldin, Karamat et al. 2013). The problem of misalignment arises when the sensor axes are not exactly aligned with the platform (Savage 1998,

Titterton and Weston 2004). In these cases, the position and heading calculated through dead-reckoning algorithms using the inertial sensors' measurements will not be representative of the platform. Giving reason to why, for high accuracy navigation solutions, fixing and tethering the inertial sensors within the platform and accurately aligning them is a requirement(Groves 2013). In portable navigation, fixing the inertial sensors and accurately aligning them is not an available option (if not impossible). Due to the mobility nature of the devices, we should always expect that there is a misalignment angle between sensors and platform. Therefore, estimating the misalignment angle in portable navigation is essential.

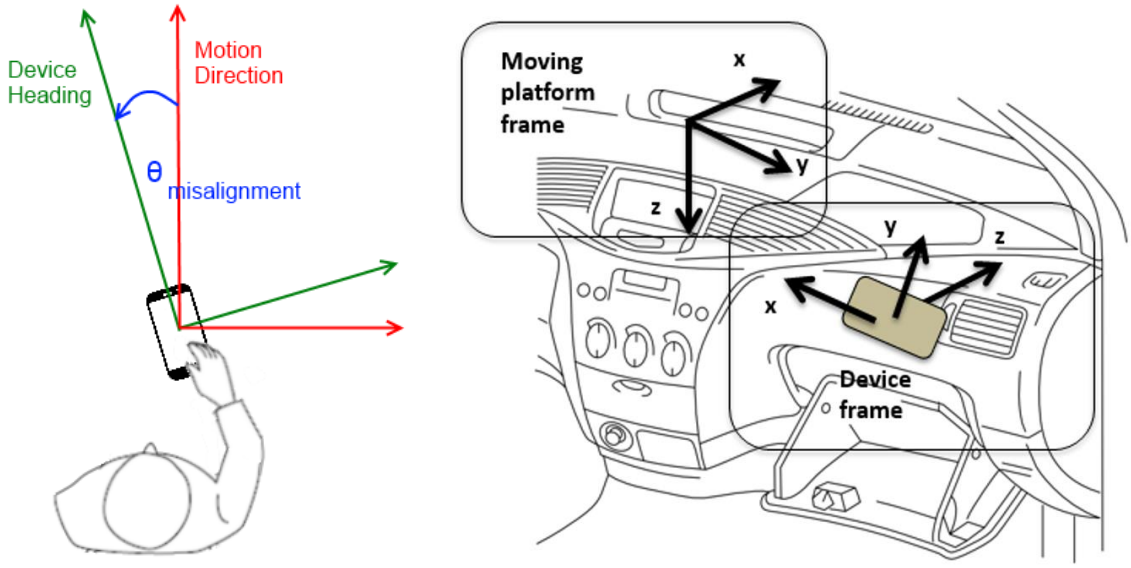


Figure 5-1: Misalignment angle definition in both walking (left) and driving (right)

The misalignment problem has not been tackled before using multiple devices, most solutions presented in previous literature use only a single device. Most commonly, other literature that uses sensors from multiple body worn devices focuses on activity recognition, context detection as in (Gellersen, Schmidt et al. 2002, Kern, Schiele et al. 2003, Maurer, Smailagic et al. 2006), joint kinematics analysis and biomedical research as in (Burns, Greene et al. 2010,

O'Donovan and Ayer 2011), or even full body motion capture as in (Roetenberg, Luinge et al. 2009). However, none calculate or estimate the heading misalignment to be later used to improve navigation and positioning solutions. Moreover, in (Roetenberg, Luinge et al. 2009, Burns, Greene et al. 2010) many sensors are needed and are required to be fixed (or tethered) in a specific position on the user's body, sometimes as specific as a muscle or joint. In the proposed method, the sensors in any device can be carried freely in any position relative to the user's body as in the case of smartphones, tablets, smart glasses or smart watches.

The most important component of misalignment is the heading misalignment angle. For example, in order to apply Pedestrian Dead-Reckoning (PDR), we need the pedestrian heading; not the device heading calculated either via gyroscope, magnetometer, or any other sensor or method. However, knowing the device heading and an estimate of the heading misalignment angle enables us to estimate the pedestrian heading. Another example on the importance of estimating heading misalignment angle in vehicular navigation is applying non-holonomic constraints. Non-holonomic constraints are constraining the vehicular traversal and/or vertical velocities to zero or very small values to aid the navigation solution. The non-holonomic constraint is only valid on a platform (vehicle) reference frame, not the body (device) reference frame. The link between the two frames is again the heading misalignment angle (the vertical component can be obtained from the gravity vector). Since this research focuses on the heading misalignment angle estimation, the heading misalignment angle will be referred to as misalignment angle directly.

This chapter proposes a solution that benefits from a user carrying more than one of the above mentioned devices (i.e. smartphone, tablet, wearable computing devices, or appcessories) while they are wirelessly connected together. A technique to obtain and enhance the misalignment of each portable device by using the information from multiple devices is presented in the following section. The availability of multiple devices with sensors benefits this solution by obtaining or enhancing the misalignment estimate for each device.

5.2 Enhanced Multiple Devices Misalignment

This section demonstrates the calculation of misalignment using multiple devices, which are commonly used electronic gadgets (e.g. smart phones or smart watches). The proposed method may be used with two or more devices. One device is used in estimating the misalignment of another. By knowing the heading and the initial misalignment of at least one device, this information can be used in estimating the free device misalignment relative to the platform heading. It is usually harder to estimate this misalignment using the free device alone.

In this method, the two devices are classified as a pseudo-tethered device and a free device. It is worth noting that this classification is not permanent, meaning that one device can be classified as pseudo-tethered at one instant of time, and be classified as free device at another. Classifying which sensor will play the pseudo-tethered role or the free sensor role is dependent on how fast the device is expected to change its misalignment angle relative to the platform. A pseudo-tethered device for now could be a belt clip or chest-mounted fitness (or activity tracker) appcessory worn by the user. The other free device could be a smart phone, smart glasses, and/or smartwatch.

The proposed method states that the misalignment angle of the free device is equal to the heading difference of the free device and the pseudo-tethered device plus the misalignment angle of the pseudo-tethered device. This can be interpreted in terms of a mathematical equation as follows:

$$M_{FD} = H_{FD} - H_{PTD} + M_{PTD} \qquad (5\text{-}1)$$

Where:

$M_{FD}: Misalignment\ of\ Free\ Device$
$H_{FD}: Heading\ of\ Free\ Device$
$M_{PTD}: Misalignment\ of\ Pseudo-Tethered\ Device$
$H_{PTD}: Heading\ of$ Pseudo $-\ Tethered\ Device$

Both the free device heading H_{FD} and the pseudo-tethered device heading H_{PTD} can be obtained from the KF state vector as shown in Chapter 2, which can integrate magnetometer measurements as well when available.

The misalignment of the pseudo-tethered device M_{PTD} can be obtained from a single device misalignment method as shown in (Syed, Georgy et al. 2012, Ali, Chang et al. 2013, Syed, Georgy et al. 2013), or an averaged value of such misalignment over the period where the device was pseudo-tethered till the current time.

Figure 5-2 is a flow chart which illustrates the abstract mathematical equation (5-1). When the initial misalignment between at least two devices, the pseudo-tethered device, and another one, (which it is desired to calculate its misalignment) is known. Any changes in the misalignment difference between the two devices is tracked. If there is a change in the misalignment difference, and no change in the pseudo-tethered device heading, then this change is due to the change in the free device heading. Otherwise, if the heading of both devices is changing, then this change is due to platform heading change. Tracking is continued until a misalignment difference occurs. In Figure 5-3, there is an example on each case described in the flow chart in Figure 5-2.

In Figure 5-3, there is an example on each case described in the flow chart in Figure 5-2, and mapping each decision branch in the flow chart to the corresponding example case in the shown figure. Each decision branch in the flow chart can be interpreted as the rule has been followed in order to calculate the misalignment angle of the free unit at that point of time.

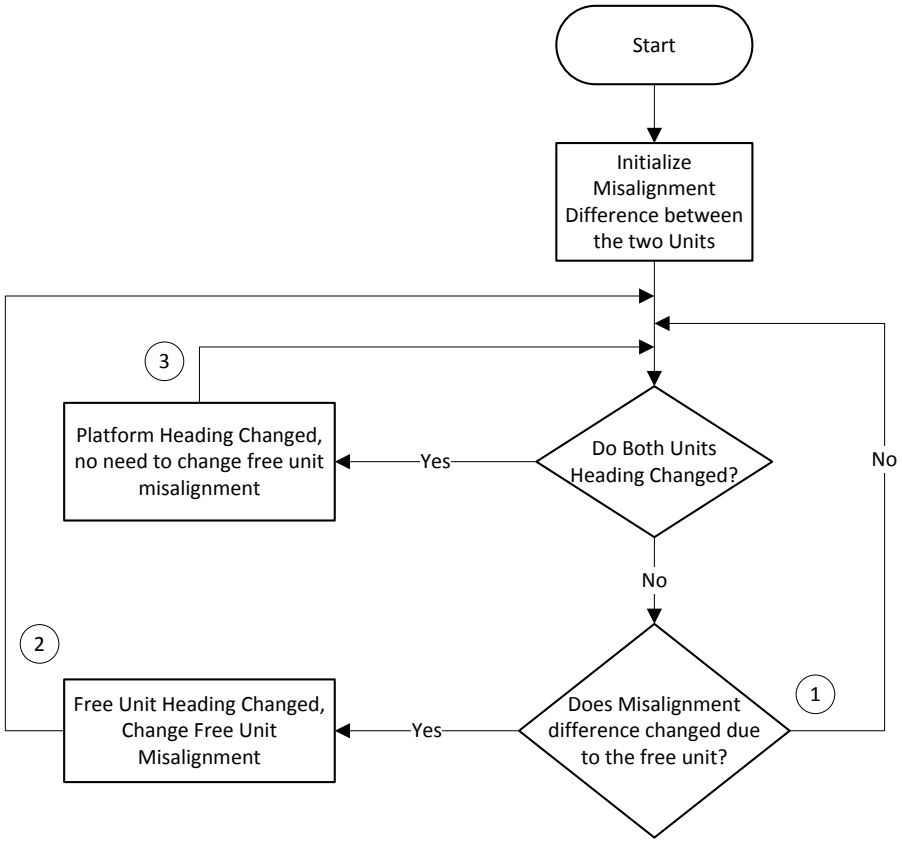


Figure 5-2: The proposed misalignment method flow chart

For example, during the first period, since neither one of the two units is changing heading nor the misalignment difference between the two units is changing, therefore, there is no change in the misalignment value of the free unit, and the misalignment value remains unchanged from the value that has been set up in the first step (the initialization step). In the second period, not both units are changing heading, the misalignment difference value is changing, this means that the free unit is the one which is responsible for this misalignment difference change, and consequently, its misalignment angle value should be changed by the same amount the misalignment difference between the two units changed. Finally, in the third period, both unit are changing heading by the same amount, which means that the platform is the one which is changing heading this time, therefore, there is no need to change the free unit misalignment value in this case.

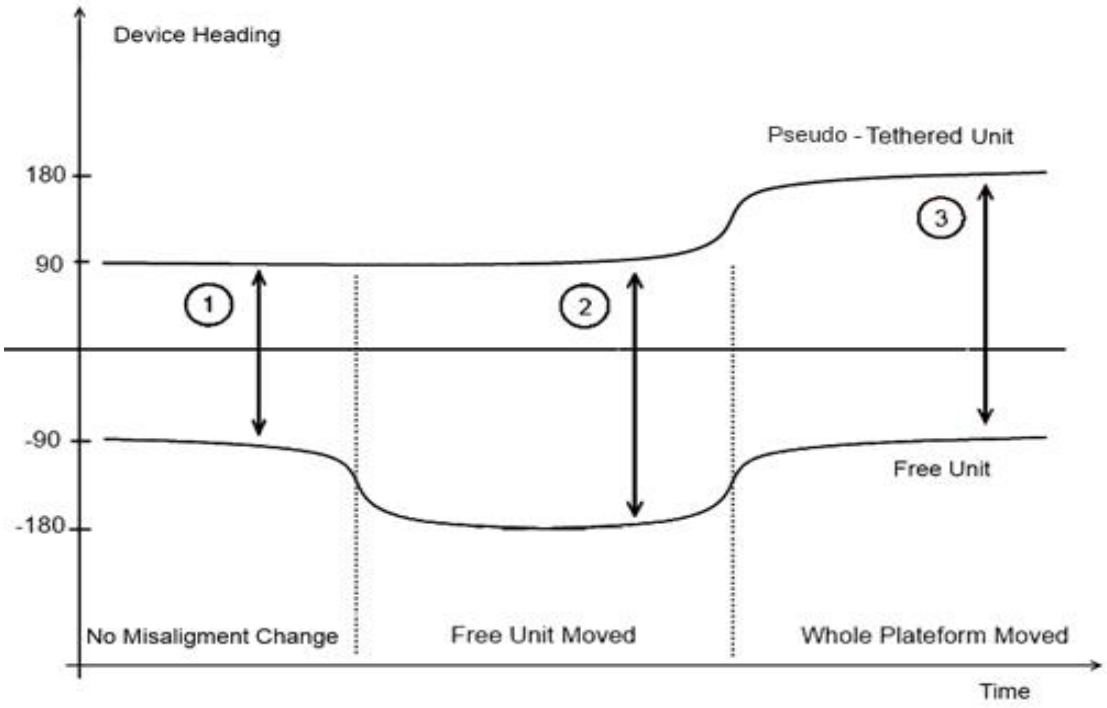


Figure 5-3: Example on flow chart different cases

The second and third cases can happen simultaneously. In other words, both the platform and the free device can both change their heading at the same time. However, since we are changing the misalignment according to the difference between the pseudo-tethered device and the free device, the platform heading change will be canceled out, and the final free device misalignment change will be due to the change in the free device heading alone. Keeping the misalignment change (or update) reflecting only the change in the free device heading, and totally isolated from change in platform heading, can be considered as one of the advantages of this method.

In addition, one more distinguishing feature of the proposed solution is that it exploits the sensors already contained in many of today's electronic devices to come up with accurate navigation solution by enhancing misalignment angle estimation. The integration of information through a connected network of multiple portable devices or appcessories provides highly accurate misalignment angle estimation. This development is further used to enhance the final navigation

solution. The final navigation solution, relying on the enhanced misalignment estimation for each device, is more accurate, reliable, and robust in all environments (including indoors and urban canyons, whether driving or on foot) thanks to the multiple sensor triads found in nearly each and every portable device found today.

Finally, the most important advantage for using a misalignment angle estimation algorithm is enabling the motion of any of the involved portable or wearable, device(s) to move within the platform (person or vehicle) without any constraints. The orientation of each device, with respect to the moving platform, can change freely to any orientation, and still be able to estimate the platform's heading accurately.

5.2.1 Misalignment Angle Estimation Applications

The misalignment of the free device M_{FD} can be used later on in the calculation of the rotation matrix C_b^v, which transforms quantities from the body frame to the platform frame. One important application which depends on C_b^v is the non-holonomic update. The non-holonomic update constrains the navigation solution from moving in the traversal and/or vertical directions by constraining the platform traversal and/or vertical velocities to zero or very small values to aid the navigation solution.

According to the non-holonomic constraint definition, two measurement updates to the navigation KF can be considered

$$\mathrm{v}_y^{\mathrm{v}} = 0 \qquad (5\text{-}2)$$

$$\mathrm{v}_z^{\mathrm{v}} = 0 \qquad (5\text{-}3)$$

The computed velocity in the v-frame can be expressed as:

$$\mathbf{v}^{\mathrm{v}} = C_b^v \mathbf{v}^b \qquad (5\text{-}4)$$

where $\mathbf{v}^b$ is the platform velocity with respect to the b-frame and can be further expressed as

$$\mathbf{v}^{\mathrm{v}} = C_b^v C_l^b \mathbf{v}^l \quad (5\text{-}5)$$

Another important application is PDR. The latitude φ_k and longitude λ_k are updated in PDR algorithm as follow

$$\varphi_k = \varphi_{k-1} + \frac{S_k \cos\theta_k}{R_m + h_k} \quad (5\text{-}6)$$

$$\lambda_k = \lambda_{k-1} + \frac{S_k \sin\theta_k}{(R_n + h_k)\cos\varphi_k} \quad (5\text{-}7)$$

Where φ_{k-1} and λ_{k-1} are the previous latitude and longitude respectively, R_m and R_n are the Meridian and prime vertical radii of curvature respectively, h_k is the current altitude, S_k is the current step length which can be estimated using one of the proposed methods in the previous Chapter. Finally, θ_k is the platform heading (the user's heading not the device heading) which can be obtained by subtracting the misalignment angle $M_{FD,k}$ from the device heading $H_{FD,k}$ as follow

$$\theta_k = H_{FD,k} - M_{FD,k} \quad (5\text{-}8)$$

5.3 Results

A large amount of data was collected for testing and verification. Several datasets were collected for walking scenarios using two or more devices freely carried by the user or attached at different positions on the user's body. The positions used for collecting data included: (i) attached to the body, such as on the waist to emulate a smart belt clip, on the wrist to emulate a smart watch, on the head to emulate smart glasses, and on the chest to emulate fitness and health monitoring appcessories; as well as (ii) freely carried or in use by the user, such as handheld, in pocket, and on the ear, which emulates different use cases for smartphones and also the transitions between them in-run. Several driving datasets were also collected using two or more devices setup at different positions in a vehicle, such as on a seat or in a smartphone cradle. The numerous datasets collected intends to cover as many real life scenarios as possible to demonstrate how the proposed method can enhance misalignment estimation, heading accuracy, and therefore, the overall navigation

solution. The results clearly demonstrated increased reliability and that more accurate misalignment estimation can be obtained by integrating multiple devices carried by gadget users.

A large number of datasets were collected for testing and verification of the proposed method. The datasets were collected using two or more devices. The tests were done on a free device to emulate a smart phone, a device strapped to the wrist to emulate a smartwatch, a device in a belt clip to emulate an electronic belt clip, and a device strapped to a glass-like head mount to emulate smart glasses.

In the following subsection, driving results from two different trajectories are presented. Next, similar walking results are shown. Finally, the results of two more walking trajectories which show the effect of accurate misalignment estimation on final position solution are illustrated.

5.3.1 Driving Results

In this subsection, results from two different driving trajectories are presented. In both trajectories the user was sitting beside the driver. In the first trajectory, three devices were used, a belt clip, smartwatch, and smart glasses. And the user was asked to act as normal as he can, i.e. looking naturally to the right and left, and looking at the watch from time to time. While in the second trajectory, besides the belt clip, a more focus was concentrated on smart phone in different use case, such as: on seat, on ear, in handheld portrait, in a cup holder, and in pocket. The sensitivity axes of the inertial sensors included in the smartphone and the smartwatch are shown in Figure 5-4. While the sensitivity axes of the electronic belt clip and smart glasses are similar to that of the smartphone.

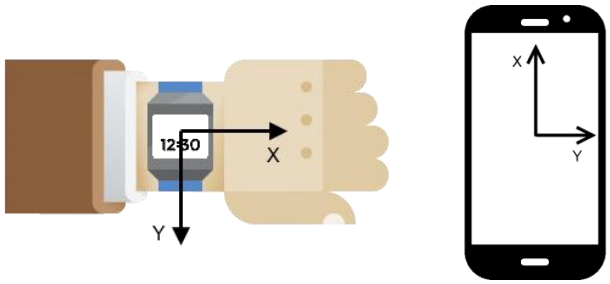


Figure 5-4: Emulated smartwatch (left) and smartphone (right) axes definition

Ideally, in the first trajectory, the misalignment of both, the smartwatch and the smart glasses should be around zero. However, the user cannot fix his hand or his head so that the misalignment be exactly zero. In fact, this is one of the main objectives of the proposed method which is not forcing any constraints on how the user carry or use any device. Therefore, we should expect the results to be around the zero (as in the ideal case) except when the user is looking at his smartwatch to know the time. In this case and according to the sensitivity axes of the smartwatch shown in Figure 5-4, we should expect the misalignment angle to be around 90˚. Similarly, in the case of smart glasses, we should expect zero misalignment most of the time except when the user looks towards the right or the left.

Figure **5-5** shows the results of applying the proposed method using the belt clip as the pseudo-tethered unit to calculate the misalignment of the smartwatch. The figure shows some spikes in the misalignment signal and it happened more frequently in the beginning of the trajectory than in the end. This spikes represents the moments when the user was looking at his smartwatch. At that instant the misalignment jumps near 90˚ and returns back quickly since the user does not last long, that is why it appears like a spike.

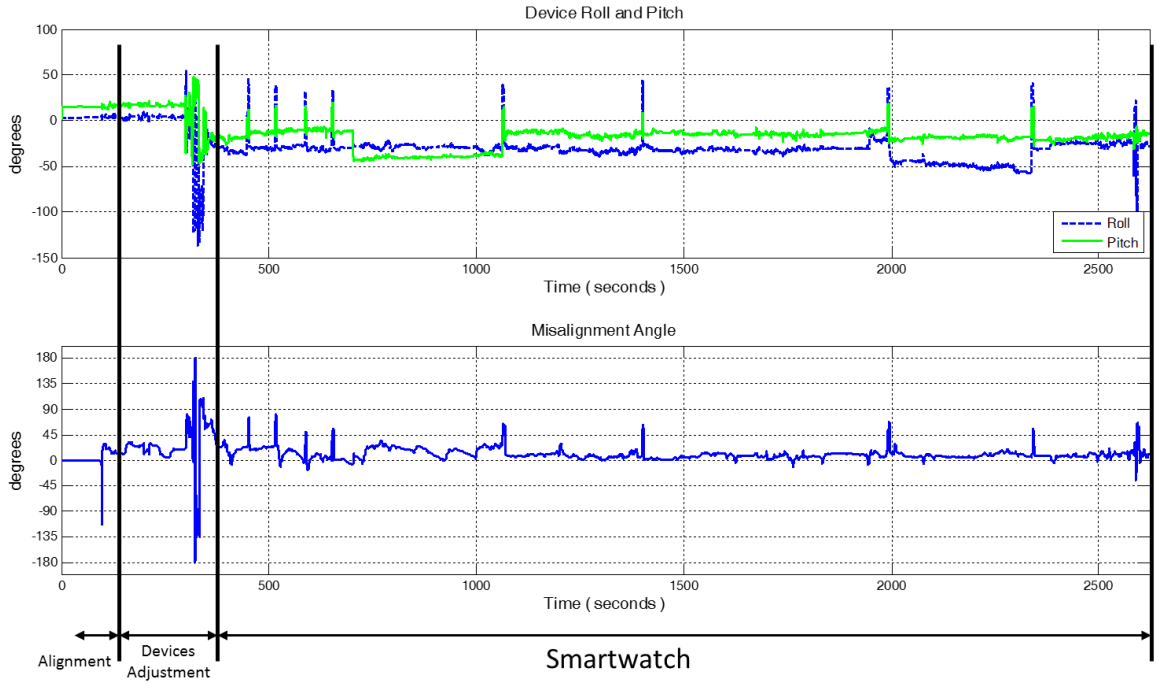


Figure 5-5: Vehicle Trajectory – Using belt clip as a reference to calculate smartwatch misalignment

Figure 5-6 shows the results of applying the proposed method using the belt clip as the pseudo-tethered unit to calculate the misalignment of the smart glasses in this case. Despite that the user was looking towards the right and the left freely and randomly as shown in the figure. The misalignment estimation reflects these movements and returns back around zero without any problems.

Figure 5-7 shows similar results to the previous ones but for the second vehicle trajectory instead. Where the proposed method was applied using the belt clip as the pseudo-tethered unit to calculate the misalignment of a free smartphone while being in different positions relative to the user, such as: seat, ear, handheld portrait, cup holder, and pocket as shown in Table 5-1.

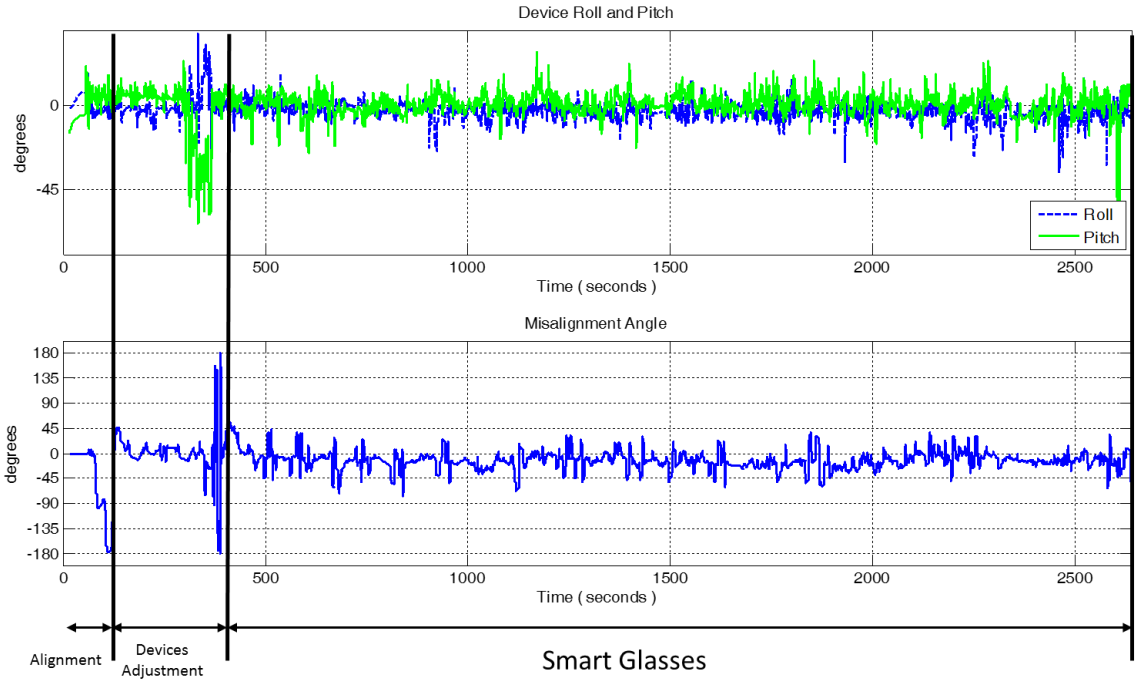


Figure 5-6: Vehicle Trajectory – Using belt clip as a reference to calculate free smart glasses misalignment

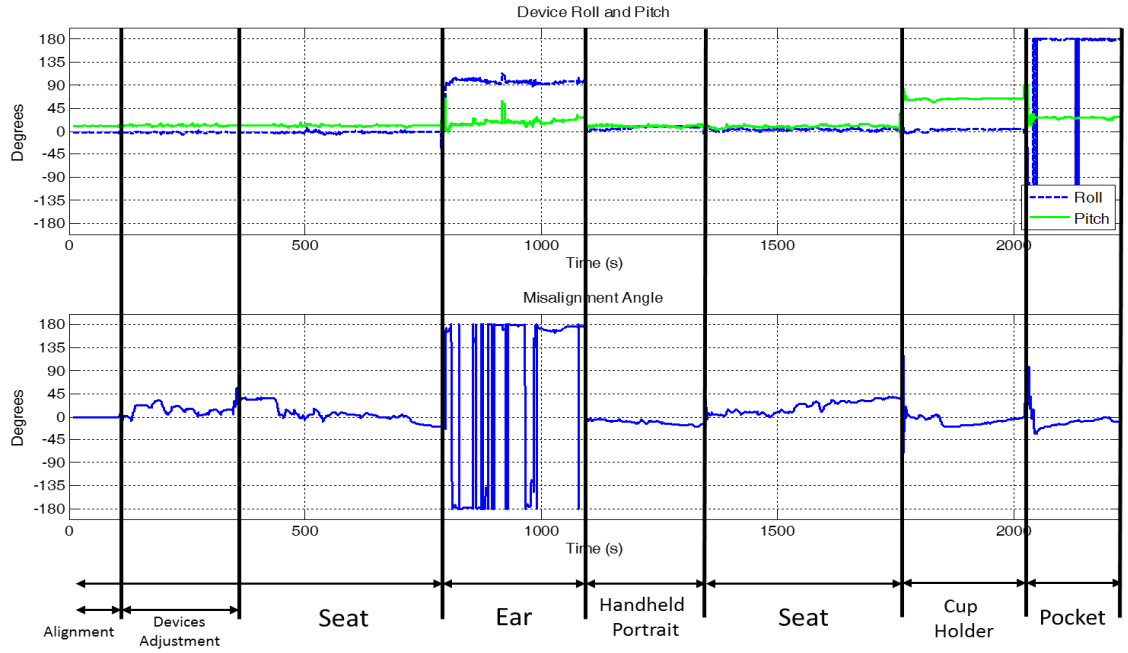


Figure 5-7: Vehicle Trajectory – Using belt clip as a reference to calculate free smartphone misalignment

Table 5-1: Use case description for each segment in the vehicle trajectory

SEGMENT #	USE CASE
1	SEAT
2	EAR
3	HANDHELD PORTRAIT
4	SEAT
5	CUP HOLDER
6	POCKET

Ideally, the misalignment of the seat, handheld portrait, cup holder, and pocket, should be around zero, and only ear misalignment should be around ±180˚. This is in agreement with the results shown in the figure, where the misalignment angle estimation remains around the zero most of the time, even when the user change the use case of the smartphone, except in the on ear use case, the misalignment estimation is around ±180˚.

5.3.2 Walking Results

In this subsection, results from two different walking trajectories are presented. Three devices were used, a belt clip, smartphone, and a smartwatch. In both trajectories the user held the smartphone by the same arm having the smartwatch, i.e. changing the smartphone use case affects both the smartphone and smartwatch as well. The sensitivity axes of the inertial sensors included in the smartphone and the smartwatch are shown in Figure 5-8. While the sensitivity axes of the electronic belt clip is similar to that of the smartphone.

Ideally, the misalignment of the smartphone in handheld portrait and dangling use cases should be around zero, because the smartphone will be pointing forward in the same direction of motion of the user. However, in handheld landscape mode, it should be around ±90˚ (depending on the device direction towards the left or the right of the user), because the smartphone will be pointing towards the transverse direction of the user. Finally, in the case of ear use case, the

misalignment should be around ±180˚, because the smartphone in this case will be pointing towards the reverse direction of the user.

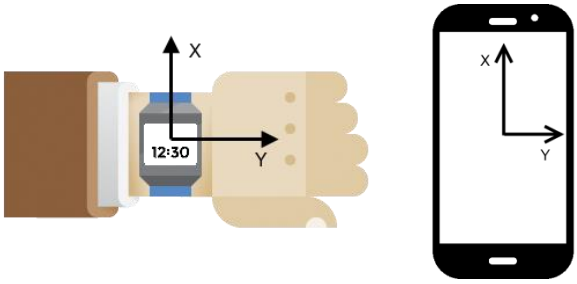


Figure 5-8: Emulated smartwatch (left) and smartphone (right) axes definition

Nevertheless, the user cannot fix his hand so that the misalignment be exactly zero, ±90˚, or 180˚ as in the ideal scenarios discussed above. In fact, this is one of the main objectives of the proposed method which is not forcing any constraints on how the user carry or use any device. That is why it is really difficult to compare the proposed method result to some reference. Instead, following are some results to show how the proposed method result is compared to the ideal values of different use cases. While at the end of this section, two more results are shown to illustrate how misalignment angle estimation can affect the final navigation solution.

Following are the results for the first example walking trajectory where four devices were used: glasses, belt clip, watch, and a free moving smartphone. In this trajectory, the same hand having the smartwatch held the smartphone. The trajectory consist of seven loops, at the beginning of each loop the user changed his use case for the free device (smartphone in this case). Table 5-2 shows the use case used at each loop. The glasses is used as the pseudo-tethered unit in this example and which have been used in calculating the misalignment of the free unit (smartphone in this case).

Table 5-2: Use case description for each loop in the walking trajectory 1

Loop #	Use Case
1	Handheld Portrait
2	Pocket Vertical
3	Handheld Landscape (phone speaker towards left)
4	Dangling
5	Ear
6	Handheld Landscape (phone speaker towards right)
7	Pocket Horizontal

As shown In Figure 5-9, at first loop, the misalignment is around zero because the free unit was in the handheld portrait use case. Following, at the second and fourth loops of the trajectory, from the point of view of misalignment calculation, both the pocket vertical and dangling use cases respectively are similar to the handheld portrait use case, where in both cases the free unit is pointing forward in the same direction of motion of the user, and the misalignment angle should be expected to be around zero, and this what is shown in the result, the resulted misalignment is indeed around the zero. Next, at the third loop of the trajectory, the misalignment is around -90˚ because the free unit was in the handheld landscape use case (pointing towards the left of the user). At the fifth loop, the misalignment is around -180˚ because the free unit was in the ear use case pointing backward of the user's direction of motion. The sixth loop is similar to the third one, except that the device is pointing towards the right of the user instead of the left, and that is why the misalignment angle is around +90˚ this time. Finally, in the seventh loop, the device is in the pocket again but horizontally pointing towards the left of the user, and that is why the misalignment angle is -90˚.

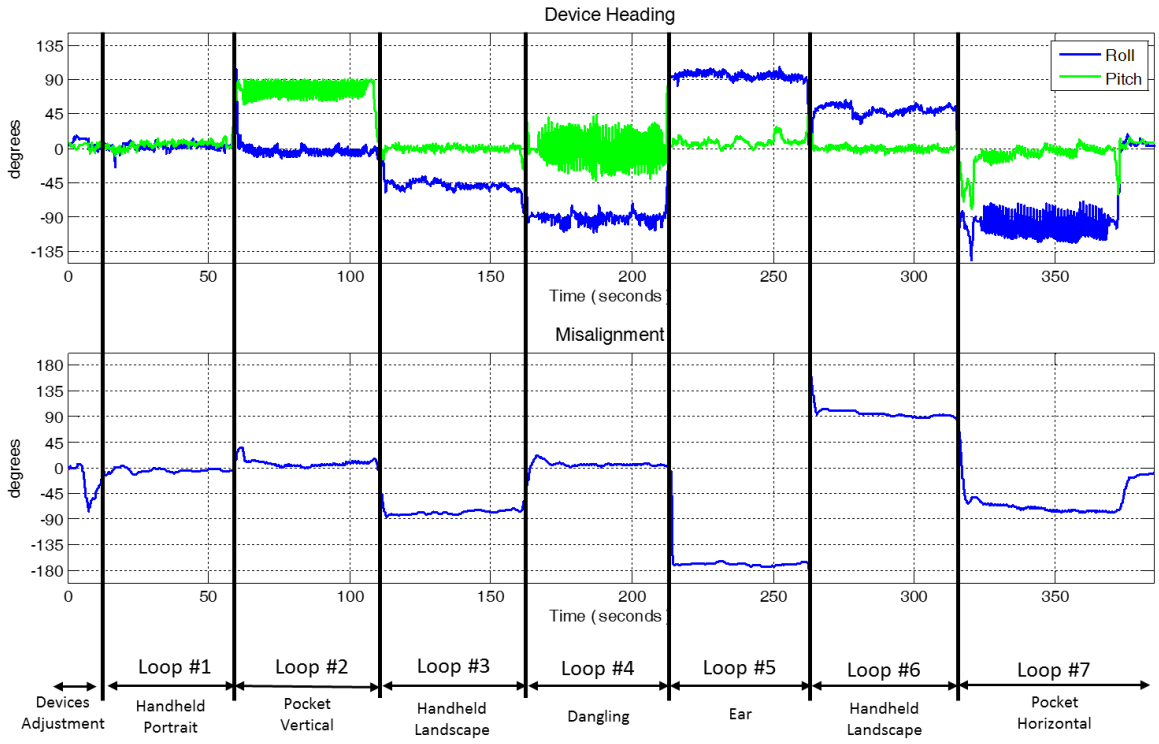


Figure 5-9: Walking Trajectory 1 – Using glasses as a reference to calculate free smartphone misalignment

Figure 5-10 is showing similar results to Figure 5-9. However, in Figure 5-10, the free device is the smartwatch and was following different use cases shown in **Table** 5-3.

Ideally, the misalignment of the smartwatch in the view mode should be around zero. On the other hand, when the smartphone is in handheld portrait use case, the smartwatch misalignment should be around ±135˚ (noting that the user will not hold the device at sharp 90˚ with respect to his wrist for his convenience, that is why ±135˚ instead of ±180˚), because while the smartphone is pointing forward, the watch would be pointing backward. In ear use cases, the smartwatch misalignment should be around zero, because the smartwatch will be pointing forward in the same direction of motion of the user, the misalignment of the ear use case here is around zero and not around ±180˚ (as mentioned in the previous section) because of the change in the axes definition

of the watch in each example (as illustrated in Figure 5-4 and Figure 5-8). Finally, in dangling use cases, it should be around ±180˚, because the smartwatch will be pointing backwards.

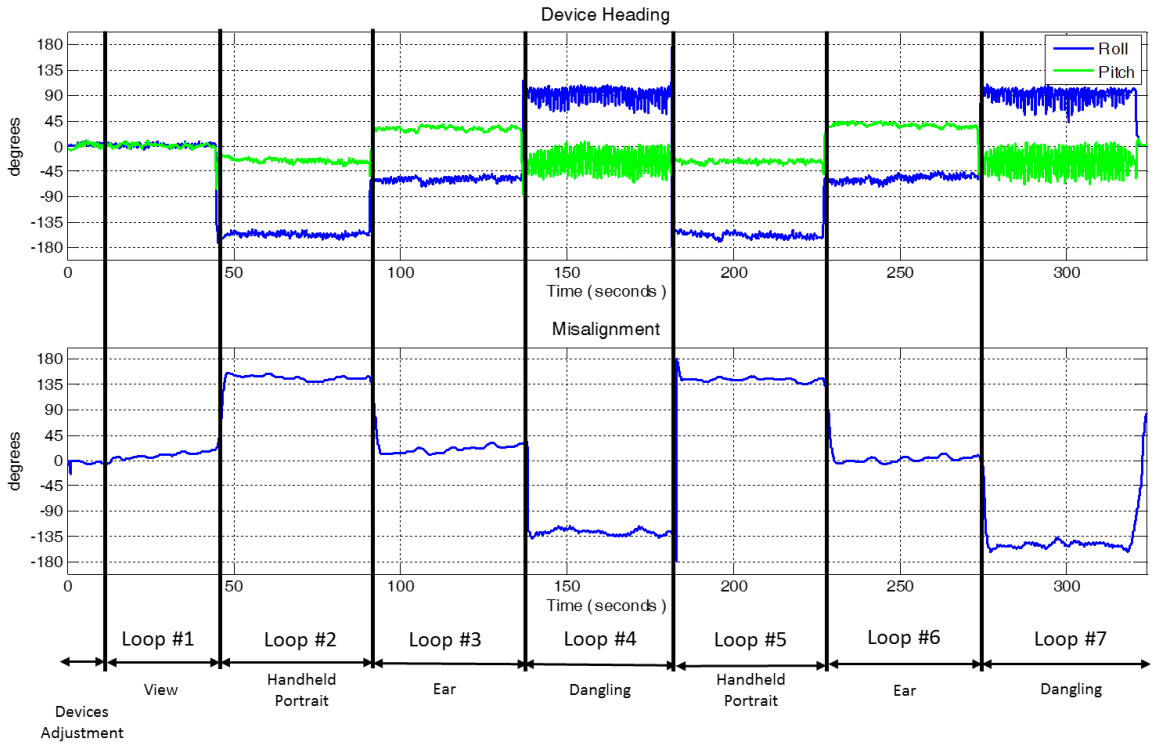


Figure 5-10: Walking Trajectory 2 – Using glasses as a reference to calculate free smartwatch misalignment

Table 5-3: Use case description for each loop in the walking trajectory 2

LOOP #	USE CASE
1	VIEW
2	HANDHELD PORTRAIT
3	EAR
4	DANGLING
5	HANDHELD PORTRAIT
6	EAR
7	DANGLING

5.3.3 Effect of Misalignment Estimation on Final Position Solution

In the following two examples, the proposed method is compared to another method from literature which requires only one device to estimate the misalignment angle. It is worth to mention here that this method from literature enjoys a great performance and the following two results are very rare to happen. However, the following two results can lead us to the conclusion that integrating both methods can result in a more robust solution than each method alone, at least when there are more than one device is available.

In Figure 5-11, the misalignment angle estimation of a smartphone in handheld landscape (texting) use case is shown for both method, the single IMU method (the method from literature) in blue vs the multiple IMU method (the proposed method). Knowing that the user did not change the use case within the whole trajectory. A nearly constant value misalignment angle estimate shall be expected in this case. It is clear in the result of the proposed method that the misalignment angle is nearly constant. On the other hand, the result of the single IMU method is far away from being constant. The result of the comparison is clearer as shown in Figure 5-12 and Figure 5-13 when we can see the effect of the error of the misalignment angle reflected on the final navigation solution of the user.

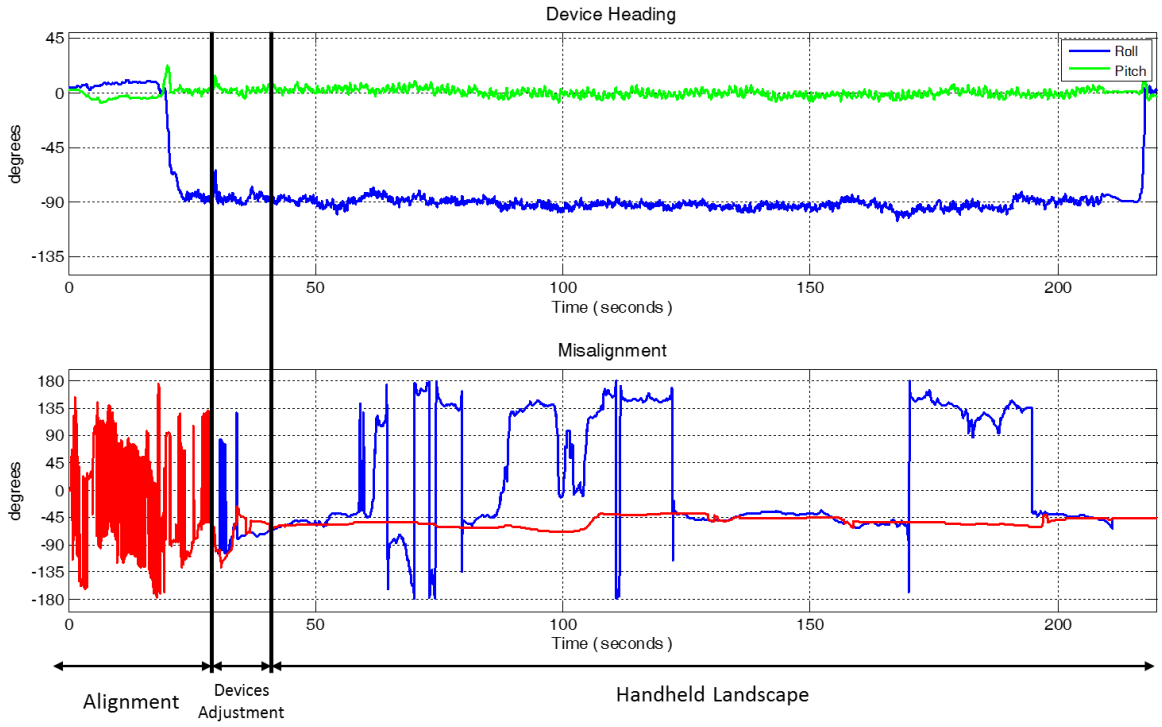


Figure 5-11: Single IMU vs. Multiple IMU Misalignment

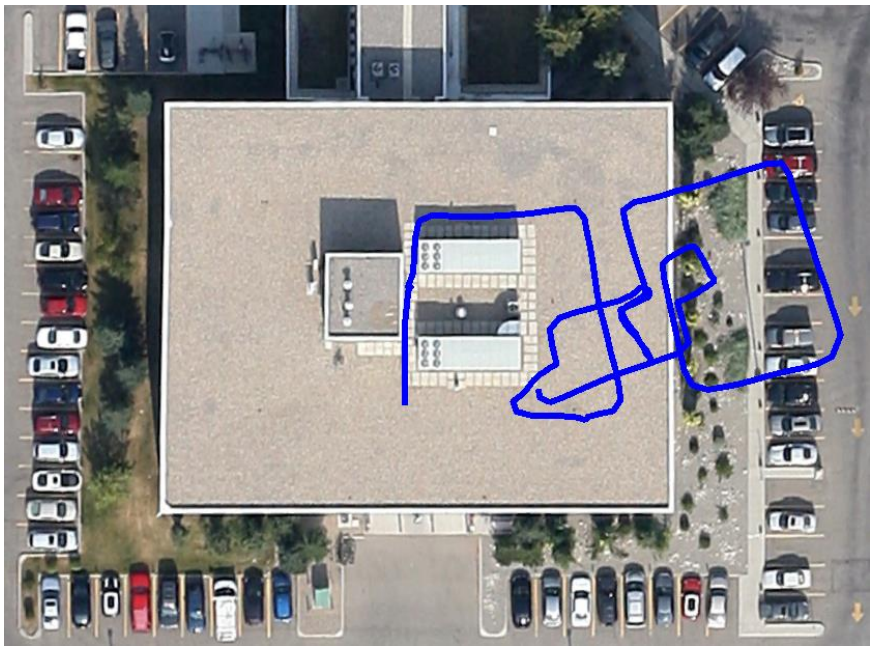


Figure 5-12: Single IMU Result

Figure 5-13: Multiple IMU Result

A similar result is shown In Figure 5-14, while in this trajectory, the free device is a smartwatch in dangling use case, and an ideal misalignment angle estimate should be around ±180˚. This is clearly achieved in the multiple IMU case, but not in the single IMU case. Where, in the single IMU case, the misalignment angle estimate is flipping between ±180˚ which represents no problem since both angle are equivalent, but it flips to some values around the zero as well as indicated by the arrows shown in the figure, which in fact represent an 180˚ error in the misalignment angle. In other words, the user motion can be interpreted as walking in the reverse direction in the final navigation solution. In fact, this is what is shown in Figure 5-15 and Figure 5-16. Instead of the user is moving in loops in clock wise direction as it is the case in the multiple IMU solution in Figure 5-16. In Figure 5-15, the user is shown to be moving in counter clock wise direction loops.

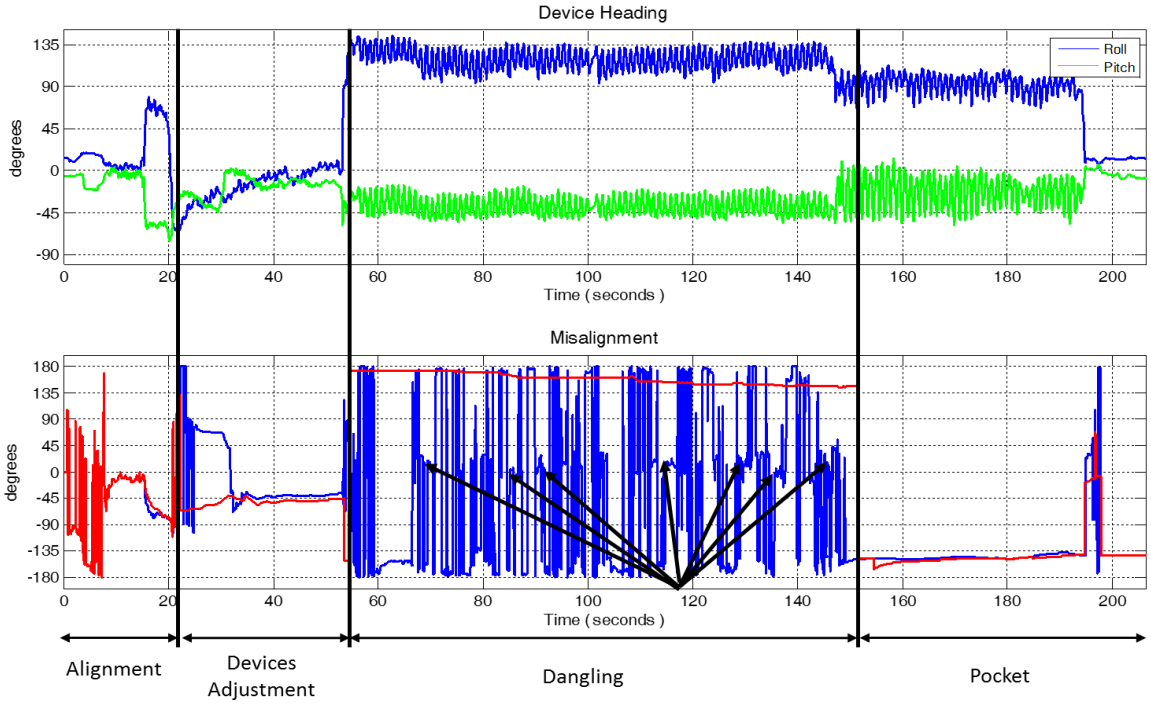


Figure 5-14: Single IMU vs. Multiple IMU Misalignment

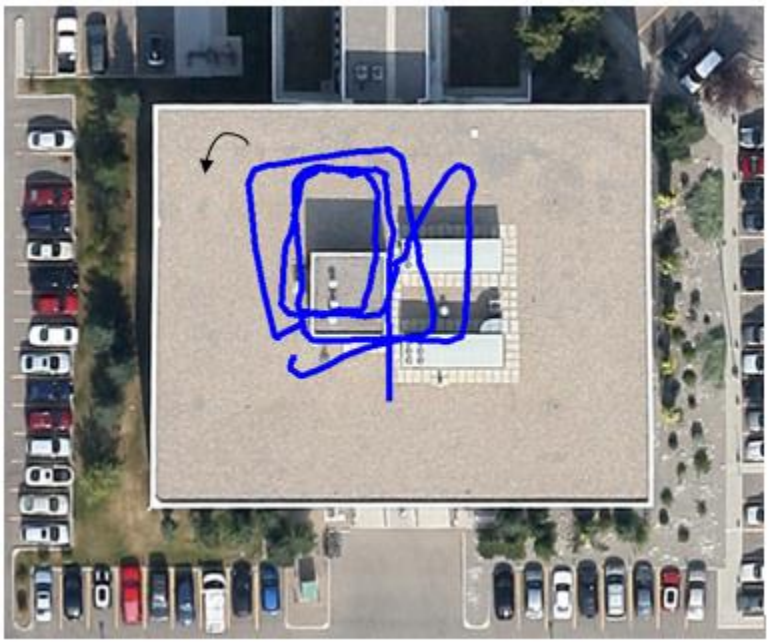


Figure 5-15: Single IMU Result

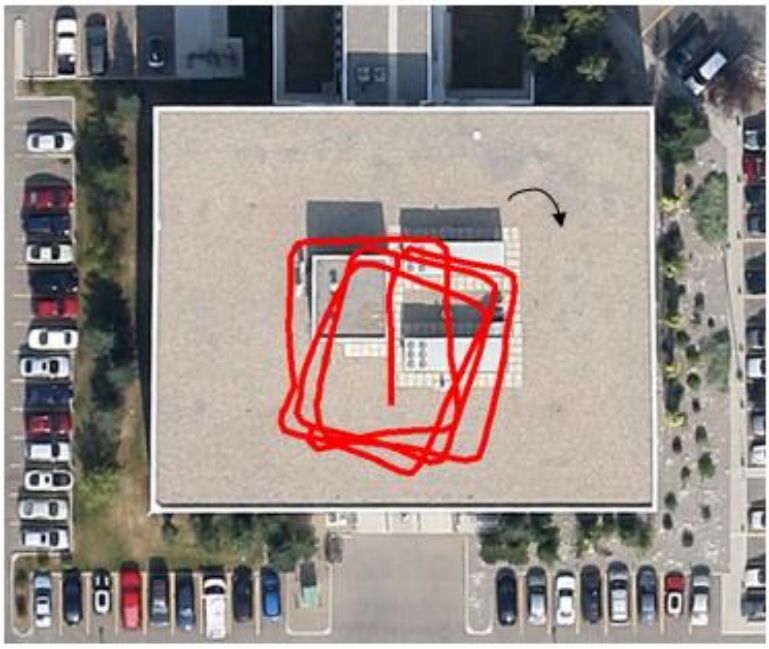


Figure 5-16: Multiple IMU Result

After analyzing the results, it is clearly demonstrated that a genuinely reliable and more accurate navigation misalignment estimation can be obtained by integrating multiple devices carried by current gadgets' users. All results in Figures 5-5 – 5-16 show how the proposed method succeeds in calculating the misalignment of the free device despite the challenge of changing its position relative to the user.

5.4 Conclusion

To conclude, portable navigation of smartphones and/or smart wearable devices is gaining worldwide attention, due to the vast number of possible applications that portable navigation can enable. However, portable navigation is a challenging problem due to the mobility nature of the devices. In order to obtain an accurate navigation solution an accurate heading estimation is required. One way to achieve highly accurate heading estimation is through accurate sensor

alignment. Another way to achieve highly accurate heading estimation and which is more appropriate to our portable navigation problem nature, is through misalignment angle estimation.

In this chapter, a new misalignment angle estimation method was presented. This new method exploits the multiple devices that can be carried by the same user. Results showed how effectively using multiple portable and wearable devices can solve the misalignment estimation problem. For example, a smart belt clip or smart glasses can help in estimating the misalignment of a smartphone carried by the same user. Consequently, a highly accurate user heading can be obtained, which leads to a more accurate portable navigation solution. One of the main advantages of the proposed method is its simplicity, which make it fit best in its targeted portable applications domain.

Chapter 6

Position and Velocity Aiding

At first, smartphones were the most common smart device, then tablets became widely used, and now, smart glasses and smartwatches have started to increase in popularity. In addition, an increasing number of different health monitoring and fitness application accessories (or appcessories for short) are becoming available in the market.

This chapter proposes a solution which exploits the fact that more than one of the following devices: smartphone, tablet, wearable computing device, or appcessory, will be carried by the same user and wirelessly connected together. In this chapter, a new type of navigation filter update and accompanying multiple sensors fusion architecture is presented. The proposed solution benefit from the sensors available in multiple devices carried by the same user to get a better overall navigation solution. In general, the proposed solution can help any filter-based navigation solution whether it is INS or PDR. However, as mentioned in Chapter 4, integrating acceleration and heading rate signals leads to fast drift. Therefore, PDR is preferred in walking and running scenarios, while INS is used in driving scenarios.

6.1 Introduction

Using multiple or redundant sensors is not new in the field of navigation (Skog, Nilsson et al. 2014, Skog, Nilsson et al. 2014). Redundant sensor triads were used for safety and reliability reasons in the beginning. For example, in applications which require navigation systems with online failure detection and fail-safe operation (Krogmann 1990, Titterton and Weston 2004).

Taking into consideration that the redundancy of the sensors generates additional observations, which can be used to estimate the sensor errors without requiring any external aid (Waegli, Guerrier et al. 2008, Martin and Groves 2013). In 2011, Yuksel and El-Sheimi (Yuksel

and El-Sheimy 2011) suggested a method to optimize the inertial sensor fusion problem, which was derived for a special type of redundant IMU, called skew redundant inertial measurement units (SRIMU).

Jin et al. (Jin 2011, Jin, Toh et al. 2011) presented a pedestrian tracking scheme using two sets of low cost DR sensors (MEMS IMUs). This pedestrian tracking scheme exploits the stable relative displacements between two sensor modules carried by the same pedestrian. The positioning problem was defined as a maximum a posteriori sensor fusion problem and a derivation for the optimal solution for this problem was proposed. The optimal solution proposed was shown to be the weighted average of the two sensor modules' solution. For the rest of this document, this architecture will be referred to as the weighted average method.

The proposed solution by Jin et al. (Jin 2011, Jin, Toh et al. 2011) was experimentally evaluated by using (i) two mobile devices, each containing a single orientation sensor, mounted with arbitrary device orientations, and (ii) one mobile device, containing two different orientation sensors, mounted with fixed device orientation. The proposed scheme had exhibited good tracking performance with good error reductions, compared to traditional Dead-Reckoning (DR), in both scenarios. The largest error reduction rate was reported at 73.7%. However, in some other runs, the proposed algorithm delivered intermediate tracking performance, with average errors in between those of the two phones' individual DR systems.

Bancroft (Bancroft 2010) also presented three methods for fusing multiple IMUs at two different layers or domains. The first layer was defined to be in the observation domain. The second layer was higher than the first one and was defined to be in the estimation domain. Two methods were presented under the first layer, and one method under the second layer. The first method is simply mapping all raw IMU measurements into a common frame (i.e. a virtual frame) and processed in a typical combined GNSS-IMU Kalman filter. The second method is similar to the first one except a large stacked filter was constructed of several IMUs. This filter construction

allowed for relative information between the IMUs to be used as updates. Finally, the third method was a federated filter used to process each IMU as a local filter. The output of each local filter is shared with a master filter, which in turn, shares information back with the local filters.

Bancroft's results (Bancroft 2010) indicated that the stacked filter provides a linear increase in accuracy, whereas other architectures typically have less improvement with the addition of more than three IMUs. Areas where GPS is sufficient show little improvement with additional IMUs. Only the stacked filter decreases the minimal detectable blunder of GPS observations by a significant amount.

Bancroft again introduced Relative Position Updates (RPUPT) in another work (Bancroft 2009). Assuming the IMUs are rigidly mounted, and the vector between them (in the body frame) remains constant, it was possible to use this information as an update to the system. This update does not provide an absolute position correction (as a GPS update does), but provides a method of constraining the rate of divergence between two IMUs. Relying on the same principle idea discussed before in the RPUPT, Relative Velocity Updates (RVUPT) and Relative Attitude Updates (RAUPT) were introduced in the same work (Bancroft 2009). It was shown that relative attitude updates provide pessimistic position variances, whereas the relative position and velocity updates provide more realistic position variances. The combination of position, velocity, and attitude updates provides the most significant improvement, where in seven 30-second GPS-outages, the position improvement using five inertial units was 34 %, whereas in urban canyons there was an improvement of 55 % on the RMS error during 400 s of data**.**

6.2 Single Device Navigation Solution

Equation (6-1) shows the mechanization equation as explained in (Titterton and Weston 2004, Noureldin, Karamat et al. 2013).

$$\begin{bmatrix} \dot{\boldsymbol{r}}^l \\ \dot{\mathbf{v}}^l \\ \dot{\boldsymbol{C}}_b^l \end{bmatrix} = \begin{bmatrix} D^{-1}\mathbf{v}^l \\ \boldsymbol{C}_b^l \mathbf{f}^b - \left(2\boldsymbol{\Omega}_{ie}^l - \boldsymbol{\Omega}_{el}^l\right)\mathbf{v}^l + \mathbf{g}^l \\ \boldsymbol{C}_b^l(\boldsymbol{\Omega}_{ib}^b - \boldsymbol{C}_l^b(\boldsymbol{\Omega}_{ie}^l + \boldsymbol{\Omega}_{el}^l)\boldsymbol{C}_b^l) \end{bmatrix} \tag{6-1}$$

The attitude of the moving body is determined by solving the time derivative equation of the transformation matrix C_b^l (rotation matrix between body frame (b-frame) and Local Level Frame (LLF or l-frame)).

$$\dot{\boldsymbol{C}}_b^l = \boldsymbol{C}_b^l[\boldsymbol{\Omega}_{ib}^b - \boldsymbol{C}_l^b(\boldsymbol{\Omega}_{ie}^l + \boldsymbol{\Omega}_{el}^l)\boldsymbol{C}_b^l] \tag{6-2}$$

where $\boldsymbol{\Omega}_{ib}^b$ is the skew symmetric matrix of the gyroscopes' readings $\boldsymbol{\omega}_{ib}^b = \left[\omega_x, \omega_y, \omega_z\right]^T$

$$\boldsymbol{\Omega}_{ib}^b = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix} \tag{6-3}$$

and $\boldsymbol{\Omega}_{ie}^l$ is the skew symmetric matrix of the rotation of the Earth with respect to the inertial frame expressed in LLF $\boldsymbol{\omega}_{ie}^l = [\omega_e \cos\varphi, 0, -\omega_e \sin\varphi]^T$

$$\boldsymbol{\Omega}_{ie}^l = \begin{bmatrix} 0 & -\omega_e \sin\varphi & 0 \\ \omega_e \sin\varphi & 0 & -\omega_e \cos\varphi \\ 0 & \omega_e \cos\varphi & 0 \end{bmatrix} \tag{6-4}$$

and $\boldsymbol{\Omega}_{el}^l$ is representing the skew symmetric matrix of the turn rate of the LLF with respect to Earth-Centered Earth Fixed Frame (ECEF) expressed in LLF $\boldsymbol{\omega}_{el}^l = \left[\frac{v_e}{R_N+h}, -\frac{v_n}{R_M+h}, -\frac{v_e \tan\varphi}{R_N+h}\right]^T$

$$\boldsymbol{\Omega}_{el}^l = \begin{bmatrix} 0 & \frac{v_e \tan\varphi}{R_N+h} & -\frac{v_n}{R_M+h} \\ -\frac{v_e \tan\varphi}{R_N+h} & 0 & -\frac{v_e}{R_N+h} \\ \frac{v_n}{R_M+h} & \frac{v_e}{R_N+h} & 0 \end{bmatrix} \tag{6-5}$$

The time rate of change of the position components has the following relation to the velocity components:

$$\dot{\boldsymbol{r}}^l = D^{-1}\mathbf{v}^l \tag{6-6}$$

$$\begin{bmatrix} \dot{\varphi} \\ \dot{\lambda} \\ \dot{h} \end{bmatrix} = \begin{bmatrix} \frac{1}{R_M+h} & 0 & 0 \\ 0 & \frac{1}{(R_N+h)\cos\varphi} & 0 \\ 0 & 0 & -1 \end{bmatrix} \begin{bmatrix} v_n \\ v_e \\ v_d \end{bmatrix}^l \tag{6-7}$$

where in the above equation, D^{-1} transforms the velocity from LLF to the rate of change of Earth-Centered Earth Fixed (ECEF) coordinates whereas R_M and R_N are the meridian and normal radii of the Earth's reference ellipsoid respectively.

Finally, the time rate of the velocity vector $\mathbf{v}^l = [v_n, v_e, v_d]^T$ can be represented as:

$$\dot{\mathbf{v}}^l = \boldsymbol{C}_b^l \mathbf{f}^{\boldsymbol{b}} - \left(2\boldsymbol{\Omega}_{ie}^l - \boldsymbol{\Omega}_{el}^l\right)\mathbf{v}^{\boldsymbol{l}} + \mathbf{g}^{\boldsymbol{l}} \tag{6-8}$$

where $\boldsymbol{C}_b^l$ is the transformation matrix from the body frame to the LLF, and which requires knowing the platform heading. Whereas, $\mathbf{f}^{\boldsymbol{b}}$ is the specific force measured by the accelerometers in body frame, $\boldsymbol{\Omega}_{ie}^l$ is the skew-symmetric matrix of angular velocity for the Earth rotation expressed in LLF and which can be calculated as shown in equation (6-4), while $\boldsymbol{\Omega}_{el}^l$ is the skew-symmetric matrix of the angular velocity vector of LLF with respect to ECEF frame expressed in LLF and which can be calculated as shown in equation (6-5). Finally, $\mathbf{g}^{\boldsymbol{l}}$ is the gravity vector in LLF.

In order to protect the navigation state vector from drifting, a 15-states Kalman filter (KF) is used. In addition to the nine states of position, velocity, and attitude, there are six more states maintained by the filter. Three states are for the gyros' biases, and another three for the accelerometers' biases. In equation (6-9), $\mathrm{x_k}$ represents the 15-state vector at time k, Φ_k is the system model as explained in (Titterton and Weston 2004, Noureldin, Karamat et al. 2013), while w_k is a zero-mean white Gaussian noise with covariance matrix Q_k, which represents the process noise.

$$\mathrm{x_k} = \Phi_k x_{k-1} + w_{k-1}, \qquad w_k \sim N(0, Q_k) \tag{6-9}$$

The KF state vector include error components of position, velocity, and attitude as well as accelerometer biases and gyroscope drifts

$$\delta\mathbf{x}_{15\mathrm{x}1}^l = [\delta\mathbf{r}_{3\mathrm{x}1}^l, \delta\mathbf{v}_{3\mathrm{x}1}^l, \delta\boldsymbol{\varepsilon}_{3\mathrm{x}1}^l, \delta\boldsymbol{\omega}_{3\mathrm{x}1}, \delta\mathbf{f}_{3\mathrm{x}1}] \tag{6-10}$$

Where

$\delta \mathbf{r}^{l}_{3\text{x}1} = [\delta\varphi, \delta\lambda, \delta h]^T$ Is the position error vector

$\delta \mathbf{v}^{l}_{3\text{x}1} = [\delta v_n, \delta v_e, \delta v_d]^T$ Is the Earth-referenced velocity error vector

$\delta \boldsymbol{\varepsilon}^{l}_{3\text{x}1} = [\delta p, \delta r, \delta A]^T$ Is the attitude error vector

$\delta \boldsymbol{\omega}_{3\text{x}1} = \left[\delta\omega_x, \delta\omega_y, \delta\omega_z\right]^T$ Is the gyroscope error vector (consisting of drifts)

$\delta \mathbf{f}_{3\text{x}1} = \left[\delta f_x, \delta f_y, \delta f_z\right]^T$ Is accelerometer error vector (consisting of biases)

The system model equation is:

$$\delta \mathbf{x}_k = (I + F\Delta t)\delta \mathbf{x}_{k-1} + G\Delta t w_{k-1} \tag{6-11}$$

where

$$F = \begin{bmatrix} 0_{3\text{x}3} & F_r & 0_{3\text{x}3} & 0_{3\text{x}3} & 0_{3\text{x}3} \\ 0_{3\text{x}3} & 0_{3\text{x}3} & F_v & 0_{3\text{x}3} & R^l_b \\ 0_{3\text{x}3} & F_\varepsilon & 0_{3\text{x}3} & R^l_b & 0_{3\text{x}3} \\ 0_{3\text{x}3} & 0_{3\text{x}3} & 0_{3\text{x}3} & F_\omega & 0_{3\text{x}3} \\ 0_{3\text{x}3} & 0_{3\text{x}3} & 0_{3\text{x}3} & 0_{3\text{x}3} & F_f \end{bmatrix} \tag{6-12}$$

and

$$G = \begin{bmatrix} \sigma_{r,1\text{x}3} \\ \sigma_{v,1\text{x}3} \\ \sigma_{\varepsilon,1\text{x}3} \\ \sigma_{\omega,1\text{x}3} \\ \sigma_{f,1\text{x}3} \end{bmatrix} \tag{6-13}$$

An expanded version of equation (6-11) is:

$$
\begin{bmatrix} \delta\varphi \\ \delta\lambda \\ \delta h \\ \delta v_n \\ \delta v_e \\ \delta v_d \\ \delta p \\ \delta r \\ \delta A \\ \delta\omega_x \\ \delta\omega_y \\ \delta\omega_z \\ \delta f_x \\ \delta f_y \\ \delta f_z \end{bmatrix}_k =
\begin{bmatrix}
1 & 0 & 0 & \frac{\Delta t}{R_M+h} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & \frac{\Delta t}{(R_N+h)\cos\varphi} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & -\Delta t & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & -f_d\Delta t & -f_e\Delta t & 0 & 0 & 0 & C_{11}\Delta t & C_{12}\Delta t & C_{13}\Delta t \\
0 & 0 & 0 & 0 & 1 & 0 & f_d\Delta t & 0 & f_n\Delta t & 0 & 0 & 0 & C_{21}\Delta t & C_{22}\Delta t & C_{23}\Delta t \\
0 & 0 & 0 & 0 & 0 & 1 & f_e\Delta t & -f_n\Delta t & 0 & 0 & 0 & 0 & C_{31}\Delta t & C_{32}\Delta t & C_{33}\Delta t \\
0 & 0 & 0 & \frac{\Delta t}{R_M+h} & 0 & 0 & 1 & 0 & 0 & C_{11}\Delta t & C_{12}\Delta t & C_{13}\Delta t & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -\frac{\Delta t}{R_N+h} & 0 & 0 & 1 & 0 & C_{21}\Delta t & C_{22}\Delta t & C_{23}\Delta t & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -\frac{\Delta t\tan\varphi}{R_N+h} & 0 & 0 & 0 & 1 & C_{31}\Delta t & C_{32}\Delta t & C_{33}\Delta t & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1-\beta_{\omega x}\Delta t & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1-\beta_{\omega y}\Delta t & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1-\beta_{\omega z}\Delta t & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1-\beta_{fx}\Delta t & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1-\beta_{fy}\Delta t & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1-\beta_{fz}\Delta t
\end{bmatrix}
\begin{bmatrix} \delta\varphi \\ \delta\lambda \\ \delta h \\ \delta v_n \\ \delta v_e \\ \delta v_d \\ \delta p \\ \delta r \\ \delta A \\ \delta\omega_x \\ \delta\omega_y \\ \delta\omega_z \\ \delta f_x \\ \delta f_y \\ \delta f_z \end{bmatrix}_{k-1}
+
\begin{bmatrix} \sigma_\varphi \\ \sigma_\lambda \\ \sigma_h \\ \sigma_{vn} \\ \sigma_{ve} \\ \sigma_{vd} \\ \sigma_p \\ \sigma_r \\ \sigma_A \\ \sqrt{2\beta_{\omega x}\sigma_{\omega x}^2} \\ \sqrt{2\beta_{\omega y}\sigma_{\omega y}^2} \\ \sqrt{2\beta_{\omega z}\sigma_{\omega z}^2} \\ \sqrt{2\beta_{fx}\sigma_{fx}^2} \\ \sqrt{2\beta_{fy}\sigma_{fy}^2} \\ \sqrt{2\beta_{fz}\sigma_{fz}^2} \end{bmatrix}
\Delta t\, w_{k-1}
\tag{6-14}
$$

The KF uses a form of feedback control by which the filter estimates the system state x_k at some time k using equation (6-9). Then obtains feedback in the form of noisy measurements z_k as shown in equations (6-15), where H_k is the measurement model as explained in (Titterton and Weston 2004, Noureldin, Karamat et al. 2013), while v_k represents a zero- mean white Gaussian measurements noise whose covariance matrix is R_k.

$$z_k = H_k x_k + v_k, \qquad v_k \sim N(0, R_k) \tag{6-15}$$

In equation (6-44), z_k is a 6x1 vector in our case; three components for position update and three other components for velocity update, more details on how to calculate them are presented in the following section, whereas H_k is a 6x15 matrix as shown follows

$$H_k = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{6-16}$$

6.3 Towards Integrated Multiple Sensor Triads' Navigation System

In this research work, three main objectives were set. These are the accuracy of the portable navigation systems. The efficiency of the computation. Finally, the scalability of the design.

The first main objective here is providing competitive accurate navigation solution using low cost/inaccurate sensors such as those found in most of nowadays portable or wearable devices. One way to achieve this and overcome sensors' inaccuracy is by exploiting all the multiple sensor triads found in portable or wearable devices connected together, and investigating different fusion architectures.

The second objective addresses computation demand. First of all, processing units in portable or wearable devices is not as capable as neither those found in personal computers

correspondents nor in the specialized navigation processors found in high end INS systems. In addition, lower computation power demand can be interpreted as longer battery life time. Since battery consumption is one of the major concerns for portable devices applications, a better navigation solution is the one which enjoys competitive accuracy but lower computation power demand. Consequently, it can run in real-time on a computational capable portable device and consume no more energy than any other typical application.

Third objective is scalability. Two scalability measures are considered here:

1. Maximum Feasible Number of IMUs: Computational efficiency may degrade on adding more devices to be integrated in the solution. After adding certain number of devices, the solution will not be feasible given the fixed and limited computation capabilities of the host device(s). This number of devices is called the maximum feasible number of IMUs, which can be interpreted as a constraint. One of the objective of this research is to remove this constraint in the proposed navigation solution
2. Dynamic number of IMUs: Dynamic number here means that the number of IMUs used by the navigation solution may vary through time and not be fixed throughout the whole trajectory. In other words, forcing no constraints on when a device can join or leave the other devices already being used in the solution. For example, the user may start the navigation solution using only two devices, such as, a smart phone and smart glasses. Afterwards, the user can put on a smart watch, at that time the solution should integrate the smart watch smoothly and without any interruption to or intervention from the user.

To wrap up, scalability here can be stated as: any number of devices can be used, and join or leave the solution at any time. That is, the user is free to have any number of portable or wearable device(s) that may change dynamically at any time.

6.3.1 Known Fusion Architectures

6.3.1.1 Sensor observation fusion architecture

The simplest fusion architecture is the one which fuses sensor observations and feeds the results to a navigation filter as shown in Figure 6-1. Sensor observation fusion architecture enjoys the lowest algorithm complexity. Therefore, having a dynamic number of IMUs as well as a high maximum feasible number of IMUs does not represent any problem for this architecture, due to its simplicity. The real problem this architecture has is that it has the lowest accuracy relative to the other known architectures. Also, more importantly in this multiple devices context, this architecture requires the multiple IMUs to be in a fixed relative position and orientation among each other, which is not applicable to the portable navigation problem, but was mentioned here as a reference for algorithms' complexity. A summary of the pros and cons of this architecture can be shown in Figure 6-2.

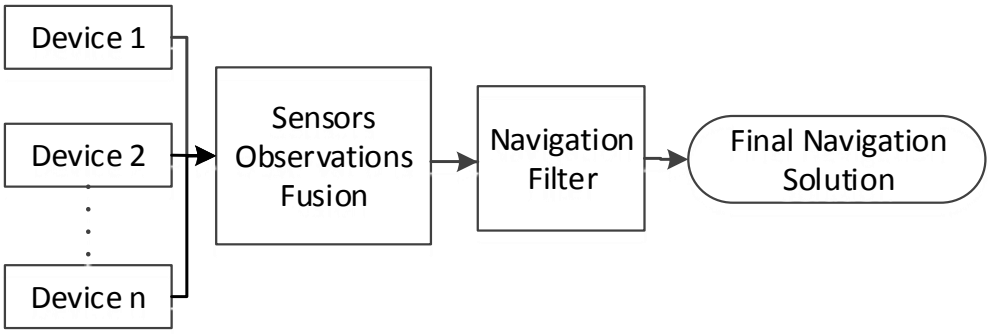


Figure 6-1: Sensor Observations Fusion Architecture

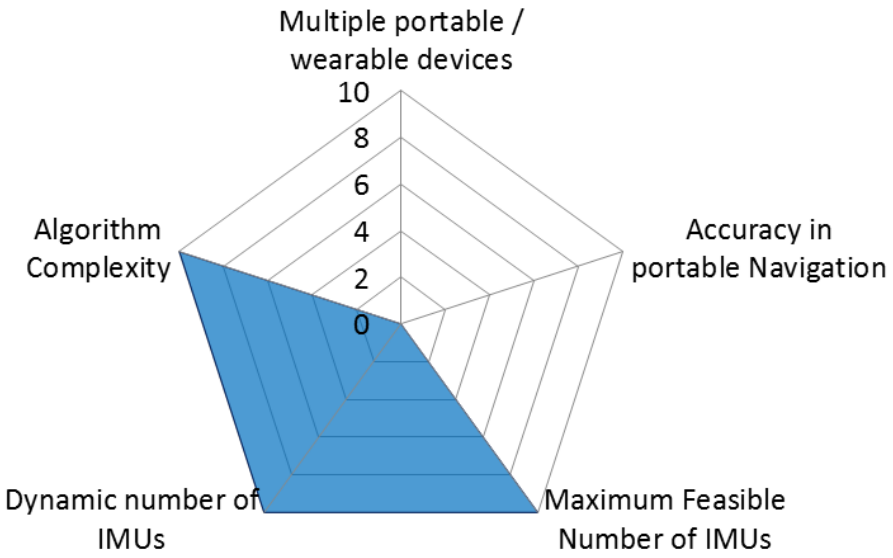


Figure 6-2: Sensor Observations Fusion Overall Performance

6.3.1.2 Centralized filter architecture

On the other hand, the extreme opposite to the previously mentioned architecture is the centralized filter architecture shown in Figure 6-3. This architecture enjoys the best observability among the other architectures, and consequently has the highest accuracy. Moreover, it solves the problem of having fixed relative position and orientation among the involved IMUs (devices) in the navigation solution. However, the main disadvantage of this architecture is its algorithm complexity that is why a low score on algorithm complexity measure is given to this architecture as shown in Figure 6-4. In addition, it does not allow a dynamic number of devices to join and leave the solution at any time, and its maximum feasible number of devices is low due to its high algorithm complexity. Figure 6-4 shows a summary of the pros and cons of the centralized architecture.

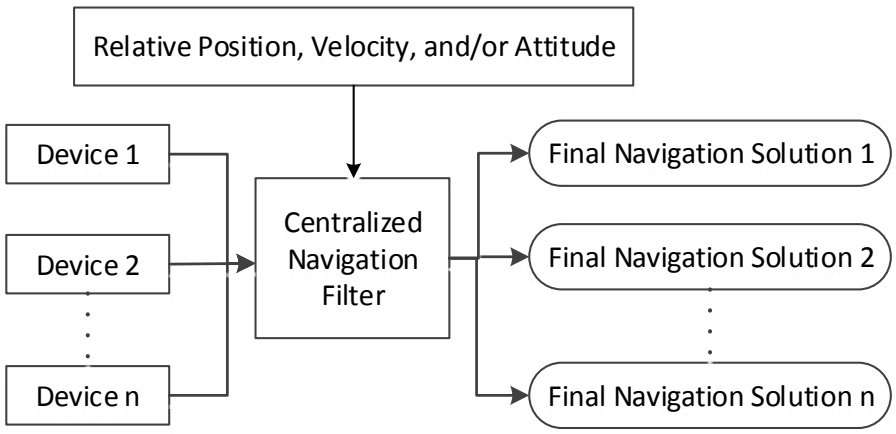


Figure 6-3: Centralized Filter Architecture

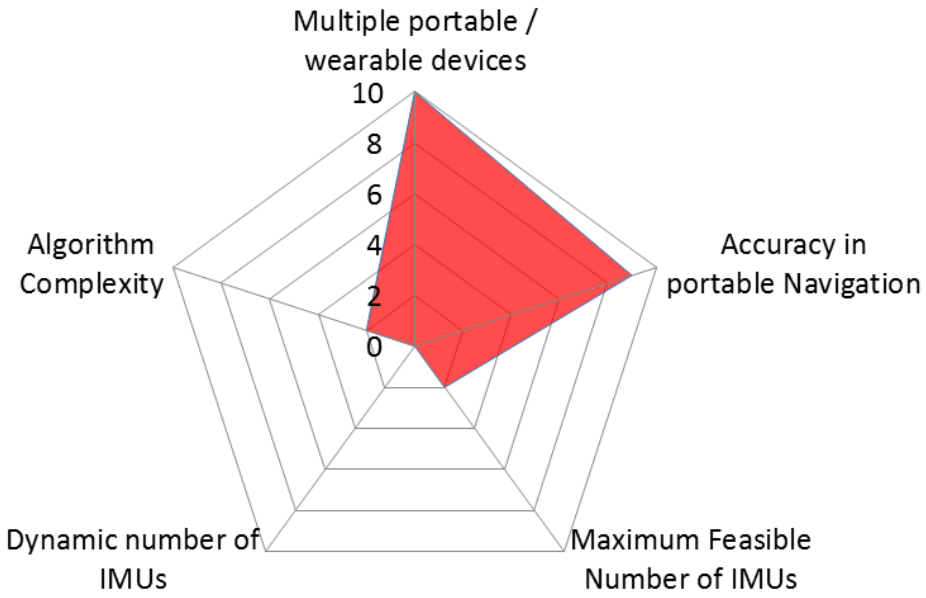


Figure 6-4: Centralized Filter Overall Performance

Kalman Filter processing load depends on the number of states, n, and number of measurements, m. The algorithm complexity of the state propagation portion of KF is $O(n^2)$, while the algorithm complexity of covariance propagation part is $O(n^3)$. Finally, the Kalman gain calculation is $O(mn^2)$ (Groves 2013).

Using the centralized filter architecture to fuse information from multiple sensor assemblies, multiplies both n and m by the number of participating sensor assemblies. For example, if the number of states of the local filter of a single device solution is 15 states, the corresponding number of states of the centralized filter fusing four devices for example will be 60. Which can be translated as 60^2 (or 3600) mathematical operations (mostly multiplication) for state propagation, and 60^3 (or 216000) mathematical operations for covariance propagation, and finally, 6×60^2 (or 21600) for Kalman gain calculation. This vs. the corresponding numbers of the weighted average or the proposed method will be, the original 15 states as is, and therefore 15^2 (or 225) for state propagation, and 15^3 (or 3375) for covariance propagation and finally 6×15^2 (or 1350) for Kalman gain calculation. Figures 6-5 – 6-7 shows the number of mathematical operations required by different KF steps, for both a 15 navigation-state case and 21 state case, starting at using one single sensor assembly till using 6 sensor assemblies.

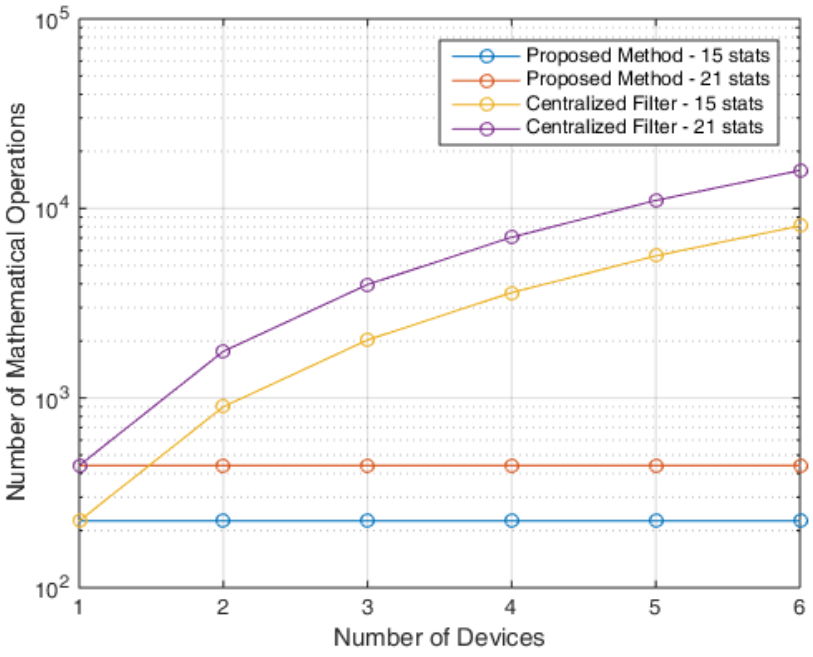


Figure 6-5: Number of mathematical operations for states propagation Vs. Number of participating devices

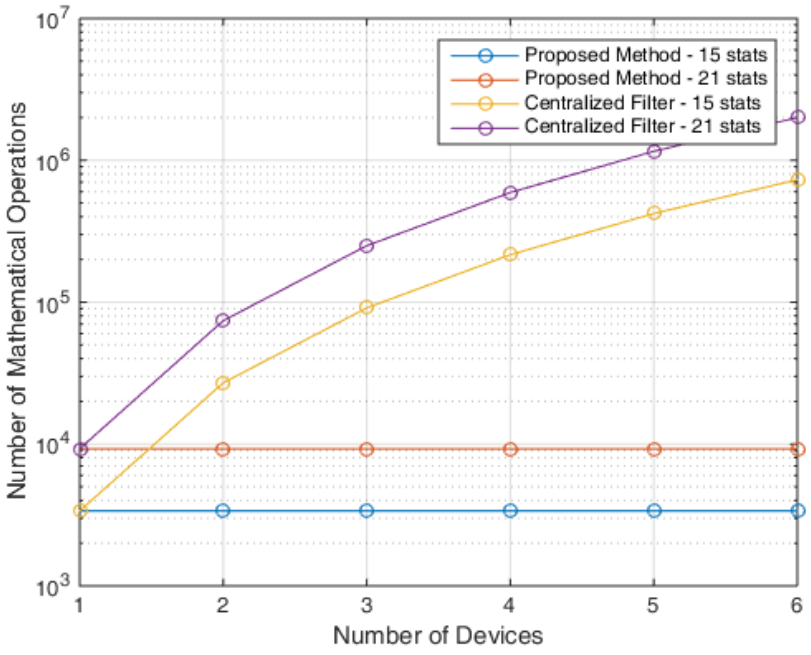


Figure 6-6: No. of mathematical operations for covariance propagation Vs. No. of participating devices

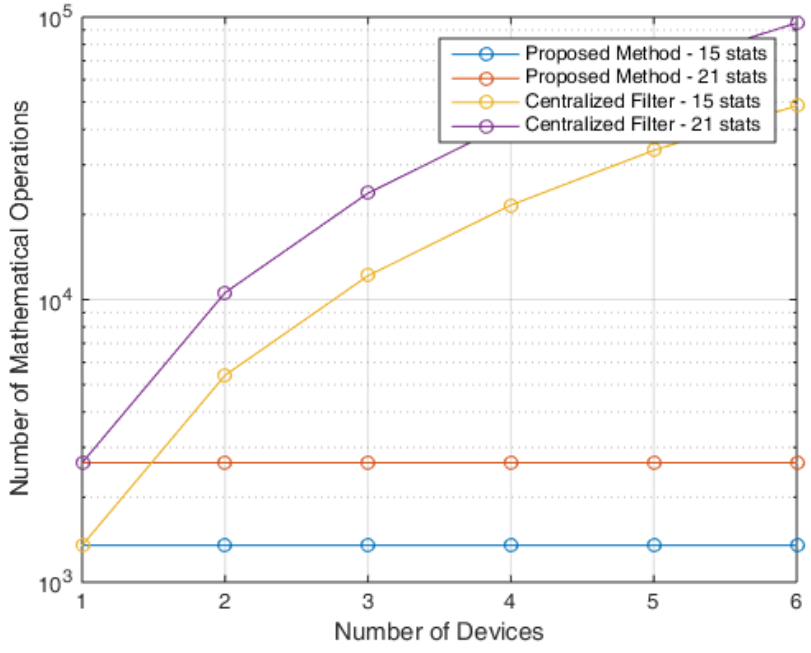


Figure 6-7: No. of mathematical operations for Kalman gain calculation Vs. No. of participating devices

6.3.1.3 Weighted average architecture

In the third architecture discussed here, the fusion problem of the multiple devices is defined as a statistical model estimation problem, which eventually reduces to a weighted average as shown in Figure 6-8. The weighted average architecture represents a mid-solution between the two previously mentioned architectures (which represents two extremes). That is why the weighted average architecture enjoys an average algorithm complexity. Yet did not lose the flexibility of allowing dynamic number of IMUs (or devices) or having low maximum feasible number of IMUs which can be involved in the final navigation solution. The only drawback of this architecture is having average overall accuracy. The pros and cons of the weighted average architecture can be summarized as shown in Figure 6-9.

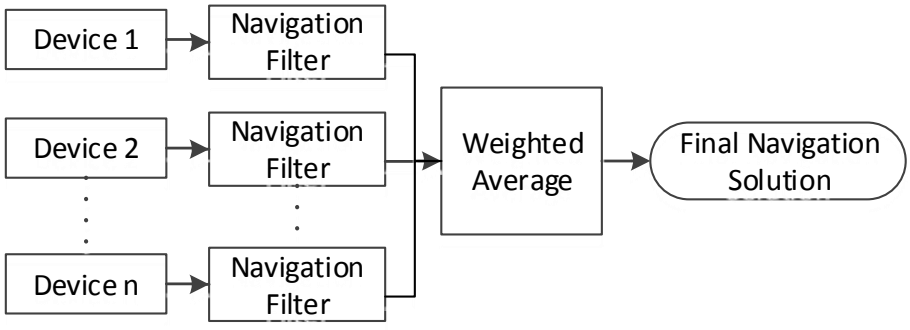


Figure 6-8: Weighted Average Architecture

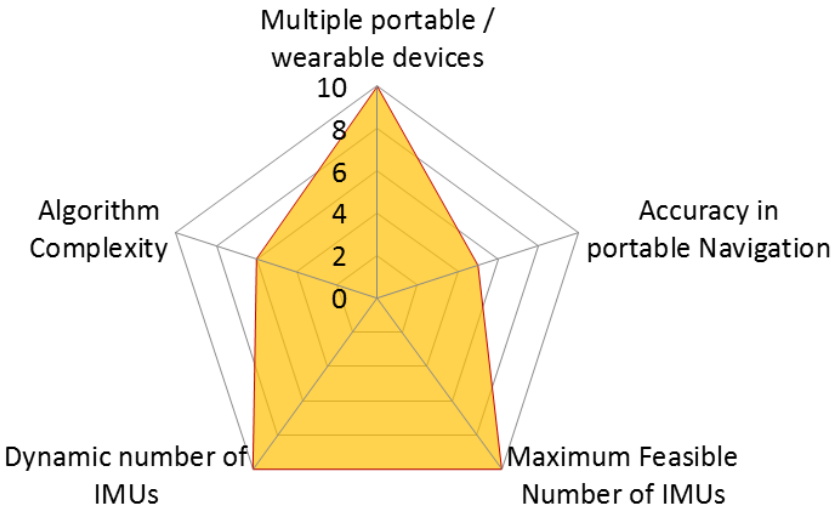


Figure 6-9: Weighted Average Overall Performance

6.3.2 Proposed Fusion Architecture

The main goal here is to provide a better navigation solution using multiple portable devices. Targeting portable devices implicitly implies no predefined configurations to be assumed, such as skew redundant IMU as in (Waegli, Guerrier et al. 2008, Yuksel and El-Sheimy 2011, Martin and Groves 2013), due to the fact that the user may carry the portable devices in any position/orientation relative to each other.

Scalability was another major goal. In this solution, there are no constraints on the number of devices that can be used and integrated in the solution, or when a device can join or leave the remaining devices already used in the solution, i.e. any number of devices can be used and they can join or leave the solution at any time.

Finally, battery saving is one of the major concerns for portable devices applications. Battery saving can be interpreted as lower computation power demand, which is exactly what was achieved in the presented solution. The presented navigation solution enjoys competitive accuracy,

but lower computation power demand. Therefore, it may run in real-time on computational capable portable devices.

In this research, the fusion problem of the multiple devices is still defined as a statistical model estimation problem similar to the weighted average architecture. However, the solution obtained for this estimation problem is further used to update a recursive statistical estimator, which is a Kalman filter in this case.

The benefit of using both a statistical model estimator plus a recursive statistical estimator with a decentralized filter architecture is to overcome the high computation power required by other types of architectures, such as the centralize filter architecture, and at the same time not sacrifice accuracy too much. The result is a highly accurate solution compared to the weighted average alone architecture, competitive to the centralized architecture in terms of accuracy, but with a much lower computation power demand and far more scalable solution than the centralized one.

The most intuitive statistical estimator is the weighted average to be used with the recursive estimator (KF in this case). However, the aim of the following few subsections is to prove that this intuitive choice is the right choice.

6.3.3 3D Position Equations

In this section it is desired to get the optimal 3D position of a platform given the solution (whether it is, for example among others, PDR, INS, or any integrated navigation solution) of two modules hosted by the same platform.

In the first step, letting X_A be the solution (or a component of the solution) of the first sensor assembly, and X_B be the solution (or a component of the solution) of the second sensor assembly, moreover, assuming X_A is independent on X_B, and by applying Maximum Likelihood Estimator (MLE) as follow

$$\max_{\widehat{X}_A \widehat{X}_B} f(\widehat{X}_A \widehat{X}_B | X_A X_B) \quad (6\text{-}17)$$

$$\max_{\widehat{X}_A \widehat{X}_B} f(\widehat{X}_A | X_A) f(\widehat{X}_B | X_B) \quad (6\text{-}18)$$

And assuming Gaussian PDF

$$F = Q e^{G} \quad (6\text{-}19)$$

Where:

$$Q = \frac{1}{\sqrt{(2\pi)^3 \sigma_{X_A} \sigma_{X_B}}}$$

$$G = -\frac{1}{2\sigma_{X_A}^2}\left(\widehat{X}_A - X_A\right)^2 - \frac{1}{2\sigma_{X_B}^2}\left(\widehat{X}_B - X_B\right)^2$$

Also taking into consideration the following constraint

$$\widehat{X}_A = \widehat{X}_B \quad (6\text{-}20)$$

One can define this constrained problem as follow and use Lagrange Multiplier for solving it,

$$L(X, \lambda) = f(X_A, X_B) + \sum_{j=1}^{m} \lambda_j h_j\left(\widehat{X}_A, \widehat{X}_B\right)$$

$$= Q\, e^{G} + \lambda\left(\widehat{X}_A - \widehat{X}_B\right) \quad (6\text{-}21)$$

$$\frac{\partial L}{\partial \widehat{X}_A} = Q\, e^{G} . \frac{\partial G}{\partial \widehat{X}_A} + \lambda$$

$$= Q\, e^{G} . \left(\frac{-1}{\sigma_{X_A}^2}\left(\widehat{X}_A - X_A\right)\right) + \lambda$$

$$= 0 \quad (6\text{-}22)$$

Similarly,

$$\frac{\partial L}{\partial \hat{X}_B} = Q\, e^G . \frac{\partial G}{\partial \hat{X}_B} + \lambda$$

$$= Q\, e^G . \left(\frac{-1}{\sigma^2_{\hat{X}_B}} \left(\hat{X}_B - X_B \right) \right) - \lambda$$

$$= 0 \qquad (6\text{-}23)$$

$$\frac{\partial L}{\partial \lambda} = \hat{X}_A - \hat{X}_B$$

$$= 0 \qquad (6\text{-}24)$$

or simply

$$\hat{X}_A = \hat{X}_B \qquad (6\text{-}25)$$

By adding (6-21) + (6-22)

$$Q\, e^G . \left(\left(\frac{-1}{\sigma^2_{\hat{X}_A}} \left(\hat{X}_A - X_A \right) \right) + \left(\frac{-1}{\sigma^2_{\hat{X}_B}} \left(\hat{X}_B - X_B \right) \right) \right) = 0 \qquad (6\text{-}26)$$

But $Q\, e^G \neq 0$, therefore

$$\frac{1}{\sigma^2_{\hat{X}_A}} \hat{X}_A + \frac{1}{\sigma^2_{\hat{X}_B}} \hat{X}_B = \frac{1}{\sigma^2_{\hat{X}_A}} X_A + \frac{1}{\sigma^2_{\hat{X}_B}} X_B \qquad (6\text{-}27)$$

Also $\hat{X}_A = \hat{X}_B$, therefore

$$\hat{X}_A = \frac{\frac{1}{\sigma^2_{\hat{X}_A}}}{\frac{1}{\sigma^2_{\hat{X}_A}} + \frac{1}{\sigma^2_{\hat{X}_B}}} X_A + \frac{\frac{1}{\sigma^2_{\hat{X}_B}}}{\frac{1}{\sigma^2_{\hat{X}_A}} + \frac{1}{\sigma^2_{\hat{X}_B}}} X_B \qquad (6\text{-}28)$$

6.3.4 3D Velocity

Following the same steps stated in the previous section. Equation (6-27) becomes

$$\frac{1}{\sigma_{\hat{X}_A}^2}\hat{V}_A + \frac{1}{\sigma_{\hat{X}_B}^2}\hat{V}_B = \frac{1}{\sigma_{\hat{X}_A}^2}V_A + \frac{1}{\sigma_{\hat{X}_B}^2}V_B \quad (6\text{-}29)$$

but $\hat{V}_A = \hat{V}_B$, therefore

$$\hat{V}_A = \frac{\frac{1}{\sigma_{V_A}^2}}{\frac{1}{\sigma_{V_A}^2} + \frac{1}{\sigma_{V_B}^2}}V_A + \frac{\frac{1}{\sigma_{V_B}^2}}{\frac{1}{\sigma_{V_A}^2} + \frac{1}{\sigma_{V_B}^2}}V_B \quad (6\text{-}30)$$

6.3.5 Multiple Sensor Assemblies Constraint

In this section it is desired to get the optimal 3D position of a platform given the solution (whether it is, for example among others, PDR, INS, or any integrated navigation solution) of three modules hosted by the same platform, Applying Maximum Likelihood Estimator (MLE)

$$\max_{\hat{X}_A\hat{X}_B\hat{X}_C} f(\hat{X}_A\hat{X}_B\hat{X}_C|X_AX_BX_C) \quad (6\text{-}31)$$

But by assuming that X_A completely independent on X_B and X_C

$$\max_{\hat{X}_A\hat{X}_B\hat{X}_C} f(\hat{X}_A|X_A)f(\hat{X}_B|X_B)f(\hat{X}_C|X_C) \quad (6\text{-}32)$$

assuming Gaussian PDF

$$F = Qe^G \quad (6\text{-}33)$$

Where:

$$Q = \frac{1}{\sqrt{(2\pi)^3\sigma_{X_A}\sigma_{X_B}\sigma_{X_C}}}$$

$$G = -\frac{1}{2\sigma_{\hat{X}_A}^2}\left(\hat{X}_A - X_A\right)^2 - \frac{1}{2\sigma_{\hat{X}_B}^2}\left(\hat{X}_B - X_B\right)^2 - \frac{1}{2\sigma_{\hat{X}_C}^2}\left(\hat{X}_C - X_C\right)^2$$

Also taking into consideration that the problem is constrained by

$$\hat{X}_A = \hat{X}_B \quad (6\text{-}34)$$

$$\widehat{X}_A = \widehat{X}_C \qquad (6\text{-}35)$$

Or

$$\widehat{X}_A + \widehat{X}_B = 2\widehat{X}_C \qquad (6\text{-}36)$$

Using Lagrange Multiplier to solve this constrained problem, therefore

$$L(X,\lambda) = f(X_A, X_B) + \sum_{j=1}^{m} \lambda_j h_j\left(\widehat{X}_A, \widehat{X}_B\right)$$

$$= Q\, e^G + \lambda_1\left(\widehat{X}_A - \widehat{X}_B\right) + \lambda_2\left(\widehat{X}_A + \widehat{X}_B - 2\widehat{X}_C\right) \qquad (6\text{-}37)$$

$$\frac{\partial L}{\partial \widehat{X}_A} = Q\, e^G . \frac{\partial G}{\partial \widehat{X}_A} + \lambda_1 + \lambda_2$$

$$= Q\, e^G . \left(\frac{-1}{\sigma^2_{X_A}}\left(\widehat{X}_A - X_A\right)\right) + \lambda_1 + \lambda_2$$

$$= 0 \qquad (6\text{-}38)$$

Similarly,

$$\frac{\partial L}{\partial \widehat{X}_B} = Q\, e^G . \frac{\partial G}{\partial \widehat{X}_B} - \lambda_1 + \lambda_2$$

$$= Q\, e^G . \left(\frac{-1}{\sigma^2_{X_B}}\left(\widehat{X}_B - X_B\right)\right) - \lambda_1 + \lambda_2$$

$$= 0 \qquad (6\text{-}39)$$

And

$$\frac{\partial L}{\partial \widehat{X}_C} = Q\, e^G . \frac{\partial G}{\partial \widehat{X}_C} - 2\lambda_2$$

$$= Q\, e^G . \left(\frac{-1}{\sigma^2_{X_C}}\left(\widehat{X}_C - X_C\right)\right) - 2\lambda_2$$

$$= 0 \qquad (6\text{-}40)$$

$$\frac{\partial L}{\partial \lambda_1} = \widehat{X}_A - \widehat{X}_B$$

$$= 0 \qquad (6\text{-}41)$$

Or

$$\widehat{X}_A = \widehat{X}_B \qquad (6\text{-}42)$$

$$\frac{\partial L}{\partial \lambda_2} = \widehat{X}_A + \widehat{X}_B - 2\widehat{X}_C$$

$$= 0 \qquad (6\text{-}43)$$

Or

$$\widehat{X}_C = \widehat{X}_A \qquad (6\text{-}44)$$

Adding (6-36) + (6-37) results in

$$Q\, e^G . \left(\left(\frac{-1}{\sigma^2_{X_A}} \left(\widehat{X}_A - X_A \right) \right) + \left(\frac{-1}{\sigma^2_{X_B}} \left(\widehat{X}_B - X_B \right) \right) + 2\lambda_2 \right) = 0 \qquad (6\text{-}45)$$

And adding (6-38) + (6-45) results in

$$Q\, e^G . \left(\left(\frac{-1}{\sigma^2_{X_A}} \left(\widehat{X}_A - X_A \right) \right) + \left(\frac{-1}{\sigma^2_{X_B}} \left(\widehat{X}_B - X_B \right) \right) + \left(\frac{-1}{\sigma^2_{X_C}} \left(\widehat{X}_C - X_C \right) \right) \right)$$

$$= 0 \qquad (6\text{-}46)$$

But since $Q\, e^G \neq 0$, therefore

$$\frac{1}{\sigma^2_{X_A}} \widehat{X}_A + \frac{1}{\sigma^2_{X_B}} \widehat{X}_B + \frac{1}{\sigma^2_{X_C}} \widehat{X}_C = \frac{1}{\sigma^2_{X_A}} X_A + \frac{1}{\sigma^2_{X_B}} X_B + \frac{1}{\sigma^2_{X_C}} X_C \qquad (6\text{-}47)$$

But $\widehat{X}_A = \widehat{X}_B = \widehat{X}_C$

$$\widehat{X}_A = \frac{\frac{1}{\sigma_{X_A}^2}}{\frac{1}{\sigma_{X_A}^2}+\frac{1}{\sigma_{X_B}^2}+\frac{1}{\sigma_{X_C}^2}} X_A + \frac{\frac{1}{\sigma_{X_B}^2}}{\frac{1}{\sigma_{X_A}^2}+\frac{1}{\sigma_{X_B}^2}+\frac{1}{\sigma_{X_C}^2}} X_B + \frac{\frac{1}{\sigma_{X_C}^2}}{\frac{1}{\sigma_{X_A}^2}+\frac{1}{\sigma_{X_B}^2}+\frac{1}{\sigma_{X_C}^2}} X_C \quad (6\text{-}48)$$

From (6-28) and (6-48), one can conclude that

$$\widehat{X}_i = \sum_{j=1}^{N} \frac{\frac{1}{\sigma_{X_i}^2}}{\frac{1}{\sigma_{X_1}^2}+\frac{1}{\sigma_{X_2}^2}+\cdots+\frac{1}{\sigma_{X_N}^2}} X_j \quad (6\text{-}49)$$

For $i = 1, 2, \ldots, N$ where N is the total number of sensor assemblies contributing to the final solution.

In conclusion, letting $\underline{X_i}$ be the user position calculated using the $i^{\underline{th}}$ set of sensor triads (through an integrated navigation solution), and ${\sigma_{X_i}}^2$ be the corresponding co-variance matrix of the position solution of this set of sensor triads for any number of sensor triads, N, involved in calculating the current position. The Kalman filter update is calculated as the difference from the estimated position $\underline{\hat{X}_i}$, which is a weighted average of all participating devices' solutions, using the co-variance matrices of the position for the weighting process as shown in equation (6-33).

This Kalman filter update is used for updating either the main solution (of the device with capable processing power) or all devices' solutions.

Similarly, V_i, ${\sigma_{V_i}}^2$ and $\widehat{V}_i$ are the corresponding values for the velocity solution, and the weighted average estimate for the velocity used for updating the Kalman filter will be as shown in equation (6-34).

$$\widehat{V}_i = \sum_{j=1}^{N} \frac{\frac{1}{\sigma_{V_i}^2}}{\frac{1}{\sigma_{V_1}^2} + \frac{1}{\sigma_{V_2}^2} + \cdots + \frac{1}{\sigma_{V_N}^2}} V_j \qquad (6\text{-}50)$$

For $i = 1, 2, \ldots, N$ where N is the total number of sensor assemblies contributing to the final solution.

Figure 6-10 illustrates the proposed architecture, and a summary of the overall performance is shown in Figure 6-11.

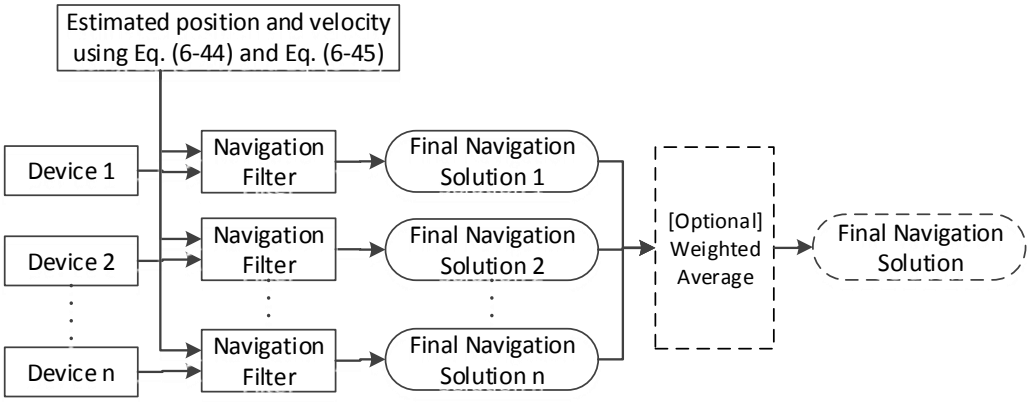


Figure 6-10: Proposed Architecture

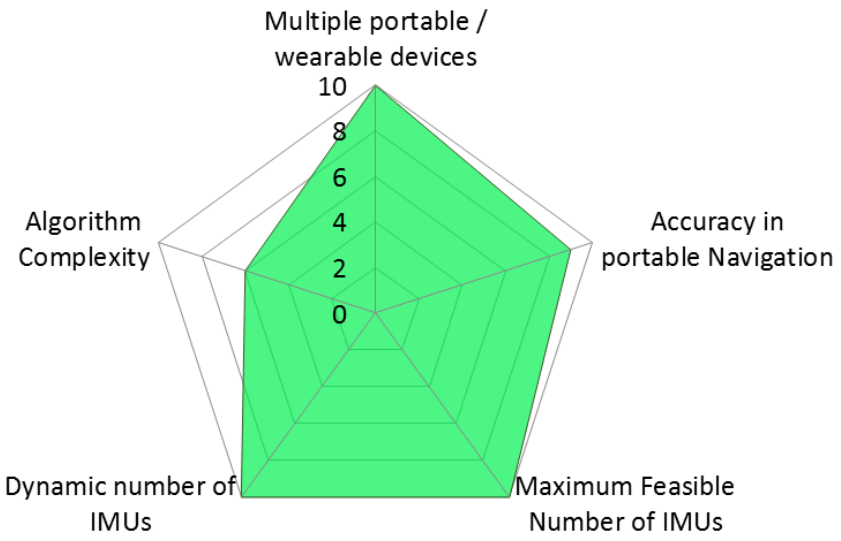


Figure 6-11: Proposed Method Overall Performance

Two major factors distinguish the proposed solution from other existing integrated solutions; first is scalability, second is accuracy while keeping lower computation demand than the state of the art. The proposed method enjoys higher scalability, because it does not force any restrictions on the number of sensor triads that can be used. Also, any number of sensor triads can join or leave the existing set of sensor triads at any time.

The centralized Kalman filter architecture is known in literature for its high accuracy, as well as high computation power (Yuksel and El-Sheimy 2011). The proposed method showed competitive accuracy with far lower computation power demands than the centralized Kalman filter method, which make the proposed method capable of running in real-time. Moreover, scalability is a real challenge for the centralized Kalman filter solution. In the proposed method, filter scalability is not a problem at all.

In Table 6-1, a summary for the pros and cons of various multiple device fusion architectures is presented. Showing that only three architectures; which can be applied in the

multiple portable/wearable devices domain; are the weighted average, the centralized filter, and the proposed architecture. Both the proposed architecture and the centralized filter enjoy the highest accuracy. However, the proposed architecture maintains a higher number of maximum feasible IMUs and allows a dynamic number of IMUs to join or leave the solution at any time. In addition, the proposed architecture has a medium level of algorithm complexity. Also, this architecture has the ability to deal with various types and qualities of IMUs. Overall, considering the aforementioned, the proposed architecture is the most recommended architecture for fusing multiple sensors from multiple devices.

Table 6-1: Architecture Preference as a Function of Development Characteristics

Development Characteristic	Sensor Observations Fusion	Weighted Average (WA)	Proposed Method	Centralized Filter
Multiple portable / wearable devices	No	Yes	Yes	Yes
Accuracy in portable Navigation	N/A	Low	High	High
Maximum Feasible Number of IMUs	High	High	High	Low
Dynamic number of IMUs	Yes	Yes	Yes	No
Algorithm Complexity	Low	Medium	Medium	High
Various Types and Qualities of IMUs Used	Not Recommended	Not Recommended	Recommended	Recommended

6.4 Explanatory Example of the underlying concept for the proposed method

In this subsection, a very simple simulation example is presented to illustrate the difference between the weighted average method and the proposed method. The key difference between the two methods, is that the proposed method not only help improve estimating certain state, but also all of this state' dependents get improved too.

For example, assuming that there is one variable which is the integral of another quantity, this quantity suffers from a noisy bias and it is desirable to estimate the original variable. For simplicity, assume this variable in this example to be the 1D position of certain object, which is the integral of an odometer sensor fixed on that object, and which this odometer sensor suffers from a random noisy bias. To solve this problem, a three states KF is used, one for the 1D position, another for the velocity from the odometer, and finally the bias of the odometer.

The model can be described in the following set of equations

$$\dot{\mathrm{P}} = \mathrm{V} \qquad (6\text{-}46)$$

$$\dot{\mathrm{V}} = 1 \qquad (6\text{-}47)$$

$$\dot{b}_{od} \sim \mathrm{N}(B_i, R_i) \qquad ; \mathrm{i} = 1, 2, \ldots \mathrm{n} \qquad (6\text{-}48)$$

Where n is the number of odometers on the object

Two cases are of particular interest here. First, is when the two biases are having different signs, which will reflect on the position solution as two solutions on opposite sides of the true solution drifting away from each other. Second, is when the two biases are having the same sign, which will reflect on the position solution as two solutions on the same side of the truth solution drifting away from this true solution.

In the first case where the two biases are having different signs, Figure 6-12 shows the result of applying just weighted average to the output of each KF solution of each sensor, while

Figure 6-13 shows the result of applying both position and velocity aiding as explained in the previous section. In Figure 6-12, one solution reached 250 m, while the other one reached about -125 m, and the weighted average is 150 m. While Figure 6-13, all solutions are about 120 m. The explanation for this improvement is that. Not only position and velocity aiding help in estimating the position and velocity more accurately, but also aid in estimating the coupled state to them which is bias. Better estimating the bias help the prediction stage of the KF to perform better in the following steps in time. Therefore, not only each device (or sensor solution) get better output, but their weighted average get better than the weighted average of the original KF solution (without the position and velocity aiding).

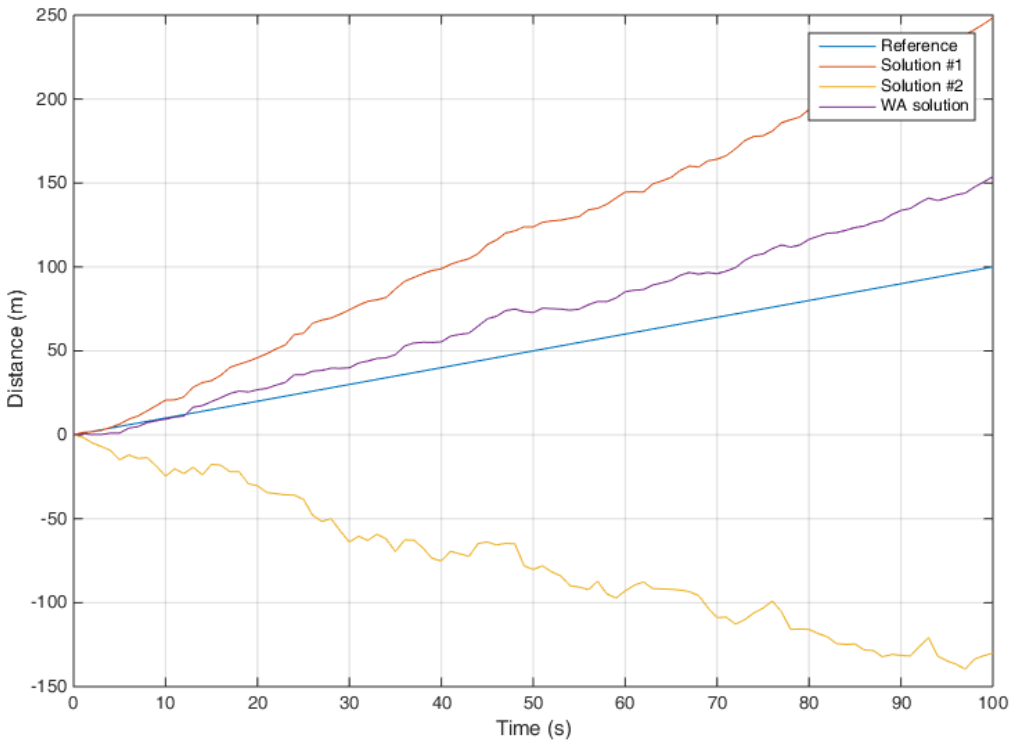


Figure 6-12: Two opposite sides biases - WA method

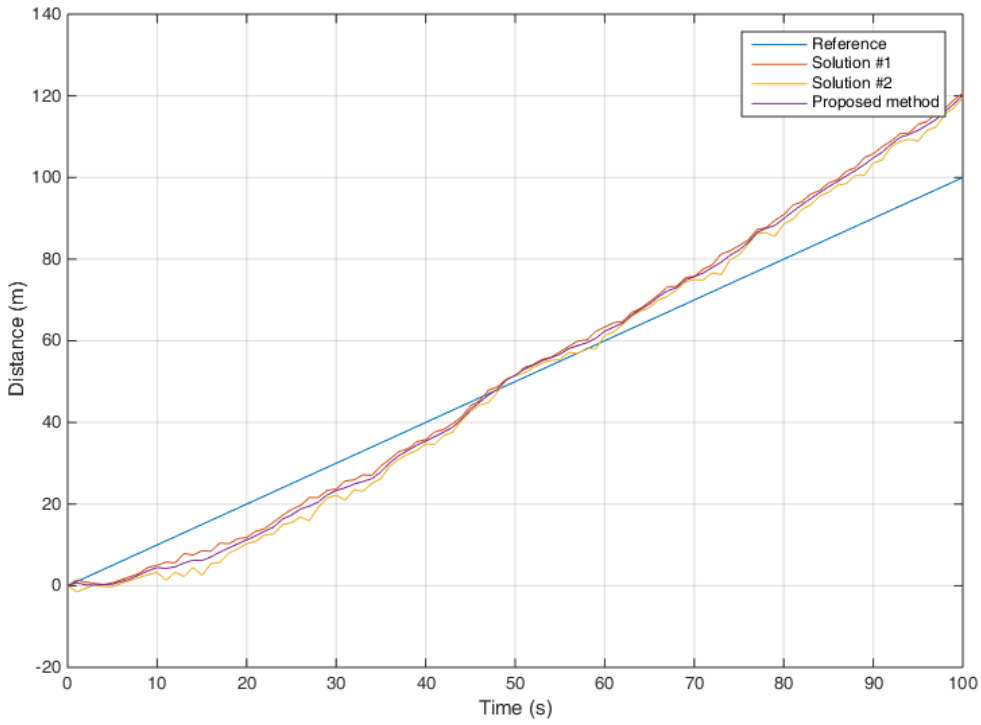


Figure 6-13: Two opposite side biases - Proposed method

In the second case where both biases are having the same sign, Figure 6-14 shows the result of applying just weighted average to the output of each KF solution of each sensor, while Figure 6-15 shows the result of applying both position and velocity aiding as explained in the previous section. This case may look harder, since one solution will improve for sure (the worse one), but what about the good solution? Thanks for Kalman gain, which compare the predicted state to the measurement update. From the covariance, the Kalman gain can trust more its predicted state than the measurement update, leading to the result shown in Figure 6-15. This result show that, even when both biases are having the same sign, the worse solution can get improved, while the better solution is not harm as that much, and leading to a weighted average solution better than the weighted average of the original KF solutions (without the position and velocity aiding).

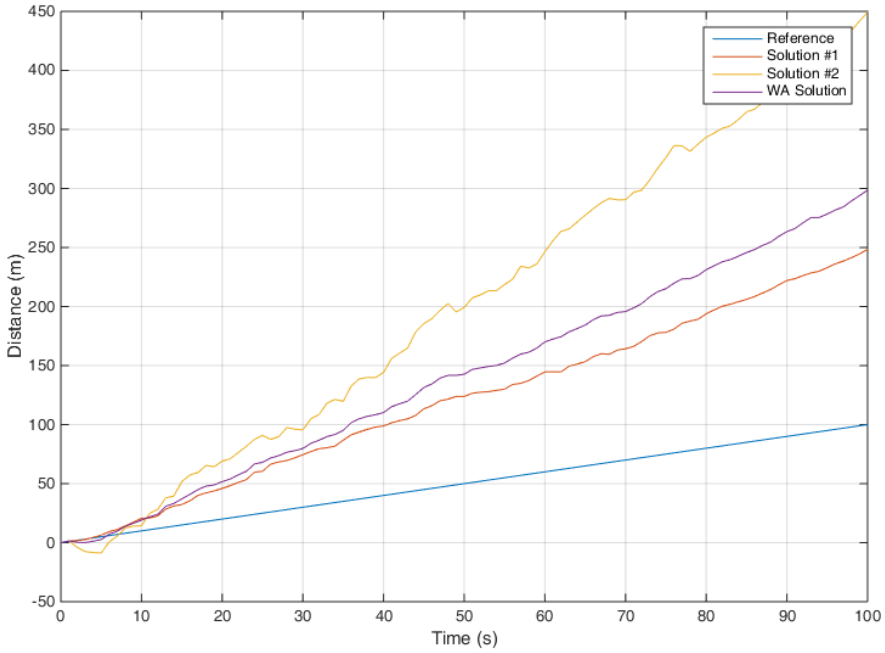


Figure 6-14: Two same side biases - WA method

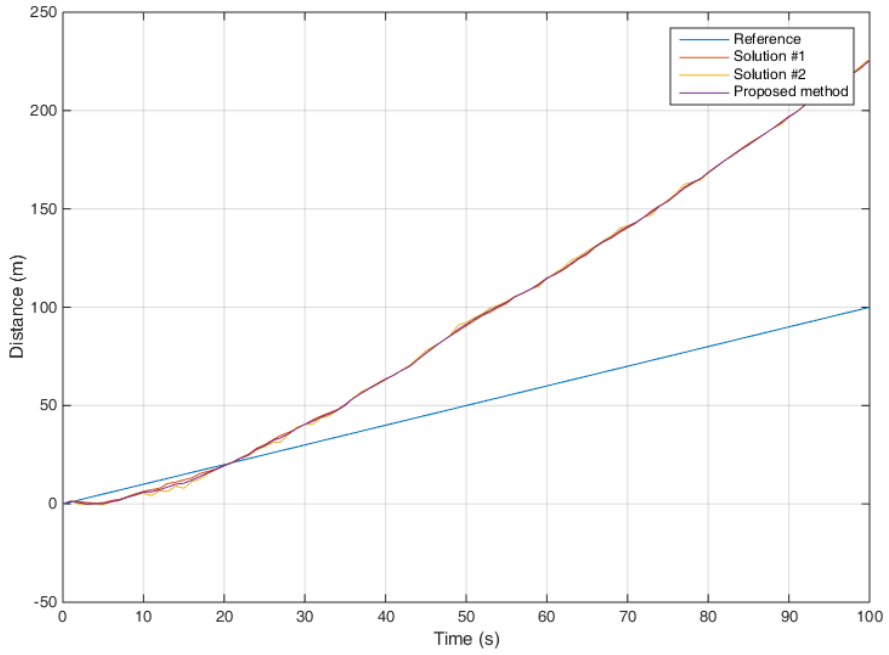


Figure 6-15: Two same side biases - Proposed method

The above results show that, as the computation demand of the centralized filter is enough to exclude it from considering it in the portable navigation problem domain. Bias estimation and stats coupling in the navigation field, consequently the accuracy of the weighted average architecture, are enough to exclude it from consideration in the portable navigation problem.

6.5 Experimental results

A large number of data sets were collected for testing and verification. The data sets were collected using two or more sensors triads in devices, carried at different positions on users' body, such as on the waist, on the wrist, on the head, on the chest, handheld, in pocket, and on ear. The results clearly demonstrates that an enhanced, reliable and more accurate navigation solution can be obtained by integrating multiple sensors triads found in many devices carried by current electronic gadgets' users.

6.5.1 Driving Trajectories

In the following figures, results are showing how much each device's navigation solution is improved after using this type of update during the GNSS signal absence (either while driving or walking). One driving trajectory with simulated 60 seconds GNSS outage plus one indoor walking trajectory of about 7 minutes are demonstrated.

In the following vehicle example trajectory, four devices were used: glasses, belt clip, watch, and smartphone in a pocket. One GNSS outage is shown, which is 60 seconds in duration. Figure 6-16 shows a map for the whole trajectory, while Figure 6-17 shows a zoom in view on the 60 seconds GNSS outage, and Figure 6-18 shows a graph for the same outage shown in Figure 6-17.

In all of the following figures, GNSS is shown in blue and is the reference during the simulated outage in the vehicle example trajectory. In all figures, a device mounted on the head emulating glasses is shown in orange, another device was put in a belt case to emulate a belt clip which includes sensors triads is shown in dark green, a third device was tethered on to the wrist to emulate a smart watch, is shown in bright green, a smartphone itself is shown in cyan, and finally, the proposed combined method is shown in red.

Figure 6-16: Vehicle trajectory 1 map

As shown in Figures 6-17 and 6-18, the proposed method is the closest solution to the GNSS reference, and it is clearly showing how using multiple devices' sensors instead of using each device sensors alone enhances the final navigation solution.

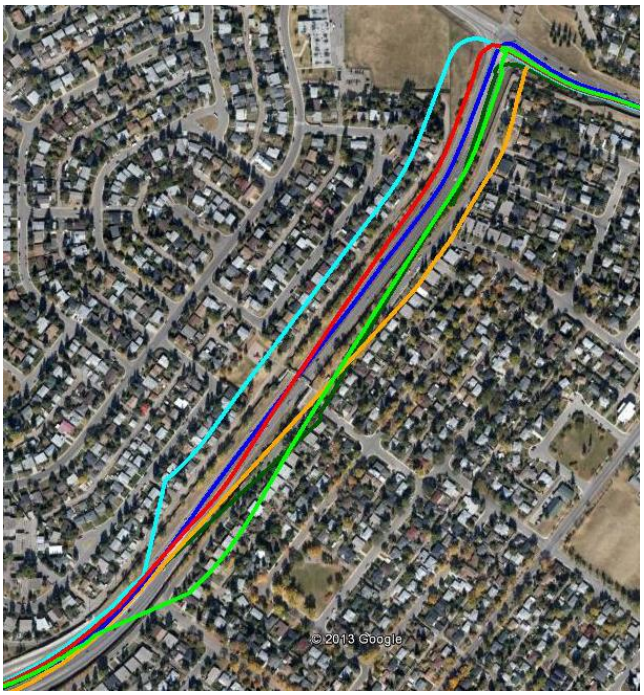


Figure 6-17: Vehicle Trajectory 1 – outage map

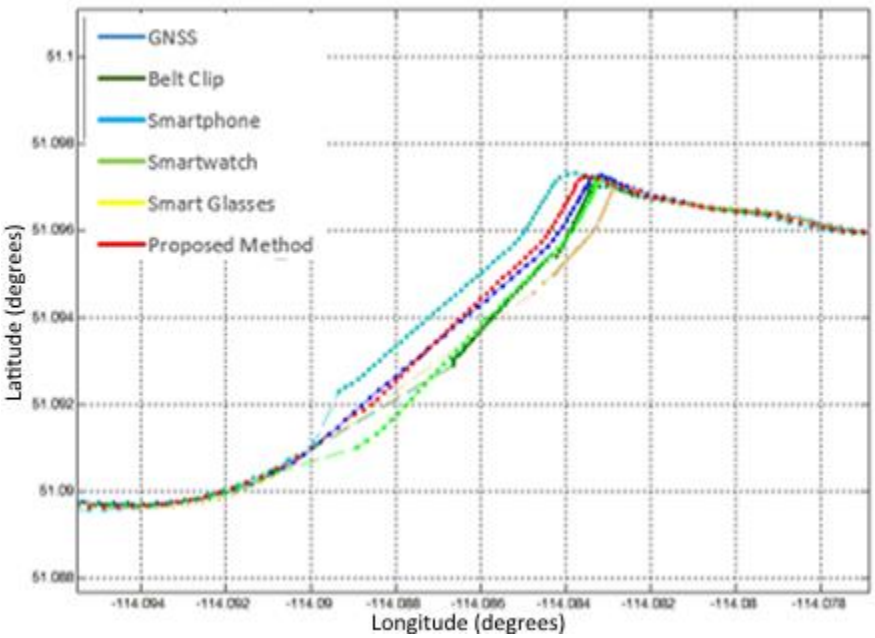


Figure 6-18: Vehicle Trajectory 1 – outage graph

In the second vehicle trajectory, three devices only were used - the smart phone, smart watch, and the belt clip – and the smart phone was in a cradle all the time. One GPS outage was introduced, 60 seconds in duration. As shown in figures 6-19, 6-20, and 6-21, the proposed method significantly improves the overall performance.

Figure 6-19: Vehicle trajectory 2 map

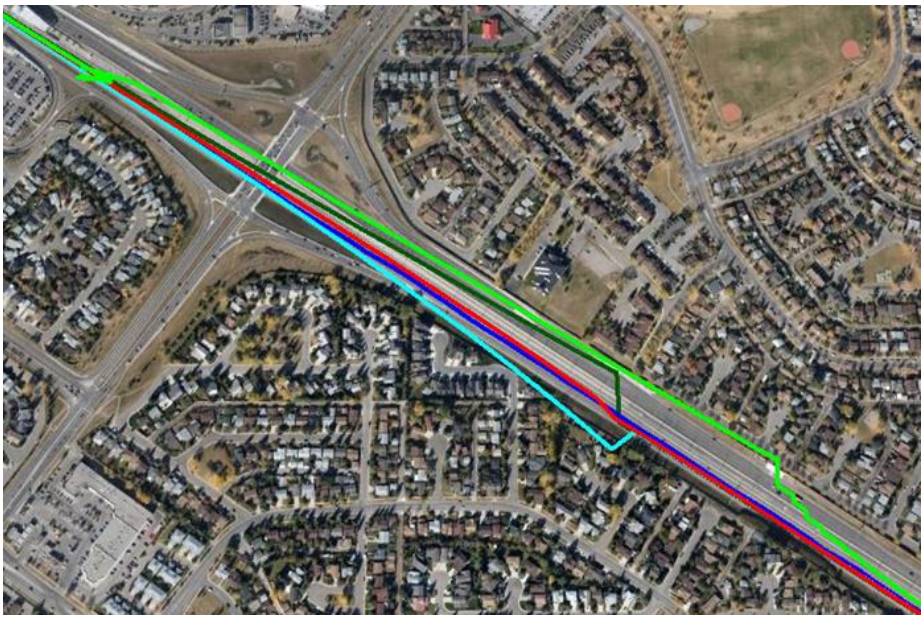


Figure 6-20: Vehicle Trajectory 2 – outage map

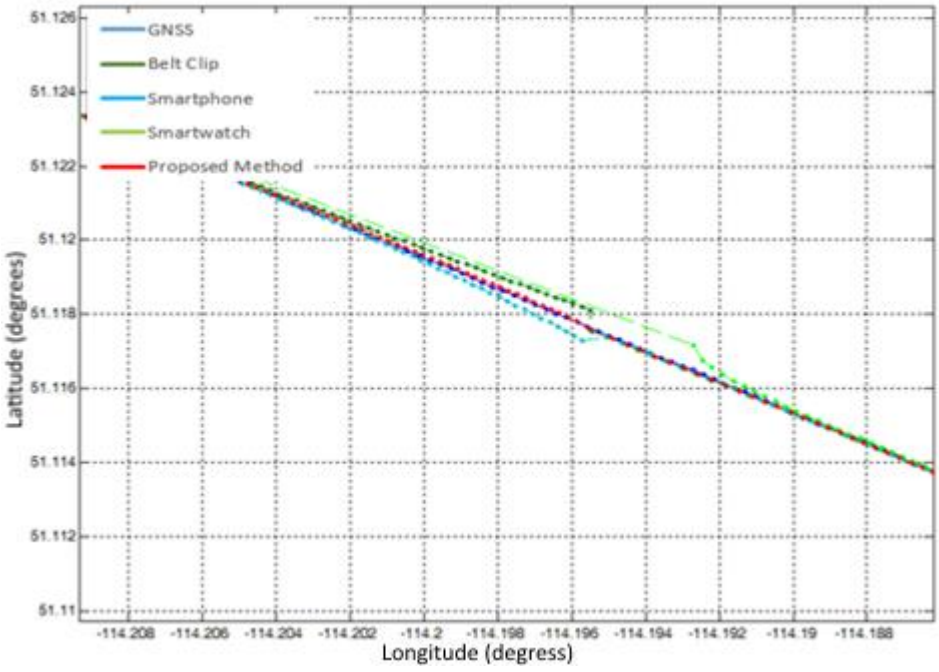


Figure 6-21: Vehicle Trajectory 2 – outage graph

In the following vehicle trajectory, two devices were used - the smartphone and the smart glasses. Figure 6-22 shows the whole trajectory, both the GPS and the proposed method solution coincide for most of the trajectory except for the part in the bottom of the figure where the user enters an underground parkade and loses the GPS signal. The smartphone solution is shown in Figure 6-23 in light blue color. The overall solution shape is correct except for two overshoots, one at the beginning of the entrance of the parkade and another towards the North-East side. This led to the shift in the position of the solution. One more problem with the phone solution is the error of the position solution at the parkade exist (the solution did run short), which can be seen by the jump in the solution near the exit of the parkade and towards the GPS solution once it became available again. Figure 6-23 shows the glasses solution in orange color. The solution has a drift problem, most apparent towards the exit of the parkade since both the entrance and the exit of the parkade are close and parallel to each other. Finally, the red in Figure 6-23 is the solution of proposed method. The proposed method solution does not have the shift found previously in the position solution of the phone, nor does it have the jump near the exit of the parkade. In addition, the proposed method solution does not have the drift problem found in the glasses solution, which determines that the proposed method solution is better than both single device solutions illustrated in Figure 6-23.

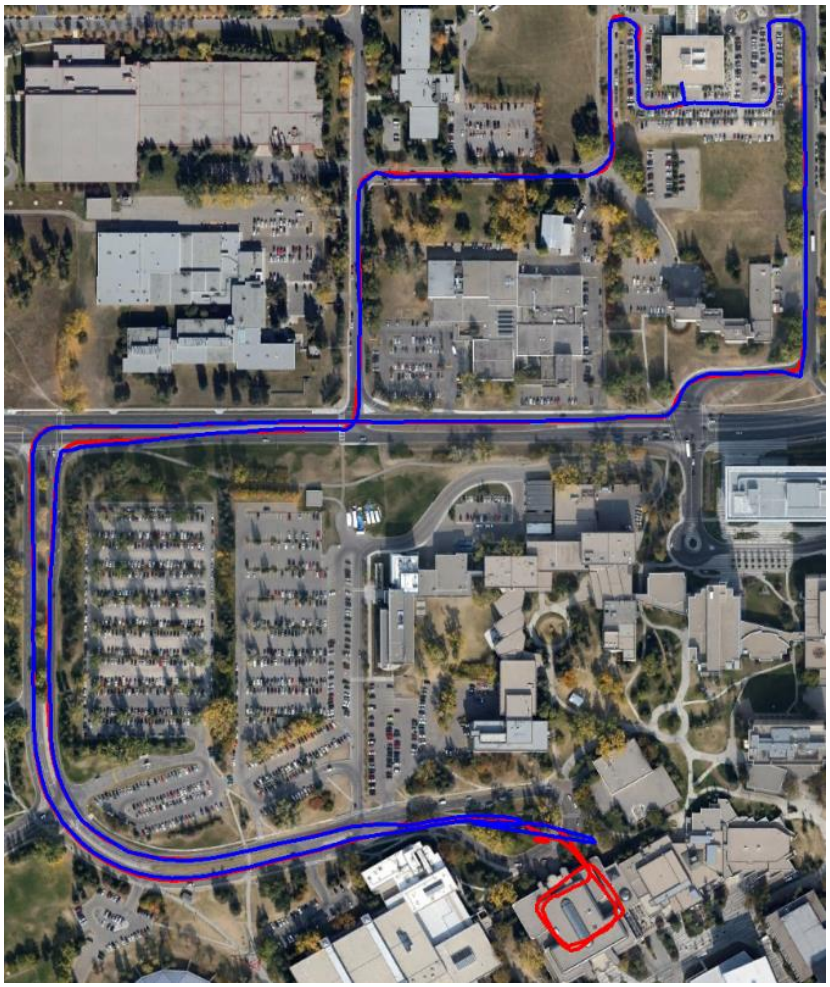


Figure 6-22: Vehicle trajectory 3 map – Proposed method solution VS. GPS

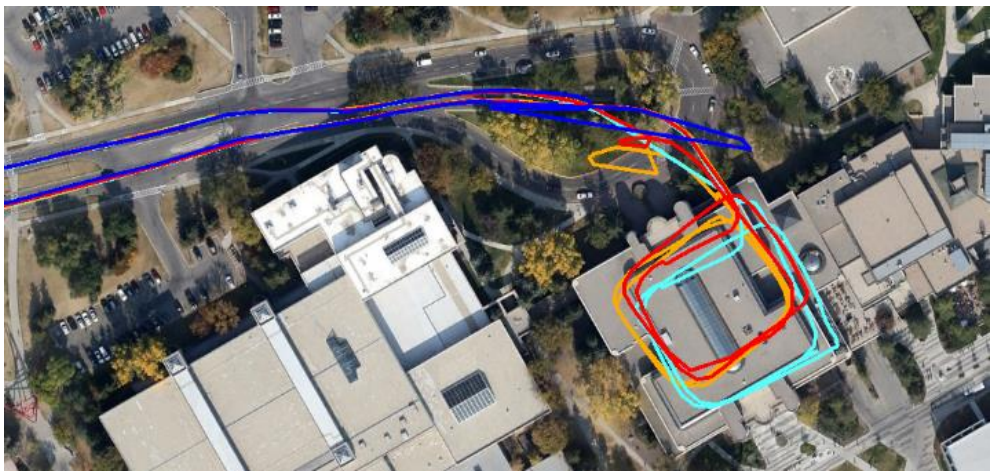


Figure 6-23: Vehicle trajectory 3 – overlay of all solutions after zoom-in on parkade area

In this last vehicle trajectory, two devices were used - the smartphone, and the smartwatch, and a change from driving mode to walking mode took place. The smartphone solution is shown in Figure 6-24. The overall solution shape is correct with exception of the clockwise rotation. Figure 6-25 shows the smartwatch solution. This time the smartwatch solution is again correct in shape, but has a rotation in counter clockwise direction. Figure 6-26 shows the solution of proposed method. The proposed method solution does not have neither rotation, which shows it better than both devices solution alone.

Figure 6-24: Vehicle trajectory 4 – smartphone result

Figure 6-25: Vehicle trajectory 4 – smartwatch result

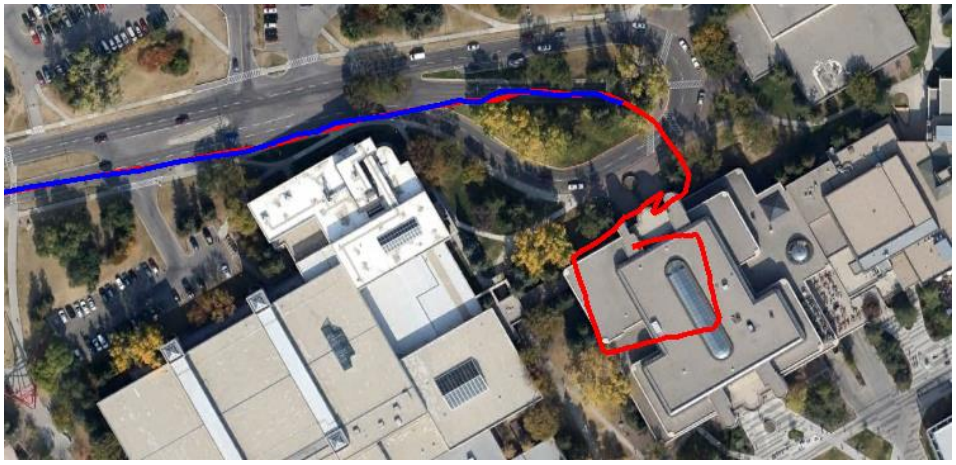


Figure 6-26: Vehicle trajectory 4 – Proposed method solution

6.5.2 Walking Trajectories

In the following walking trajectory, four devices were used: glasses, belt clip, watch, and a free moving smartphone. In this trajectory, the same hand having the watch held the smartphone, and both experienced different use cases as shown in the following table.

Table 6-2: Use case description for each loop in the walking trajectory

Loop #	Use Case
1	Handheld Portrait
2	Handheld Portrait
3	Dangling
4	Handheld Landscape
5	Handheld Portrait
6	Ear

Involving four different use cases for the smartphone and changing the use case each loop shows how challenging this trajectory is, yet very close to real life smartphone daily usage scenario. Users usually use the smartphone in the handheld portrait use case when he/she is trying to navigate to some place inside a building, and switch to dangling while walking for long straight lines or after recognizing his/her path he/she needs to follow to reach the destination. In addition to this, users may wish to use the smartphone in the handheld landscape use case to send a message, or switching back and forth to handheld portrait if he/she lost their path and want to use the navigation application again or finally, making or receiving a call while walking as emulated with the ear use case.

Figures 6-27 – 6-30 show the navigation solution of each device alone, starting by the glasses device in Figure 6-27, then belt clip in Figure 6-28, then free moving watch in Figure 6-29, and finally, the free moving smart phone in Figure 6-30.

Sometimes, the integrated solution is good in terms of tracking the shape of the trajectory, but the heading is drifting as in the case of the glasses and the belt as shown in Figures 6-27 and 6-28 respectively. Other times, the situation is more challenging, as in the case of the smartphone and

the watch as shown in Figures 6-29 and 6-30 respectively, and this was expected as the user keep on changing the smartphone use case each loop.

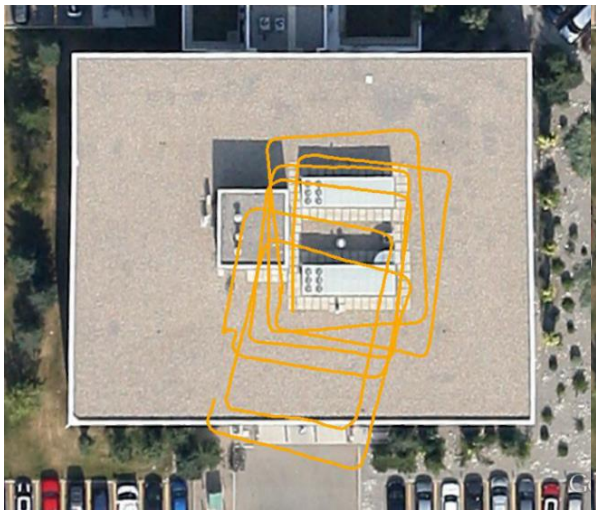


Figure 6-27: Walking trajectory 1 – glasses device

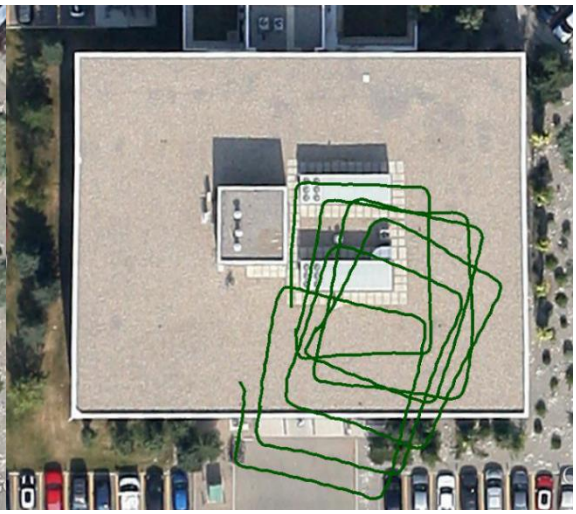


Figure 6-28: Walking trajectory 1 – belt clip

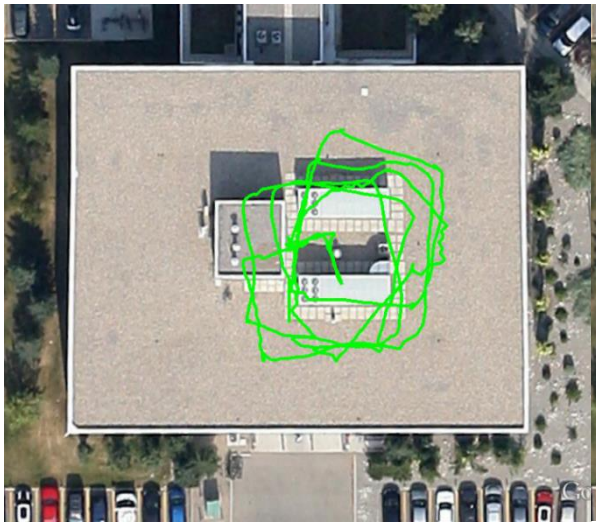


Figure 6-29: Walking trajectory 1 – watch

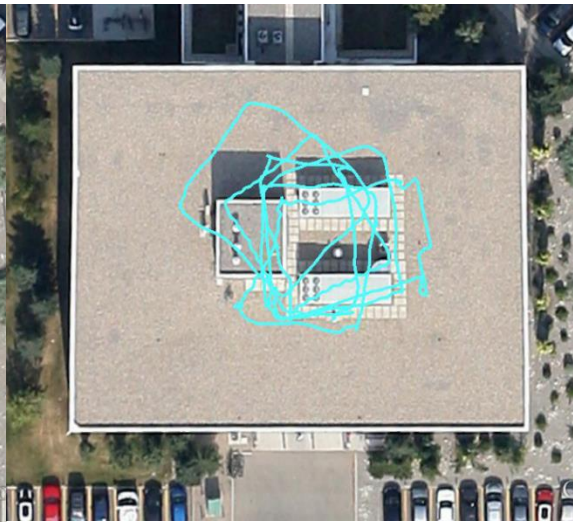


Figure 6-30: Walking trajectory 1 – smartphone

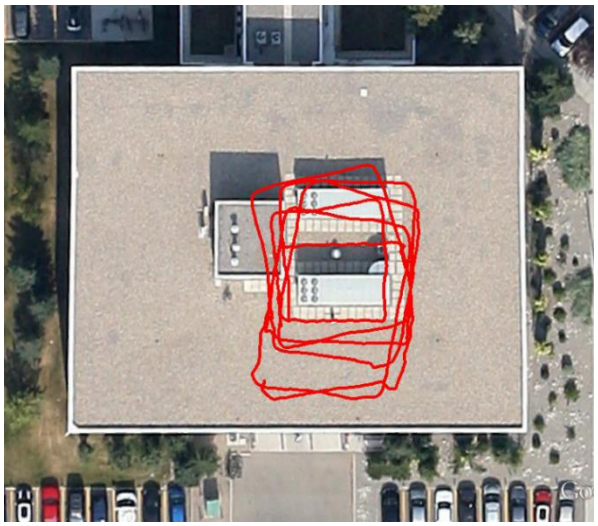


Figure 6-31: Walking trajectory 1 – Proposed method solution

The proposed method solution is shown in Figure 6-31, and as it is shown, the proposed method solution is the best among all other devices' solutions which are depending on the embedded set of sensors in each device alone.

In the second walking trajectory, same track as in the previous trajectory were followed, but in this trajectory both the smart phone and the smart watch were at the same side (left hand of the user). In this trajectory, not only the smart phone is changing its position and direction relative to the user (platform) body, but watch as well. Figures 6-32 – 6-36 again show how the proposed method significantly enhances the overall performance of all participated devices, and specially the free device (the smart phone and the watch).

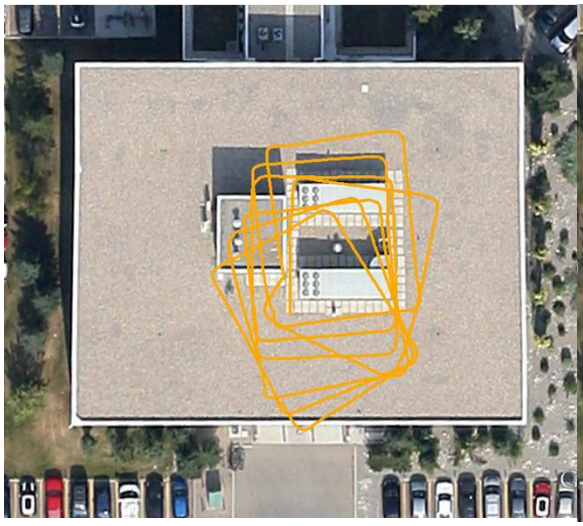


Figure 6-32: Walking trajectory 2 – glasses device

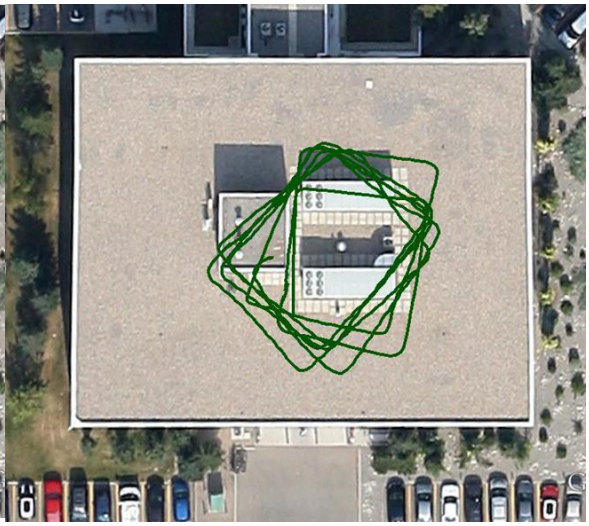


Figure 6-33: Walking trajectory 2 – belt clip

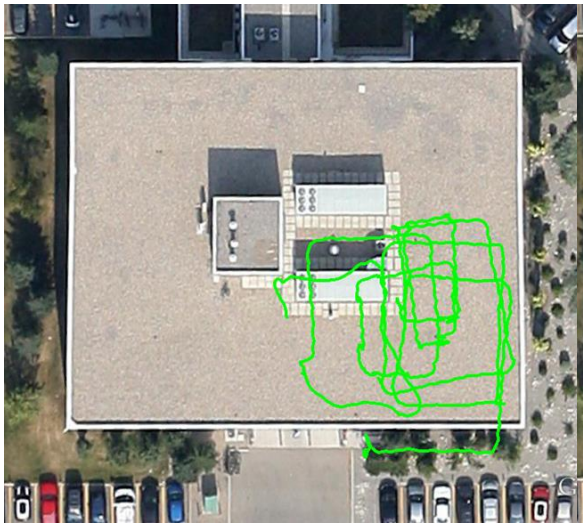


Figure 6-34: Walking trajectory 2 – watch

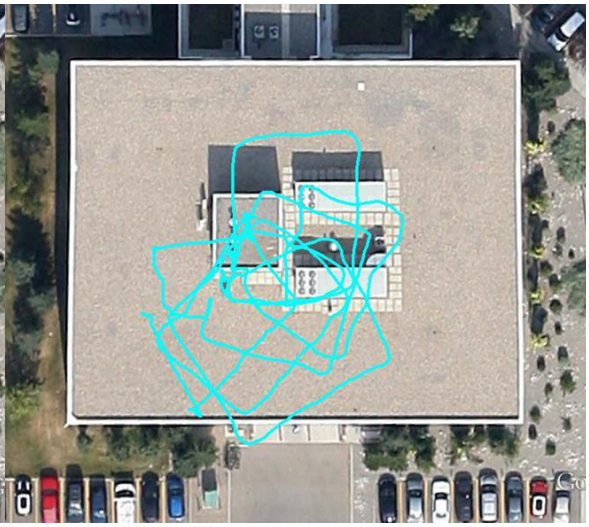


Figure 6-35: Walking trajectory 2 – smartphone

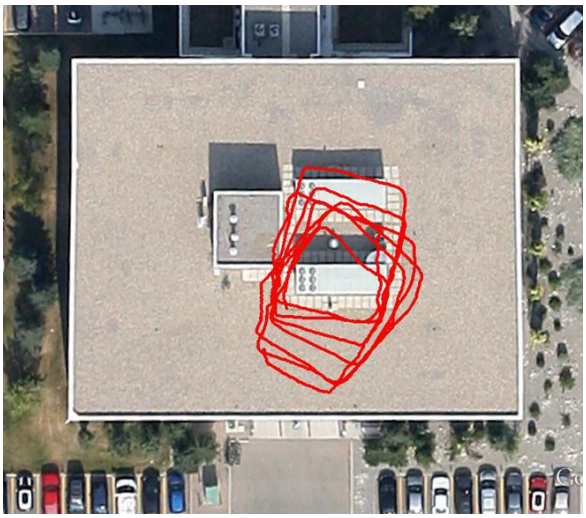


Figure 6-36: Walking trajectory 2 – Proposed method solution

In the third example walking trajectory, only two devices were used: a smartwatch and a smartphone, both in viewing (or compass) mode while the user walked in four loops indoors. The smartwatch and the smartphone solutions are shown in figures 6-37 and 6-38 respectively. As Figure 6-37 illustrates, the smart watch solution suffers from some drift, which causes the solution to rotate in the clockwise direction. On the other hand, Figure 6-38 illustrates a similar drift in the smartphone solution, but in the opposite (counter-clockwise) direction. Figure 6-39 shows the proposed method solution. It shows how integrating both device solutions significantly improves the final solution.

Figure 6-37: Walking Trajectory 3 – smartwatch device

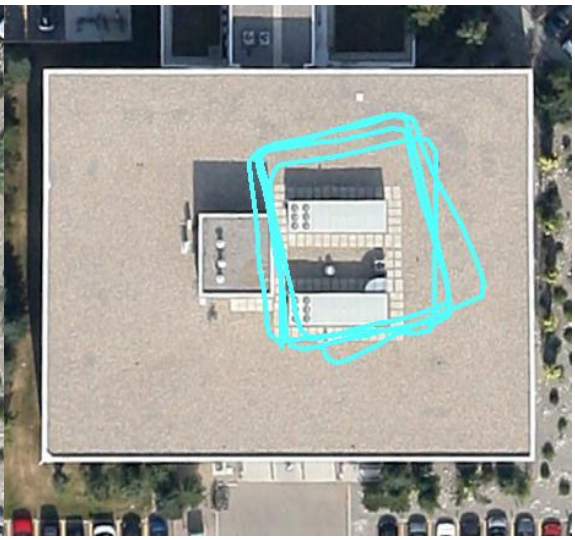


Figure 6-38: Walking Trajectory 3 – smartphone device

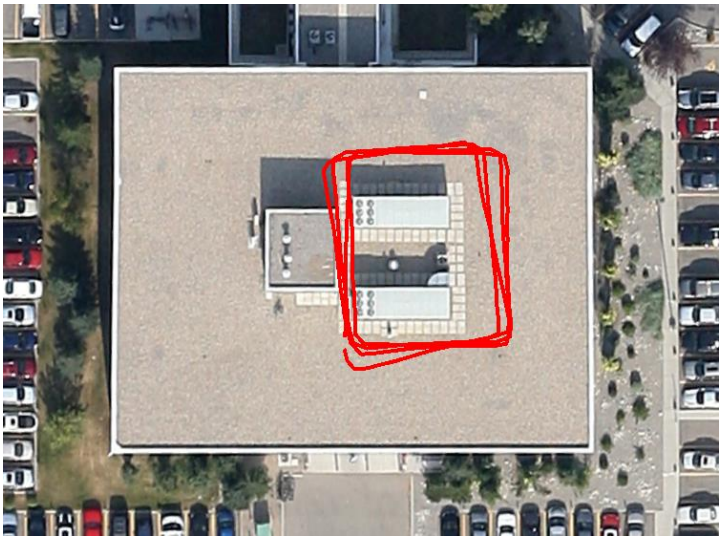


Figure 6-39: Walking Trajectory 3 – Proposed method solution

In the fourth example walking trajectory, a more challenging scenario is presented. Three devices were used: a smart watch, a smart phone, and smart glasses. However, in this trajectory, the same hand wearing the smart watch held the smart phone, and both experienced different use

cases. The trajectory consists of four loops indoors, where in each, the user changes the position of the smart phone; and consequently the smart watch is put in a different position than the previous loop. In the first loop, the user starts with the handheld portrait position, then dangling in the second loop, then ear in the next one, and finally, handheld landscape in the last loop.

Involving four different use cases for the smart phone and changing the use case each loop shows how challenging this trajectory is, yet very close to a real life smartphone usage scenario. Users commonly hold the smartphone in the handheld portrait use case when he/she is trying to navigate inside a building, and switch to dangling while walking for long straight lines or after recognizing his/her path he/she needs to follow to reach the destination. In addition, users may wish to use the smartphone in the handheld landscape use case to send a message, or switching back and forth to handheld portrait if he/she lost their path and want to use the navigation application again or finally, making or receiving a call while walking as emulated with the ear use case.

Figures 6-40 – 6-42 show the navigation solution of each device alone, starting with the smart glasses device in Figure 6-40, then free moving smart phone in Figure 6-41, and finally, the free moving smart watch in Figure 6-42.

Sometimes the standalone device solution is good, in terms of tracking the shape of the trajectory, but the heading drifts over time, as in the case of the glasses shown in Figure 6-40. Other times, the situation is more challenging, as in the case of the smart phone and the smart watch as shown in Figures 6-41 and 6-42 respectively, and this was expected as the user keeps on changing the smartphone use case after each loop.

The proposed method solution is shown in Figure 6-43, and as it is shown, the proposed method solution is the best among all other devices' solutions which are depending on the embedded set of sensor assemblies in each device alone.

Figure 6-40: Walking Trajectory 4 – smart glasses device

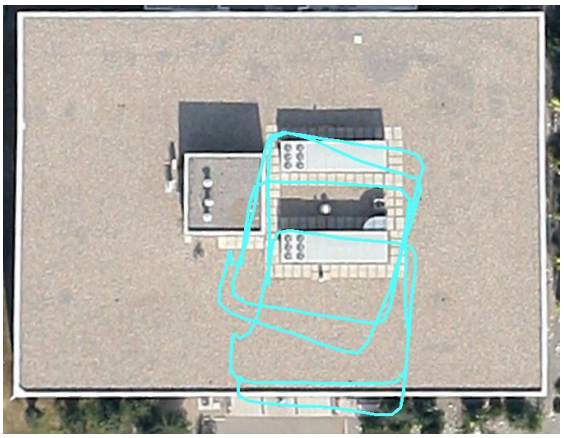


Figure 6-41: Walking Trajectory 4 – smartphone device

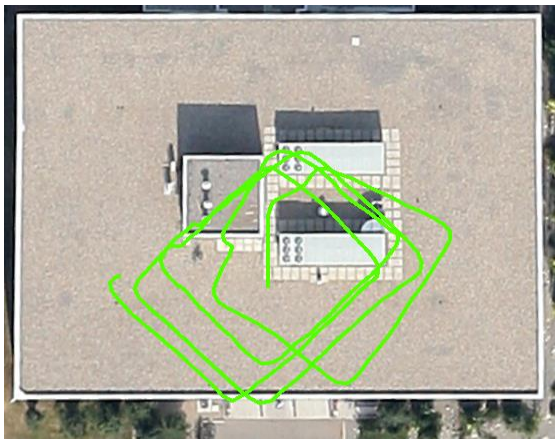


Figure 6-42: Walking Trajectory 4 – smartwatch device

Figure 6-43: Walking Trajectory 4 – Proposed method solution

In the fifth walking trajectory, an Outdoors/Indoors mixed example is presented. Two devices were used, smart glasses and a smart watch. The hand that had the smart watch was dangling during this trajectory. The user starts navigation outdoors where the GPS signal is available, and then walks indoors where he loses the GPS signal. Figure 6-44 shows the smart glasses device solution alone, while Figure 6-45 shows the smartwatch device solution alone.

Figure 6-46 shows the proposed method solution vs GPS. As shown in Figures 6-44 – 6-46, the GPS solution is not available indoors, while the integrated navigation solution is available both indoors and outdoors. Furthermore, as shown in Figure 6-46, integrating multiple sensor assemblies further enhances the integrated solution calculated by the sensor assembly found in each device alone.

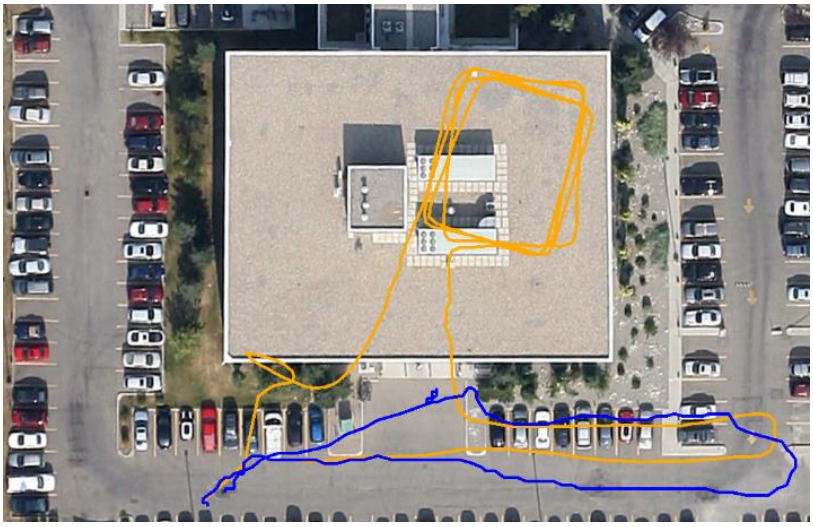


Figure 6-44: Walking Trajectory 5 – smart glasses device VS. GPS

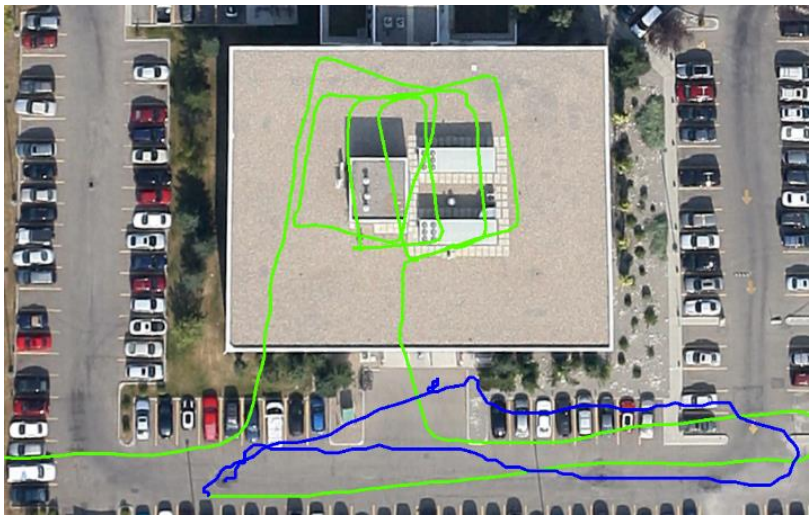


Figure 6-45: Walking Trajectory 5 – smartwatch device VS. GPS

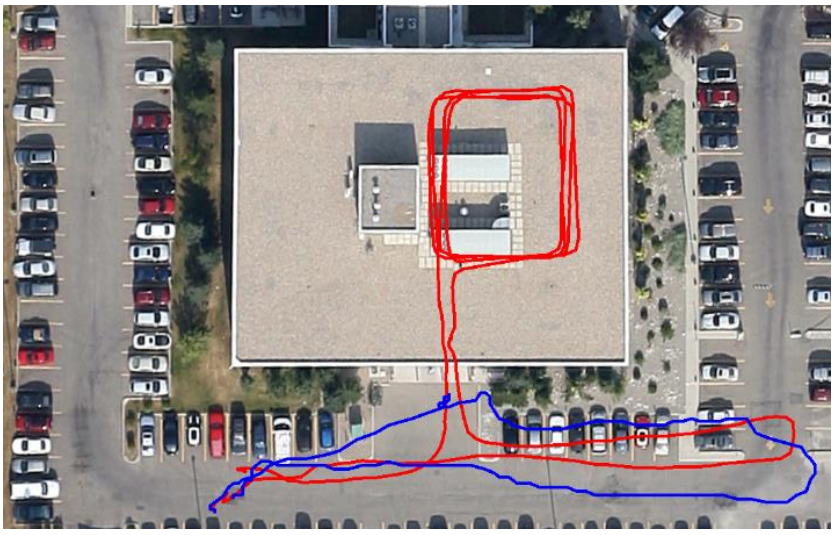


Figure 6-46: Walking Trajectory 5 – Proposed method solution VS. GPS

6.6 Conclusion

In conclusion, two main factors distinguish the proposed solution from other existing multiple triads' integrated solutions. The first main factor is scalability, while the second factor is

accuracy while keeping lower computation demand than the state of the art. In the proposed method, there are no restrictions on the number of the sensor triads which can be used, any number of sensor triads can join or leave the existing set of sensor triads at any time. Furthermore, the proposed method showed competitive accuracy with far lower computation power demand than the state of the art, and that is why the proposed method can be implemented and run in real-time, and on any device with capable computing power, whether a smartphone, tablet, or wearable computing device.

Chapter 7

Conclusions and Recommendations

7.1 Conclusions

Creating synergy among multiple connected portable devices was the ultimate aim of this thesis. Working on delivering an integrated portable navigation solution that is better than each or at least the majority of each portable device solution alone was the true goal.

Multiple ways were investigated to reach this thesis goal. One way was improving specific components or modules of portable navigation. As one example, improving step length using multiple sensor assemblies as shown in Chapter 4. As another example, improving misalignment angle estimation using multiple sensor assemblies as discussed in Chapter 5. However, this was not the only way to achieve this goal. Another way was improving the overall navigation solution at a higher level of abstraction, as by introducing a different kind of KF update, the position and velocity aiding update, as were presented in Chapter 6.

Chapter 4 stressed on the conclusion of the nonlinear relationship between motion parameters such as: step frequency and vertical acceleration variance, and step length. Also how FOS is efficient in selecting the right models parameters from an over-complete set, and how it is fast and can be used later on in online learning and user identification

The presented results demonstrated that the step length estimated from the nonlinear model are more accurate than the one estimated from linear regression model in all the speeds which clearly indicates that the nonlinear model is more capable of solving the varying step length estimation problem for on foot motion without the need of linear approximations either in walking or in running in different speeds.

Chapter 5 focused more on the portability nature of portable navigation problem, and how to obtain an accurate heading estimation through the introduction of a simple yet efficient method to estimate the misalignment angle between a sensor assembly and the hosting platform, whether it is the user's body while walking or a vehicle while driving.

The new proposed method exploits the multiple devices that can be carried by the same user. Not only improving misalignment angle estimation enables portable devises be free to be oriented in any direction while usage (as they are supposed to be), but also the simplicity of the proposed method allows running it in parallel with any other method which uses one single sensor assembly and together enhances the reliability and robustness of the overall misalignment angle estimation algorithm. Results showed in general how effectively using multiple portable and wearable devices can solve the misalignment estimation problem. Whereas, the last couple of results showed specifically how misalignment angle estimation is a key player in estimating platform heading.

On the other hand, instead of following bottom-up approach for solving the portable navigation problem, Chapter 6 followed a top-down approach for solving the portable navigation problem. Its focus was on how to improve the overall navigation solution using multiple sensor assemblies and any underlying techniques, at least as long as a KF was used in estimating the navigation state of the solution.

Two main factors distinguished the proposed solution in Chapter 6 from other existing multiple sensor assemblies' integrated solutions. First is accuracy while keeping lower computation demand than the state of the art, while the second is scalability. The proposed method showed competitive accuracy with far lower computation power demand than the state of the art, and that is why the proposed method was implemented and ran in real-time, and tested on many devices

with capable computing power, whether smartphones, or wearable computing devices. Furthermore, in the proposed method, there are no restrictions on the number of the sensor triads which can be used, any number of sensor triads can join or leave the existing set of sensor triads at any time. Many results at the end of that chapter showed how balancing accuracy and computation demand is possible. And how the proposed architecture can work in both outdoors and indoors, when GNSS signal is available or not.

7.2 Recommendations

In this thesis, the main focus was on inertial sensors, more specifically, accelerometers and gyroscopes. A wider range of sensors such as, magnetometers, Bluetooth (BLE) or WiFi is recommended for further research. In addition, using multiple sensor assemblies (IMUs) with WiFi or GPS in an ultra-coupled fashioned way is strongly recommended since in this way an improved inertial solution is expected to help the WiFi or GPS more than a single inertial solution could.

Improving related components to portable navigation such as context recognition and motion mode such as: walking, running, or driving, and use case classification such as: device handheld or in pocket is another candidate for future work.

In this thesis, Chapter 4 showed how the nonlinear system identification method FOS was efficient. Trying nonlinear system identification methods other than FOS is worth investigation in the future.

Despite the efficiency of the computation demand of the proposed algorithms, still not all these algorithms can operate for always-on applications. Power and software requirements still to be optimized to enable continuous always-on operation. One suggested way to achieve this is through trying one of the modified KF implementations such as one which uses sparse matrix multiplication routines that is more computationally optimized than standard matrix multiplication.

Finally, for accuracy seek and not computation power efficiency, particle filters and/or Neural Networks instead of KF can be investigated with a similar or matching kind of position and velocity aiding update as discussed in Chapter 6.

List of Publications

Patents

[1] Method and apparatus for varying step length estimation using nonlinear system identification
Inventors: **M. Omr,** A. Wahdan, J. Georgy, A. Noureldin
Owner: Invensense Inc.
U.S. Provisional Patent Application No. 61/759,522, filing date: 1 February 2013

[2] Method and apparatus for determination of misalignment between device and vessel using radius of rotation
Inventors: J. Georgy, **M. Omr,** A. Noureldin
Owner: Invensense Inc.
U.S. Provisional Patent Application No. 61/878,336, filing date: 13 September 2013

[3] Method and apparatus for enhanced navigation with multiple sensors assemblies
Inventors: **M. Omr,** J. Georgy, A. Noureldin
Owner: Invensense Inc.
U.S. Provisional Patent Application No. 61/878,952 filing date: 17 September 2013

[4] Method and apparatus for enhanced step detection and step length estimation from multiple sensors assemblies
Inventors: **M. Omr,** J. Georgy, A. Noureldin
Owner: Invensense Inc.
U.S. Provisional Patent Application No. 62/091,442 filing date: 12 December 2014

Submitted Journal Papers

[1] **M. Omr**, J. Georgy, A. Wahdan, M. Elhoushi, and A. Noureldin, "Enhanced Step Length Estimation Using Multiple Portable/Wearable Devices," submitted to IEEE Trans. on Mobile Computing.

[2] **M. Omr**, J. Georgy, and A. Noureldin, "Using Multiple Portable/Wearable Devices for Enhanced Misalignment Estimation in Portable Navigation," submitted to GPS Solutions, Springer.

[3] **M. Omr**, J. Georgy, and A. Noureldin, "Enhanced Navigation Solution Using Multiple Portable/Wearable Consumer Devices," submitted to Hindawi Journal of Sensors.

Published Conference Papers

[1] **M. Omr**, J. Georgy, A. Noureldin, "Using Multiple Sensor Triads for Enhancing the Navigation Solution of Portable and Wearable Devices", ION GNSS+ 2013, Nashville, USA, September 2013.

[2] A. Wahdan, **M. Omr**, J. Georgy, A. Noureldin, "Varying Step Length Estimation Using Nonlinear System Identification", ION GNSS+ 2013, Nashville, USA, September 2013.

[3] **M. Omr**, J. Georgy, A. Noureldin, "Using Multiple Sensor Triads for Enhancing the Misalignment Estimation of a Navigation Solution of Portable and Wearable Devices", IEEE/ION PLANS 2014, California, USA, May 2014.

[4] **M. Omr**, J. Georgy, W. Abdelfatah, D. Auld, and Z. Shen, "Using Multiple Sensor Triads for Enhancing the Navigation Solution of Portable and Wearable Devices", ION GNSS+ 2014, Tampa, USA, September 2014.

[5] A. S. Ali, **M. Omr**, A. Al-Hamad, J. Georgy, "Portable Real-time Indoor/Outdoor Navigation for Runners using Smart Devices", ION GNSS+ 2014, Tampa, USA, September 2014.

References

Abdulrahim, K., et al. (2012). "Using constraints for shoe mounted indoor pedestrian navigation." Journal of Navigation **65**(01): 15-28.

Aimonen, P. (2014). "Basic concept of Kalman filtering." from http://en.wikipedia.org/wiki/File:Basic_concept_of_Kalman_filtering.svg.

Ali, A. S., et al. (2013). "Heading Misalignment Estimation Between Portable Devices and Pedestrians." Proceedings of the 26rd International Technical Meeting of The Satellite Division of the Institute of Navigation (ION GNSS 2013), Nashville, TN, September 2013.

Alvarez, D., et al. (2006). Comparison of step length estimators from weareable accelerometer devices. Engineering in Medicine and Biology Society, 2006. EMBS'06. 28th Annual International Conference of the IEEE, IEEE.

Bancroft, J. B. (2009). Multiple IMU Integration for Vehicular Navigation. Proc. ION GNSS.

Bancroft, J. B. (2010). Multiple Inertial Measurement Unit Integration for Pedestrian Navigation, Dept. of Geomatics Eng., Univ. of Calgary, Calgary, AB, Canada.

Breiman, L., et al. (1984). Classification and regression trees, CRC press.

Burns, A., et al. (2010). "SHIMMER™–A wireless sensor platform for noninvasive biomedical research." Sensors Journal, IEEE **10**(9): 1527-1534.

Cho, S. Y. and C. G. Park (2006). "MEMS based pedestrian navigation system." Journal of Navigation **59**(01): 135-153.

El-Sheimy, N., et al. (2006). "The Utilization of Artificial Neural Networks for Multisensor System Integration in Navigation and Positioning Instruments." Instrumentation and Measurement, IEEE Transactions on **55**(5): 1606-1615.

Ergin, M. A., et al. (2007). Understanding the effect of access point density on wireless LAN performance. Proc. 13th Annu. ACM Int. Conf. Mobile Computing and Networking, ACM.

Gellersen, H. W., et al. (2002). "Multi-sensor context-awareness in mobile devices and smart artifacts." Mobile Networks and Applications **7**(5): 341-351.

Georgy, J., et al. (2011). "Enhanced MEMS-IMU/odometer/GPS integration using mixture particle filter." GPS solutions **15**(3): 239-252.

Georgy, J. and A. Noureldin (2011). "Tightly Coupled Low Cost 3D RISS/GPS Integration Using a Mixture Particle Filter for Vehicular Navigation." Sensors **11**(4): 4244-4276.

Georgy, J., et al. (2012). "Vehicle navigator using a mixture particle filter for inertial sensors/odometer/map data/GPS integration." Consumer Electronics, IEEE Transactions on **58**(2): 544-552.

Grewal, M. S. and A. P. Andrews (2011). Kalman filtering: theory and practice using MATLAB, Wiley-IEEE press.

Grewal, M. S., et al. (2007). Global positioning systems, inertial navigation, and integration, Wiley-Interscience.

Groves, P. D. (2013). Principles of GNSS, inertial, and multisensor integrated navigation systems, Artech House.

Gu, Y., et al. (2009). "A survey of indoor positioning systems for wireless personal networks." Communications Surveys & Tutorials, IEEE **11**(1): 13-32.

Hightower, J. and G. Borriello (2001). "Location systems for ubiquitous computing." Computer **34**(8): 57-66.

Iqbal, U., et al. (2013). "Pseudoranges Error Correction in Partial GPS Outages for a Nonlinear Tightly Coupled Integrated System." Intelligent Transportation Systems, IEEE Transactions on **14**(3): 1510-1525.

Jahn, J., et al. (2010). Comparison and evaluation of acceleration based step length estimators for handheld devices. Indoor Positioning and Indoor Navigation (IPIN), 2010 International Conference on, IEEE.

Jin, Y. (2011). Algorithms and performance analysis for indoor location tracking systems, Dept. of Elec. And Comp. Eng., National Univ. of Singapore, Singapore.

Jin, Y., et al. (2011). A robust dead-reckoning pedestrian tracking system with low cost sensors. Pervasive Computing and Communications (PerCom), IEEE Int. Conf. on, IEEE.

Jyh Ching, J. and C. Yu-Hsuan (2009). "Accounting for data intermittency in a software gnss receiver." Consumer Electronics, IEEE Transactions on **55**(2): 327-333.

Kailath, T., et al. (2000). Linear estimation, Prentice-Hall.

Kaplan, E. D. and C. J. Hegarty (2006). Understanding GPS: principles and applications, Artech House Publishers.

Kern, N., et al. (2003). Multi-sensor activity context detection for wearable computing. Ambient Intelligence, Springer**:** 220-232.

Korenberg, M. J. (1988). "Identifying nonlinear difference equation and functional expansion representations: the fast orthogonal algorithm." Annals of biomedical engineering **16**(1): 123-142.

Korenberg, M. J. (1989). "A robust orthogonal algorithm for system identification and time-series analysis." Biological Cybernetics **60**(4): 267-276.

Korenberg, M. J. and L. D. Paarmann (1989). "Applications of fast orthogonal search: Time-series analysis and resolution of signals in noise." Annals of biomedical engineering **17**(3): 219-231.

Köse, A., et al. (2012). "Bilateral step length estimation using a single inertial measurement unit attached to the pelvis." Journal of neuroengineering and rehabilitation **9**(1): 1-10.

Krogmann, U. (1990). Design considerations for highly reliable hard- and software fault tolerant inertial reference systems. Symposium Gyro Technology 1990, Stuttgart, Federal Republic of Germany.

Liu, H., et al. (2007). "Survey of wireless indoor positioning techniques and systems." Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on **37**(6): 1067-1080.

Lloyd, S. P. (1982). "Least Squares Quantization in PCM." IEEE Transactions on Information Theory **28**: 129–137.

Martin, H. and P. Groves (2013). "A new approach to better low-cost MEMS IMU performance using sensor arrays." ION GNSS.

Maskell, S. and N. Gordon (2001). A tutorial on particle filters for on-line nonlinear/non-Gaussian Bayesian tracking. Target Tracking: Algorithms and Applications (Ref. No. 2001/174), IEE.

Maurer, U., et al. (2006). Activity recognition and monitoring using multiple sensors on different body positions. Wearable and Implantable Body Sensor Networks, 2006. BSN 2006. International Workshop on, IEEE.

Maybeck, P. S. (1982). Stochastic models, estimation and control, Academic.

Minkler, G. and J. Minkler (1993). Theory and application of Kalman filtering, Magellan Book Company.

Noureldin, A., et al. (2009). "Performance enhancement of MEMS-based INS/GPS integration for low-cost navigation applications." Vehicular Technology, IEEE Transactions on **58**(3): 1077-1096.

Noureldin, A., et al. (2013). Fundamentals of Inertial Navigation, Satellite-Based Positioning and Their Integration, Springer.

O'Donovan, K. and S. Ayer (2011). Real-time joint angle measurement using the shimmer wireless sensor platform. Proceedings of the First ACM Workshop on Mobile Systems, Applications, and Services for Healthcare, ACM.

Rekleitis, I. M. (2004). "A particle filter tutorial for mobile robot localization." Centre for Intelligent Machines, McGill University, Tech. Rep. TR-CIM-04-02.

Roetenberg, D., et al. (2009). "Xsens MVN: full 6DOF human motion tracking using miniature inertial sensors." Xsens Motion Technologies BV, Tech. Rep.

Savage, P. G. (1998). "Strapdown inertial navigation integration algorithm design part 1: Attitude algorithms." Journal of guidance, control, and dynamics **21**(1): 19-28.

Shen, Z., et al. (2011). "Low cost two dimension navigation using an augmented Kalman filter/Fast Orthogonal Search module for the integration of reduced inertial sensor system and Global Positioning System." Transportation Research Part C: Emerging Technologies **19**(6): 1111-1132.

Shin, S., et al. (2007). Adaptive step length estimation algorithm using low-cost MEMS inertial sensors. Sensors Applications Symposium, 2007. SAS'07. IEEE, IEEE.

Skog, I., et al. (2014). An open-source multi inertial measurement unit (MIMU) platform. Inertial Sensors and Systems (ISISS), 2014 International Symposium on.

Skog, I., et al. (2014). Pedestrian tracking using an IMU array. Electronics, Computing and Communication Technologies (IEEE CONECCT), 2014 IEEE International Conference on.

Sun, G., et al. (2005). "Signal processing techniques in network-aided positioning: a survey of state-of-the-art positioning designs." Signal Processing Magazine, IEEE **22**(4): 12-23.

Sun, Z., et al. (2009). "Activity classification and dead reckoning for pedestrian navigation with wearable sensors." Measurement Science and Technology **20**(1): 015203.

Syed, Z., et al. (2013). "Showing Smartphones the Way Inside Real-time, Continuous, Reliable, Indoor/Outdoor Localization." GPS World, March 2013, VOL. 24, NO. 3, 2013.

Syed, Z., et al. (2012). "Real-time, Continuous and Reliable Consumer Indoor/Outdoor Localization for Smartphones." Proceedings of the 25rd International Technical Meeting of The Satellite Division of the Institute of Navigation (ION GNSS 2012), Nashville, TN, September 2012.

Thrun, S., et al. (2005). Probabilistic robotics, MIT press.

Titterton, D. and J. Weston (2004). "Strapdown Inertial Navigation Technology. 2-nd Edition." The Institution of Electronical Engineers, Reston USA.

Waegli, A., et al. (2008). Redundant MEMS-IMU integrated with GPS for Performance Assessment in Sports. Position, Location and Navigation Symposium, IEEE/ION, IEEE.

Weinberg, H. (2002). "Using the ADXL202 in pedometer and personal navigation applications." Analog devices AN-602 application note.

Welch, G. and G. Bishop (1995). An introduction to the Kalman filter.

Wikipedia (2014). "Wearable computer." from http://en.wikipedia.org/w/index.php?title=Wearable_computer&oldid=602360935.

Yuksel, Y. (2011). "Design and analysis of inertial navigation systems with skew redundant inertial sensors." UCGE Report **20328**.

Yuksel, Y. and N. El-Sheimy (2011). "An optimal sensor fusion method for skew-redundant inertial measurement units." Journal of Applied Geodesy **5**(2): 99-115.

Zheng, Y., et al. (2010). "Spreading code phase measurement technique for snapshot GNSS receiver." Consumer Electronics, IEEE Transactions on **56**(3): 1275-1282.

Zijlstra, W. and A. L. Hof (1997). "Displacement of the pelvis during human walking: experimental data and model predictions." Gait & posture **6**(3): 249-262.