

2016

Improving GPS Global Navigation Accuracy for Connected Vehicles in an Urban Canyon

Ian J. Douglas
University of Windsor

Follow this and additional works at: <http://scholar.uwindsor.ca/etd>

Recommended Citation

Douglas, Ian J., "Improving GPS Global Navigation Accuracy for Connected Vehicles in an Urban Canyon" (2016). *Electronic Theses and Dissertations*. Paper 5815.

This online database contains the full-text of PhD dissertations and Masters' theses of University of Windsor students from 1954 forward. These documents are made available for personal study and research purposes only, in accordance with the Canadian Copyright Act and the Creative Commons license—CC BY-NC-ND (Attribution, Non-Commercial, No Derivative Works). Under this license, works must always be attributed to the copyright holder (original author), cannot be used for any commercial purposes, and may not be altered. Any other use would require the permission of the copyright holder. Students may inquire about withdrawing their dissertation and/or thesis from this database. For additional inquiries, please contact the repository administrator via email (scholarship@uwindsor.ca) or by telephone at 519-253-3000ext. 3208.

Improving GPS Global Navigation Accuracy for Connected Vehicles in an Urban Canyon

By:

Ian J. Douglas

A Thesis

Submitted to the Faculty of Graduate Studies
through the School of Computer Science
in Partial Fulfillment of the Requirements for
the Degree of Master of Science at the
University of Windsor

Windsor, Ontario, Canada

2016

© 2016 Ian J. Douglas

Improving GPS Global Navigation Accuracy for Connected Vehicles in an Urban Canyon

By:
Ian J. Douglas

APPROVED BY:

Dr. Bill Anderson
Dept. Political Science
Chair, Cross-Border Institute

Dr. Robert Kent
School of Computer Science

Dr. Arunita Jaekel, Advisor
School of Computer Science

September 14, 2016

Author's Declaration of Originality

I hereby certify that I am the sole author of this thesis and that no part of this thesis has been published or submitted for publication.

I certify that, to the best of my knowledge, my thesis does not infringe upon anyone's copyright nor violate any proprietary rights and that any ideas, techniques, quotations, or any other material from the work of other people included in my thesis, published or otherwise, are fully acknowledged in accordance with the standard referencing practices. Furthermore, to the extent that I have included copyrighted material that surpasses the bounds of fair dealing within the meaning of the Canada Copyright Act, I certify that I have obtained a written permission from the copyright owner(s) to include such material(s) in my thesis and have included copies of such copyright clearances to my appendix.

I declare that this is a true copy of my thesis, including any final revisions, as approved by my thesis committee and the Graduate Studies office, and that this thesis has not been submitted for a higher degree to any other University or Institution.

Abstract

Connected Vehicles are expected to provide a major improvement in road safety. By broadcasting Basic Safety Messages (BSM) using Dedicated Short Range Communications (DSRC) all connected vehicles will have situational awareness of other connected vehicles in the area near them, and capability to provide ample warning of impending collisions. These systems rely on highly accurate GPS location data.

GPS by design expects a clear line of sight (LoS) to four or more satellites for accuracy. City roads are often surrounded by buildings. These structures create areas isolated from sky views. Intelligent Transportation System (ITS) researchers have called these areas “urban canyons”. Buildings may block and/or bounce satellite signals, which can cause receivers to ‘see’ these signals either directly, indirectly, or both direct and indirect signals at the same time—which is the so-called multipath problem.

Driving test results have been published which demonstrate the challenge. ITS researchers have noticed that position data taken by on-board units (OBU’s) contain these anomalies. When analyzed, these plots show vehicles as if they were driving through buildings. This is not helpful in preventing collisions. I will show that there is a data based approach to identify when Global Navigational Satellite System (GNSS) receivers are identifying impossible position results. I will also show a method using other available CAN bus data to interpolate expected geographic location and eliminate sending erroneous position reports.

Dedication

My parents acknowledged and encouraged my intense curiosity at an early age. Whether it was dad showing me how take apart a bicycle and laying the parts out so that I could get them to go back together, or mom building a crystal radio set with me, they spent time and encouraged me to read, assemble and solve problems. Getting a good start is so very important in the journey, and I can't thank them enough.



My wife, Mary, knew I had always wanted to dive into research. My undergraduate studies were complete several years before a masters program existed at our school. I wish that I had pursued this path many years ago—many thanks to Mary for encouraging my return to University.

Acknowledgements

University can be challenging, but it is especially challenging thirty years after your undergraduate studies, not to mention while working full time.

Research partners are essential, and our courses allowed great opportunities for collaborative teamwork. Nothing beats the excitement of working with different perspectives from amazing and brilliant grad students. My main collaborator was Jordan Willis, who was with me from Literature Review (and trials with L^AT_EX), Grid computing, but especially working on Connected Vehicle research projects. Jordan inducted me into python programming and considering other current trends like software defined radio. His keen insight and broad interesting perspectives helped my knowledge flourish. To my research partner and now great friend, Jordan, please accept my sincerest heartfelt "thank you very much". I aspire to your level of prowess in python.

I also would like to kindly thank my thesis readers: Dr. Bill Anderson from the Cross-Border Institute, and Dr. Robert Kent from Computer Science. Bill, it was highly interesting to learn about the issues, challenges and concerns you have had to deal with in projects for the Cross-Border Institute. Your guidance on reaching the right people at the research department (ORIS) was invaluable. I would not have expected collaboration paperwork with US DOT would take many months to complete, and my data gathering would not have been as relevant without this. Your insight was very relevant and timely.

Bob, you and Subir are the only remaining faculty from my undergrad days. It has been a pleasure to learn from you and a privilege to know you. Thanks very much for your encouragement and helping me hone my skills.

My master's advisor, Dr. Arunita Jaekel, has been exceptionally helpful and has reinvigorated my critical thinking and my research. Arunita, you have inspired me to think more creatively and channelled my efforts to be better focussed. I really appreciate the opportunity to explore developments with connected vehicles. Research into the Intelligent Transportation System is leading to many interesting and exciting opportunities.

And finally, a heartfelt thanks also to Herald Press, specifically my wonderful brother-in-law, Joe Aquilina, for professionally printing the bound version of this document.

Contents

Author's Declaration of Originality	iii
Abstract	iv
Dedication	v
Acknowledgements	vi
List of Tables	xi
List of Figures	xii
List of Appendices	xiv
Terms of Reference	xv
Introduction	1
1.1 The Scope Of The Thesis	1
1.2 Importance of this Research Topic	1
1.3 Problem Statement	2
1.4 Solution Outline	4
1.5 The Structure Of The Thesis	5

Background Information	6
2.1 Connected Vehicles	6
2.1.1 Why are Connected Vehicles Important?	6
2.1.2 DSRC/WAVE	7
2.1.3 Frequency Allocation	9
2.2 Urban Canyon	10
2.2.1 What is the Urban Canyon?	10
2.3 Satellite Information	11
2.3.1 Why use Satellites?	11
2.3.2 Satellite Frequency Allocation	11
2.3.3 Position Determination	13
2.4 Satellite Signal Anomalies	15
2.5 Position Algorithms	16
2.6 Linear Quadratic Estimation	18
2.6.1 What is a Kalman Filter?	18
2.6.2 Kalman Optimal Estimates	19
2.7 Road Grade or Slope	20
2.7.1 Elevations	21
2.8 Literature Review	22
2.8.1 Annotation - <i>Townsend 1995</i>	22
2.8.2 Annotation - <i>Psiaki 2001</i>	23
2.8.3 Annotation - <i>Groves 2005</i>	25
2.8.4 Annotation - <i>Meguro 2009</i>	27

2.8.5	Annotation - <i>Khodjaev 2010</i>	28
Identifying & Mitigating GPS Anomalies		30
3.1	Research Methodology Overview	31
3.2	Work Flow Diagram	32
3.3	Steps Completed	32
3.4	Data Collection	35
3.4.1	GPS Data Collection	35
3.4.2	OBU Data Collection	36
3.5	Data Analysis	37
3.5.1	Questions on Data from the Urban Canyon	37
3.5.2	Statistical Analysis of OBU Data	37
3.5.3	DSRC Details for On-Board Unit Data	39
3.6	Visualization	40
3.7	More Kalman Filter Information	42
3.7.1	Kalman Prediction Correction Cycle	42
3.7.2	Kalman State Vectors	43
3.7.3	Kalman Filter Explanation	43
3.8	Evaluate and Refine	44
3.9	Urban Tunnels	45
3.10	GPS Position Discard Threshold Method	46
3.10.1	GPS Position Discard Threshold – Pseudocode	48
Experiment Results		51

4.1	Steps Completed	51
4.2	Method	54
4.3	Charts and Graphs	55
4.4	Position Maps	58
4.4.1	Arada Raw OBU Data	59
4.4.2	Arada OBU Data - Standard Kalman Filter Smoother	60
4.4.3	Arada OBU Data - Extended Kalman Filter	62
4.5	Position Discard Threshold Method Results	63
Conclusions and Future Work		65
5.1	Conclusions	66
5.2	Possible Areas for Future Work	67
Bibliography		69
Appendices		74
	Python Kalman Smoothing	75
	Weka Random Forest	84
	Arada OBU Data Fields	86
	Colophon	87
Vita Auctoris		88

List of Tables

2.1	WHO 2013 Global Status Report on Road Safety	6
2.2	World DSRC Frequency Allocations	8
2.3	US DSRC Frequency Allocation	9
2.4	Satellite frequency overview	12
2.5	Global Navigation Satellite Systems	12
2.6	GPS Position and Elevation References	21
3.7	Position / Elevation Confidence	40
4.8	Epoch Cross-Reference for GPS Map Position Figures	56
C.9	Arada OBU Data Fields	86

List of Figures

1.1	Satellite signal paths	4
1.2	Choke Ring Dual Polarization Antenna	4
2.1	US DSRC protocol stack layered communication architecture	7
2.2	US DSRC Channel Assignments	9
2.3	Detroit DSRC Testbed Urban Canyon photos	10
2.4	Triangulation	13
2.5	Trilateration in 2-D	14
2.6	Trilateration in 3-D	14
2.7	Road Grade	20
2.8	GPS Bit Grabber	24
3.1	Workflow Diagram	32
3.2	Kalman Filter-Smoother Prediction-Correction Cycle	42
4.1	ITS 2014 Detroit DSRC Urban Canyon Test Track	51
4.2	Test Run 1 - True Track	52
4.3	Test Run 1 - GPS Track showing Urban Canyon Issue	52
4.4	Test Run 2 - True Track	52
4.5	Test Run 2 - GPS Track showing Urban Canyon Issue	53

4.6	ARADA OBU Velocity Comparison CAN Bus vs. GPS	54
4.7	DSRC GPS Position Analysis	56
4.8	Arada OBU Raw Nav Data	59
4.9	Arada Nav Data - Standard Kalman Smoother	60
4.10	Standard KF after smoothing (including IMU velocity & acceleration)	61
4.11	Extended KF after smoothing (using yaw ψ, velocity & yaw rate $\dot{\psi}$)	62
4.12	Test Drive results after Position Discard Threshold method	63

List of Appendices

Appendix A. Python Kalman Smoothing	75
Appendix B. Weka Random Forest	84
Appendix C. Arada OBU Data Fields	86
Appendix D. Colophon	87

Terms of Reference

ADAS	Advanced Driver Assistance Systems
ADS-B	Automatic dependent surveillance-broadcast augmented aircraft information system
AES	Advanced Encryption Standard
APNT	Alternative Positioning, Navigation, and Timing FAA system to handle widespread disruption of GNSS
ASIC	Application-specific integrated circuit
Beidou	Chinese GNSS system (北斗 Big Dipper)
BSS	Basic Service Set (DSRC message)
C/A code	L1 Course/Acquisition code (in GNSS)
CALM	Communication Access for Land Mobiles
CAMP	Crash Avoidance Metrics Program (1995 Ford & GM partnership with VIIC)
CAN Bus	Controller Area Network bus
Car2X	Car to Infrastructure
CDMA	Carrier Division Multiplexing
CEN	Comité Européen de Normalisation European Committee for Standardization
CRPA	Controlled Reception Pattern Antenna
DoS	Denial of Service
DGPS	Differential GPS (using a known nearby fixed position ground based receiver)
DSRC	Dedicated Short Range Communication (IEEE 802.11p)

ECC	Electronic Communications Committee (Europe)
- or -	Elliptic Curve Cryptography
ECDSA	Elliptic Curve Digital Signature Algorithm
ECU	Electronic Control Unit
EDA	Electronic Design Automation
Euclidean	Euclidean space
EGNOS	European Geostationary Navigation Overlay Service European version of WAAS to improve GNSS accuracy
EIRP	Equivalent (<i>Effective</i>) isotropically radiated power
ETSI	European Telecommunications Standards Institute
EVITA	E-safety vehicle intrusion protected applications
Exterior	Consumer Reports
Dimensions	Dimensions:Exterior & Cargo Comparison
<hr/>	
FCC	Federal Communications Commission
FPGA	Field Programmable Gate Array
<hr/>	
Galileo	Global Navigational Satellite System (EU & ESA)
GBAS	Ground Based Augmentation System
GIS	Geographical Information System
GLONASS	<i>Globalnaya navigatsionnaya sputnikovaya sistema</i> (Russia)
GNSS	Global Navigational Satellite System [1]
GPRS	General Packet Radio Service (over GSM)
GPS	US GNSS (Global Positioning System)
GSA	European Global Nav Sat Systems Agency
GSM	Groupe Spécial Mobile (now Global System for Mobile) ETSI Std for 2G cellular data communications max power 2W in 850/900, 1W in 1800/1900
GSV	GPS Satellites in View
<hr/>	
HMI	Human Machine Interface
HSM	Hardware Security Module
HSPA	High Speed Packet Access

ICI	Intelligent Car Initiative EC policy framework for connected vehicles
IEEE 802.11p	Wireless Access in Vehicular Environment
IMU	Inertial Measurement Unit
International Standards	IEEE 1609 for U.S. operation at 5.9 GHz ETSI TC-ITS for European use at 5.9 GHz ARIB STD-T109 Japanese standard at 760 MHz
ITS	Intelligent Transportation System [2]
ITU	International Telecommunication Union UN agency sets frequency allocation standards
<hr/>	
KF	Kalman Filter
<hr/>	
LAAS	Local Area Augmentation System Local ground reference to improve GPS accuracy
LBS	Location-Based Service
LOS	Line of Sight (Satellites in View)
LWD	Low Water Datum
<hr/>	
MANET	Mobile <i>Ad hoc</i> Network
MSAS	Multi-functional Satellite Augmentation System Japanese system to improve GPS accuracy
<hr/>	
NLOS	Non-line of Sight
NMEA	National Marine Electronics Association
NTIA	National Telecommunication and Information Administration control spectrum allocation in US
NoW	Network on Wheels (Europe)

OBD	On-board Diagnostics
OBU	On Board Unit
PNT	Positioning, Navigation and Timing
PRN	Pseudo-Random Noise practical use is satellite ID number
PVT	Position, Velocity, Time, used in satellite measurements
P/Y	L2 Precision Encrypted GPS signal (in GNSS)
RTTT	Road Transport and Traffic Telematics <i>European standards</i>
RSA	Rivest, Shamir, Adleman (public key encryption)
RSSI	received signal strength indicator
RSU	Road Side Unit
SEVECOM	Secure Vehicular Communication (Europe)
SHRP 2	Second Strategic Highway Research Program United States National Research Council Transportation Safety Board
SLAM	Simultaneous localization and mapping [3]
SPaT	Signal Phasing and Timing
TEC	Total Electron Count (solar flares increase ionospheric scintillation)
TESLA	Timed Efficient Stream Loss-tolerant Authentication RFC 4082 RFC 4383 RFC 4442
TAI	Temps atomique international International Atomic Time
TCXO	Temperature Compensated Crystal Oscillators
TOA	Time of Arrival (helps define receiver distance from satellite)
TOW	Time of Week aka Z-count (satellite signal transmission)
TPEG	Transport Protocol Experts Group
TPM	Trusted Platform Module

UC	Urban Canyon
UMTRI	University of Michigan Transportation Research Institute
UTM	Universal Transverse Mercator coordinate system
UWB	Ultra Wide Band FCC defines as any signal which has an absolute bandwidth larger than 500 MHz

V2I	Vehicle to infrastructure
V2V	Vehicle to vehicle
VANET	Vehicular <i>Ad hoc</i> Network
VHDL	VHSIC Hardware Description Language
VHSIC	Very High Speed Integrated Circuit
VII	Vehicle Infrastructure Integration (US DOT) which became Intellidrive
VIIC	Vehicle Infrastructure Integration Consortium Ford, BMW, Chrysler, GM, Honda, Daimler, Nissan, Toyota and VW-Audi

WAVE	Wireless Access in Vehicular Environments (IEEE 802.11p)
WAAS	Wide Area Augmentation System monitors GPS constellation to certify integrity
WBSS	WAVE basic service set
WGS-84	World Geodetic System basis for GPS
Wi-Fi	Wi-Fi Alliance of products certified to work within IEEE 802.11
WiGLE	WiFi Geographic Logging Engine Wireless hotspot registry
WSIE	WAVE service information element
WSMP	WAVE Short Message Protocol

Introduction

1.1 The Scope Of The Thesis

The Intelligent Transportation System (ITS) [2] attempts to solve the issue of public safety by warning drivers of impending collisions, providing sufficient time for accident avoidance. Notification of impending danger will prevent many accidents, injuries and fatalities. US DOT (Department of Transport) [4] researchers preliminary statistics show that up to 70% of these non-impaired accidents can be prevented. High accuracy position references are required for this system to function. Global Positioning System (GPS) signals are available and highly accurate, although signal problems can cause errors [5–13]. Line of sight (LoS) view from several satellites to the receiver is essential. Urban areas with roads adjacent to very tall buildings suffer from poor LoS required by GPS receivers. A connected vehicle testbed was built for the ITS World Congress 2014 in Detroit, using ARADA Road Side Units (RSU) [14]. We have arranged with US DOT and the University of Windsor Office of Research (ORIS) to collaborate in research, and obtained access to this US DOT testbed. (The agreement is for a term of three years and concludes in April 2018.) Using ITS data collected by DSRC on-board units (OBU's) on the Detroit testbed, I will present a proposal to demonstrate a method to improve accuracy of individual vehicle position information.

1.2 Importance of this Research Topic

Many researchers have identified the so called ‘Urban Canyon’ problem. [5, 6, 10–12, 15–18] The issues are not trivial to solve, with reflected signals causing many calculated positions to appear as if we are driving through buildings. Antenna systems for vehicles

are now fairly standardized, and they are compact enough to not be obtrusive, yet sufficient to provide adequate real world performance. The signals from satellite to receiver can follow three distinct pathways; 1) direct ‘Line of Sight’ (LoS) and 2) indirect (meaning non-LoS) or 3) both LoS and non-LoS at the same time, causing the so called ‘Multi-Path’ problem — which leads to wild position inaccuracy. [5, 6, 17]

Antennas have been designed that are capable of reducing errant signals by an order of magnitude. [11] Connected Vehicles (CV) have tiny (typically 35mm square by less than 12mm thick) antennas, positioned for good sky view by the rear view mirror. Choke plate and beam forming antennas are typically 1m in size and therefore unsuitable for CV applications.

Eminent researchers working with Global Navigational Satellite Systems (GNSS) have designed elaborate strategies [10, 11, 18] to combat these issues with expensive specialized equipment. For example, a ‘3-D shadow mapped database’ of the building terrain in a specific area of London, UK, was demonstrated to assist in determining which satellites to ‘score lower’ in probability, which has been shown to significantly improve cross street position accuracy. Android smartphones were discussed at the ITS 2014 World Congress, most highly considered by the EU community, but not considered ‘mainstream’ in North America or Asia—mostly due to higher adoption of other personal mobile devices outside of Europe. It is not realistic to me that we will see elaborate 3-D databases on connected vehicles in the near term.

1.3 Problem Statement

Satellite receivers expect clear line of sight to accurately determine GPS position. These receivers are intended to be light weight and low power units, which works well in most use cases. GNSS systems are designed to provide orbital (ephemeris) details and have the (relatively) less complex timing signal processing done by compact receivers. Ephemeris is an historic term used to describe a table of orbital data. Mariners typically carried and used these tables to determine their current position. GPS techniques still rely on this type of data, and electronically provide updated ephemeris details (including clock correction)

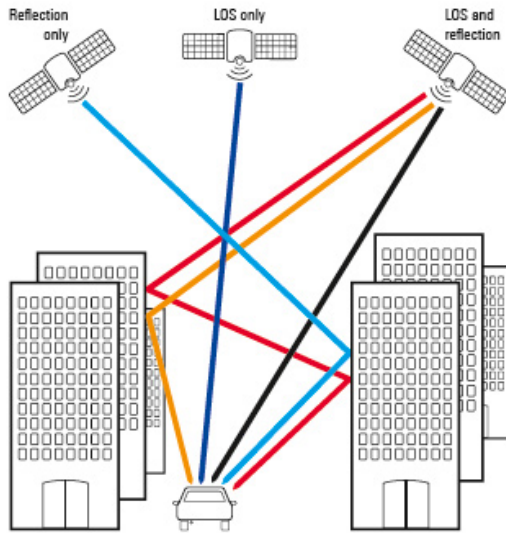
dynamically through their carrier signals. Aircraft and most open water marine uses do not encounter satellite obstructed views. Only in natural or urban canyons do we lose sight of many satellites, and therefore notice these wild position reporting issues.

Vehicles leaving any parking structure or tunnel will take some time to acquire satellite signals, typically five to twenty seconds, sometimes significantly longer. GNSS receivers calculate their position based on the known ephemeris data, and the timing signals received. Ephemeris data must be interpreted and loaded into the receiver for all satellites in range. Only once the receiver knows where the satellites are can it then determine where it's own position is relative to these known satellite positions. Potentially up to 13 satellite orbital paths and timing signals need to be processed, and signals interpreted. These signals can commonly take well over a minute to initialize the receiver before position can be determined.

A minimum of four satellites is required to solve the geometry for location. These non-trivial calculations are complex enough that significant human time (several seconds) can elapse before a position fix is reported. In a steady state of motion, position can be predicted well enough, but acceleration and deceleration pose difficulties and significant lag times occur due to solving these complex equations. Various physical system and environmental errors also can occur which will alter transmission of satellite signals. [19]

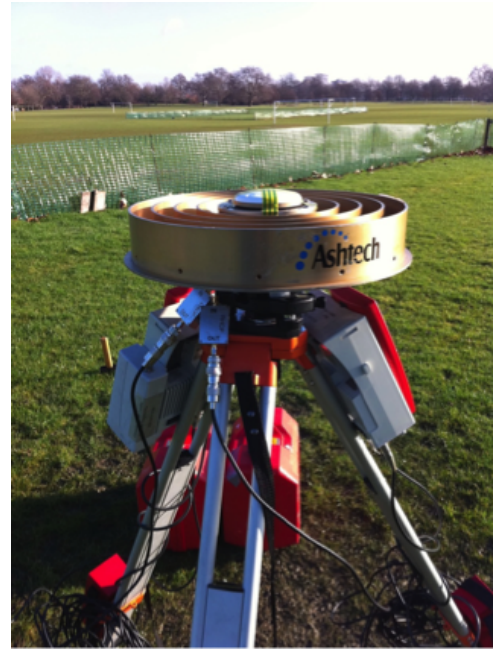
My work is not directly concerned with these environmental issues, rather I've focused on the following specific signal path issues. Three situations are considered for satellite to receiver geometry: i) direct line of sight, ii) non-LoS bounced signals which extend the apparent distance from a satellite, and iii) receiving both the direct and reflected signals, known colloquially as the 'Multi-path' problem, greatly throwing off the ability to accurately make a valid position calculation.

Figure 1.1: Satellite signal paths



Bounced non-LoS signals shown in orange/red
Direct LoS signal shown in black
diagram from <http://www.rohde-schwarz.fr>

Figure 1.2: Choke Ring Dual Polarization Antenna



http://www.insight-gnss.org/WPs_figures/ChockRingDualPolAnt.png (image source)

1.4 Solution Outline

Anomalous position reports will render collision avoidance systems useless. After a preliminary review of the published literature, it became apparent that a method to provide on-board calculation of potential GPS position error was required. Vehicles will neither have a large 3-D shadow mapped database on-board in the near term, nor on-board physics labs. A data-driven solution is required, making use of the available OBU processors and handling error detection and correction.

Kalman-Busy filters were adopted by the navigation community in the early 1960's, when Rudy Kálmán presented the technique to engineers at NASA Ames Research Center who were working to solve telemetry and control system gaps. [20] Noisy transmissions and particularly transmission gaps were causing them trouble reading telemetry and providing control data. Kalman's filter technique was successfully used to help with Apollo missions. Later application to RADAR systems and other guidance and control

systems has left the Kalman filter as the leading choice for navigation system calculations. [3,21–26]

Our intent is to design a solution to be implemented programatically, as a proof of concept, for application on current vehicles working with current DSRC equipment on-board. Preliminary data examination will be done using a commercial Garmin nüvi 2798 GPS, with data gathered while driving on the Detroit DSRC testbed. This was done to demonstrate the issue and provide insight into mapping the results. True DSRC testbed data was also gathered, with recorded results from ARADA On-Board Units (OBU's) [42] and including the integrated CAN Bus sensors, inertial measurement unit (IMU) and GPS position results. The majority of the research centred on this DSRC testbed data. The main proposal is a method for detection of the anomalous position, creating thresholds to interpret errors, and also exploring use of a Kalman-Busy filter in python as proof of concept for a path smoothing solution implementation.

1.5 The Structure Of The Thesis

For this thesis §1 covers the *Introduction*, §2 discusses *Background Information* with features of *Connected Vehicles*, *Urban Canyons* and *Satellite Navigation* as well as a *Literature Review*, §3 discusses my Research Methodology for *Identifying & Mitigating GPS Anomalies* including *Data Acquisition* and *Data Analysis*. §4 provides *Experiment Results* and insight into my *Proposed Solutions*. Closing remarks in §5 offer *Conclusions* including areas for *Future Work*. Each chapter will close with four diamonds on the far right. ❖

Background Information

2.1 Connected Vehicles

2.1.1 Why are Connected Vehicles Important?

World Health Organization (WHO) reports statistics on automobile related deaths. Their "GLOBAL REPORT ON TRAFFIC SAFETY" [27] numbers are astounding—1.24 Million deaths per year, or nearly 3400 deaths daily. This is a staggering number, and includes drivers, occupants and pedestrians. Table 2.1 shows annual Road Safety statistics for Canada, and the countries developing GNSS systems.

Table 2.1: WHO 2013 Global Status Report on Road Safety

Country	Population	Road Fatalities	Death Rate per 100,000 pop.
Canada	34 016 594	2 296	6.8
US	310 383 968	35 490	11.4
Russia	142 958 156	26 567	18.6
China	1 348 932 032	275 983	20.5
India	1 224 614 272	231 027	18.9
Europe	496 465 562	36 144	7.3

European totals include: Belgium, Czech Rep., France, Germany, Greece, Hungary, Italy, Netherlands, Poland, Portugal, Romania, Spain, Switzerland, United Kingdom, Ukraine [27]

Developing a method to reduce this very large number of fatalities would be a great benefit to society. The loss of life, and the economic costs are measured by country in percentage of each countries GDP. Developing a technical solution for reducing fatalities remains a principal motivation of the Connected Vehicle initiatives. IEEE have developed specifications and standards to allow vehicles to communicate using a modified version of WiFi. IEEE 802.11p standards outline "Wireless Access in a Vehicular Environment".

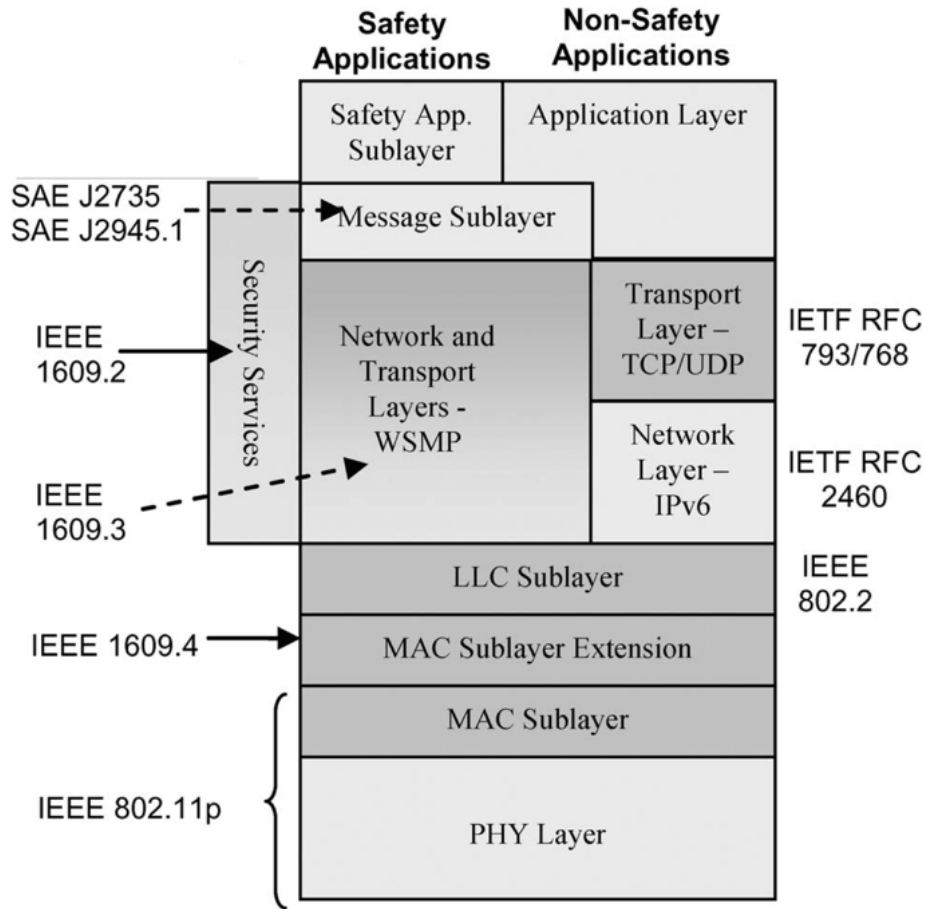


Figure 2.1: US DSRC protocol stack layered communication architecture

from "Dedicated Short-Range Communications (DSRC) Standards in the United States", John Kenney, 2011 [28]

2.1.2 DSRC/WAVE

"Dedicated Short-Range Communication" (DSRC) [28,29] was developed at the same time in North America and Europe. The FCC allocation of 75MHz of bandwidth in the 5.9 Ghz range to allow safety communications for the Intelligent Transportation System was formalized in the 2004 "Report and Order FCC-03-324" for the US. European rules were deployed in 2009 through CEN and ETSI. These rules define the frequency space for ITS operations. DSRC is specifically the term used to describe the technology for various Connected Vehicle communications, including the mandated 5.9 Ghz frequency band.

“Wireless Access in a Vehicular Environment” (WAVE) **802.11p** is an amendment to the IEEE 802.11 specification standard. Traditional 802.11 wireless connections required involved relatively slow handshakes and negotiation—this is not possible in ITS applications due to the transitory nature of the connections. Vehicle speed and changing positions of nodes in the Vehicular Ad-Hoc Network (VANET) do not allow stable connections to be established for a long period of time. WAVE was envisioned as ‘connectionless’, and there are extended standards in IEEE 1609 for security (.2), Network and Sub-Layers WSMP (.3), MAC Sublayer Extension (.4)

It was interesting to discover at the 2014 ITS World Congress that standards around the world for ITS vary greatly. Spectrum allocation for WAVE services in Japan are only available in the 700MHz range, due to previous use of the 5.9 GHz space for toll collection. 15 Million vehicles in Japan use this toll collection, it would now be nearly impossible to transition these vehicles into the 5.9 GHz band. While the Europeans standards (through ETSI) are using 5.9 GHz, they have other incumbents in their frequency space and are only able to offer three available channels for DSRC/WAVE. Several independent European standards groups have convened regarding different facets of WAVE communications. These groups are not in agreement with each other, and that fact makes implementation a bit more interesting. In North America we have seven channels available for DSRC/WAVE—yet the BSM (basic safety message transmissions) are all sharing one common channel (channel 172).

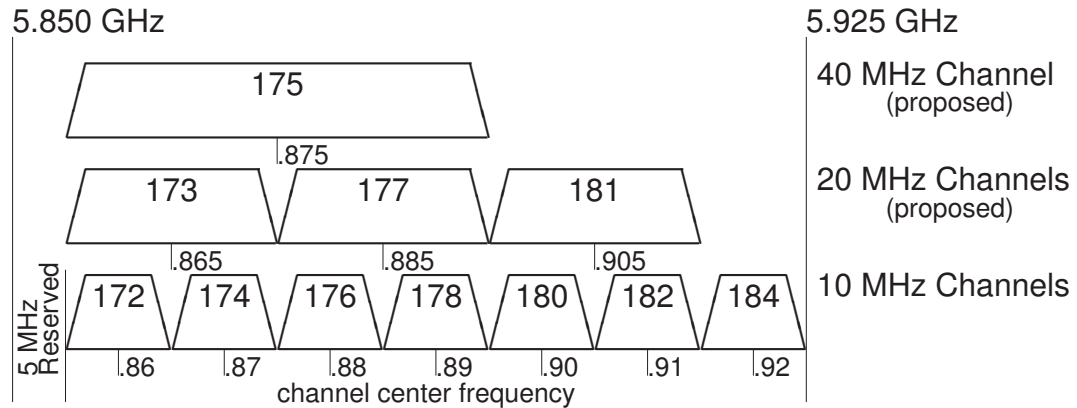
Table 2.2: **World DSRC Frequency Allocations**

Region	Start Frequency	Channel Width	End Frequency	Channels Available
USA	5.850 GHz	10 MHz	5.925 GHz	7
	Channels 172,174,176,178,180,182,184			
EU	5.795 GHz	10 Mhz	5.815 GHz	3
	5.795-5.805			
Japan	700 MHz	10 Mhz		x

2.1.3 Frequency Allocation

US DSRC Frequency Allocations have been established using channel numbers as follows:

Figure 2.2: US DSRC Channel Assignments



The proposed one 40 MHz and two 20 MHz channels are actually created by combining the underlying channels, i.e. Channel 175 uses the full bandwidth of four 10MHz channels 172 through 178. Similarly Channels 173,177, and 181 combine the smaller channels.

Table 2.3: US DSRC Frequency Allocation

Frequency Center	Channel	Application	Start	End
5.8525	170	Reserved 5 MHz	5.850	5.855
5.860	172	Collision Avoidance Critical Safety of Life	5.855	5.865
5.870	174	Service	5.865	5.875
5.880	176	Service	5.875	5.885
5.890	178	Control Channel & Service Announcement	5.885	5.895
5.900	180	Service	5.895	5.905
5.910	182	Service	5.905	5.915
5.920	184	High Power Public Safety	5.915	5.925
Proposed three 20MHz Channels 171,177,181				
5.865	173	Control Channel	5.865	5.885
5.885	177	Control Channel	5.885	5.905
5.905	181	Service	5.905	5.925
Proposed one 40MHz Channel 175				
5.865	175	Control Channel	5.865	5.905

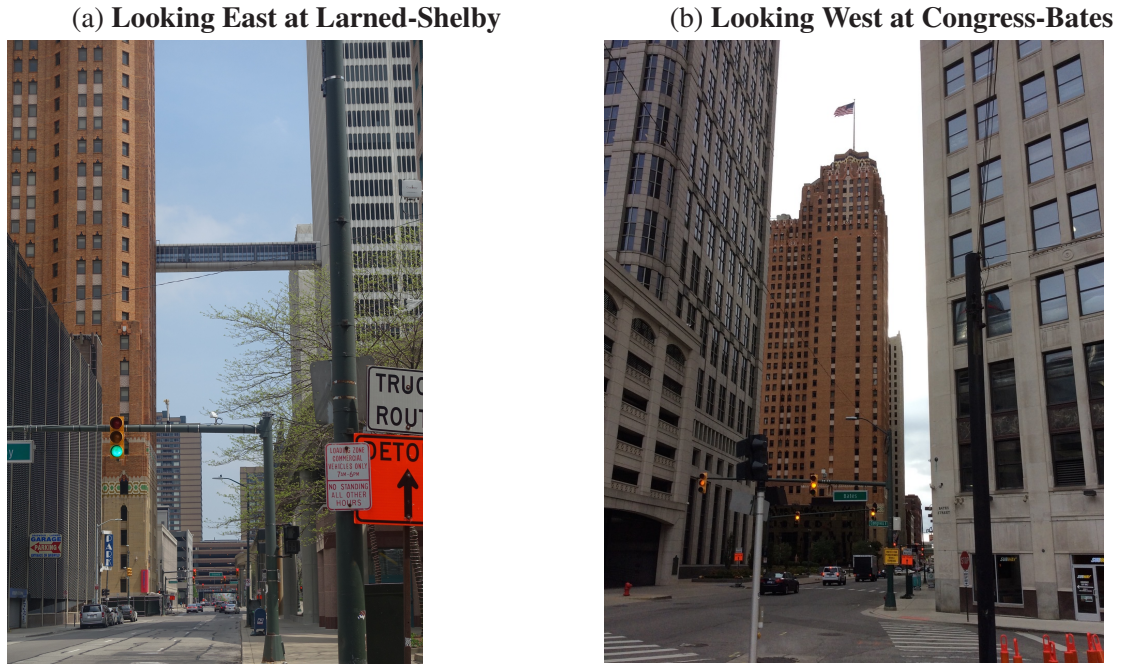


Figure 2.3: **Detroit DSRC Urban Canyon** photos taken by the author (October 18, 2014)

2.2 Urban Canyon

2.2.1 What is the Urban Canyon?

Blocked or mis-directed GPS signals can easily be disrupted, leading to inaccurately calculated position. Tall buildings line most urban streets, creating limited sight lines to overhead sky, and effectively blocking a majority of satellite signals from direct LoS to a street level receiver. Areas that are blocked are considered to be in GNSS shadow [10,12]. Signals coming along the direction of travel are fine—yet signals along the cross-streets are required for accuracy and only available in short bursts, which further complicates receiver processing.

The “Urban Canyon” has become a well known problem in GNSS research [10, 12, 15, 17, 30, 31]. Non-line of sight (nLoS) signals are only part of the problem. Some signals may be received both directly (LoS) and indirectly (nLoS) at almost the same time—which is the so-called ‘Multi-Path’ problem. These multipath signals give the receiver further conflicting data.

2.3 Satellite Information

2.3.1 Why use Satellites?

Ground based navigational references run into many problems. An observer looking out at 1.8m above sea-level can only detect features within roughly 11 km. This can be much reduced due to atmospheric conditions. Line of sight is better when taller landmarks are available yet the earth's curvature is a significant barrier to maritime navigation. Loran and other radio based systems were devised, and still cover much of the coastal areas. Other signal propagation issues that made these great for marine operations made them sluggish and inconsistent for aviation use. Inertial navigation systems are still widely used today yet they remain complex and expensive. Military operations demanded higher precision and led to the always visible Global Navigational Satellite System (GNSS) [1] concept.

2.3.2 Satellite Frequency Allocation

The United Nations advocates co-operation between member countries on many matters. Their initiative to align radio frequency use is the International Telecommunication Union (ITU). World Radiocommunications Conferences (WRC) are convened regularly every three to four years by the ITU—most recently during November 2015 in Geneva, Switzerland. Inside each country allocations are controlled by their respective government agencies. Industry Canada allocates frequencies here and for the USA it's their National Telecommunication and Information Administration (NTIA). The general public are more familiar with the regulatory agencies. Regulations and enforcement are undertaken by the CRTC in Canada, and the FCC in the USA.

A Satellite frequency overview is detailed in [Table 2.4](#) below, showing current in use GNSS frequencies. [Table 2.5](#) provides an overview of various GNSS systems in operation. GNSS data has been compiled from many sources.

Table 2.4: **Satellite frequency overview**

L1	1575.42 MHz	Course-Acquisition (C/A) and encrypted precision (P(Y)) codes, with L1 civilian (L1C) and military (M) codes on future Block III satellites. P(Y) code, with L2C and military codes on the Block IIR-M and newer satellites. Nuclear detonation (NUDET) detection. Used to study ionospheric correction. Proposed for civilian safety-of-life (SoL) signals.
L2	1227.60 MHz	
L3	1381.05 MHz	
L4	1379.913 MHz	
L5	1176.45 MHz	

source: [Wikipedia - GPS](#)Table 2.5: **Global Navigation Satellite Systems**

Launch FullOp	Coun- try	Project	in Orbit	Region Covered	Frequencies
1978- 1993	USA	GPS (Global Positioning Satellite)	27	global	L1 - 1575.42 MHz L2 - 1227.60 MHz
1982	Russia	GLONASS (<i>Globalnaya navigatsionnaya sputnikovaya sistema</i>)	24	global	FDMA L1 $1602 + n \times 0.5625$ MHz L2 $1246 + n \times 0.4375$ MHz $n = -7$ through $+6$ CDMA 1600.995 MHz 1248.06 MHz 1202.25 MHz Interoperable CDMA (2025 planned) 1575.42 MHz 1207.14 MHz 1176.45 MHz
2002	China	Beidou-1 (Big Dipper)	12	Asia — Australia	
2011 2015	EU	Galileo	8	Europe	
2007	China	COMPASS or BeiDou-2	35	global	
2013	India	IRNSS	7	India Region	L5 (1176.45 MHz) S band (2492.028 MHz)

GNSS satellite information compiled from numerous sources

2.3.3 Position Determination

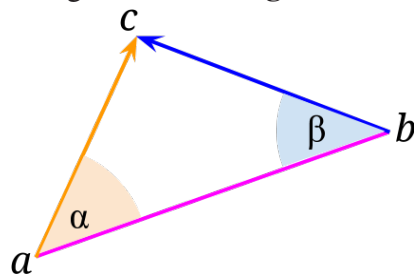
Navigation on land or sea requires that a magnetic bearing is taken (a compass angle from a reference point) usually from the navigators present position to whatever intermediate ‘waypoint’ they are intending to end up reaching. This bearing is known as the ‘heading’ and will represent the direction of travel. Orienteering practise is that these short navigation ‘legs’ are used and that direction will fluidly change as terrain requires, with the goal of reaching the next waypoint. The objective should be to change the general intermediate ‘heading’ to reach the next waypoint. Determining a new ‘heading’ either at the next waypoint or through pre-planning will allow the navigator to reach their destination.

Triangulation

In a standard euclidean 2-dimensional space, when given three non-collinear points, a triangle can be constructed using the points as vertices. By definition the sum of interior angles will be 180° and standard trigonometry will apply. The known distance ab and known angle α allow a quick calculation of $\sin(\alpha)$, which is the distance perpendicularly from the *line* ab to *point* c .

And so given two known positions, with exact distances between these, and the reference angles from these to a third point, you can determine the exact location using trigonometry (literally “triangle measure”). (See Triangulation [Figure 2.4](#)). Determining a map position of any object using a two dimensional map can be accomplished through this concept of triangulation.

Figure 2.4: **Triangulation**

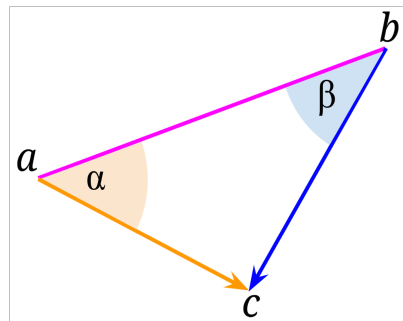


Triangulation ([Figure 2.4](#)): given these two known points a and b combined with known angles α and β referenced from the line ab to our unknown point c , we can geometrically complete the triangle. This allows precise determination of an intersection at the third (remote) point c .

Trilateration

In 2-dimensions, measuring a bearing from two reference points (for instance on a lakeshore) allows a mariner to plot two lines on a map, where the intersection of those lines will pinpoint their present position. This can be especially useful when known underwater features exist that might present challenges to navigation. The key difference is that trilateration is solving for our position as the unknown, rather than the destination.

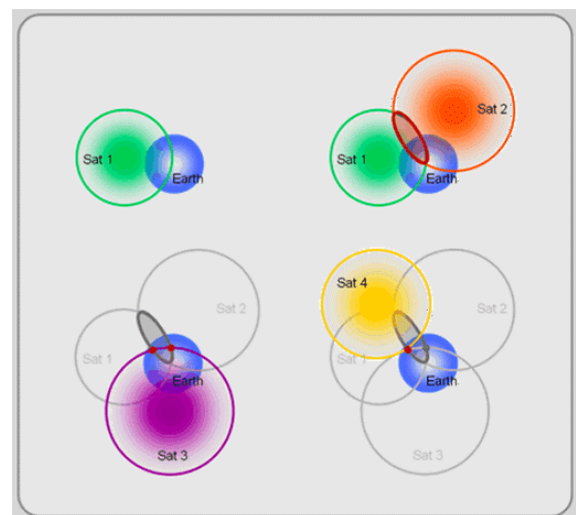
Figure 2.5: **Trilateration in 2-D**



Trilateration in 2-D (Figure 2.5): given two known points a and b at a distance combined with known angles α and β referenced from the line ab to our unknown point c , we can geometrically complete the triangle to allow precise determination of our location c from the intersection of the lines ac and bc .

Adding a third dimension changes this slightly. Trilateration in 3-D appears a bit puzzling at first. Looking at Figure 2.6 this explanation follows. The intersection set of two satellite signals form a circle (upper right, circle shown in dark red). A third signal will add another circle, which intersects the first creating two probable positions (lower left depicted in magenta). A fourth satellite will further refine the intersection down to a precise location (lower right, yellow circle meets at magenta point.) [19].

Figure 2.6: **Trilateration in 3-D**



Trilateration in 3-D (Figure 2.6): three dimensional signal intersections narrow a determined position. Fourth satellite pinpoints estimated position.

from <http://giscommons.org/files/2010/01/2.141.gif>

In practise, these satellites are all moving, and the critical factor becomes the time that each signal was received. In fact with the possibility that all the satellites have timing errors the complexity can grow very quickly. GNSS satellites all contain at least a pair of highly accurate atomic clocks, which are referenced to a ground based network of cesium atomic clocks, and synchronized as required. These atomic clock sources are very expensive, published estimates above US\$100,000 each. Ground based receivers have inexpensive clocks, which effectively forces them to adjust for their time inaccuracy by requiring a fourth satellite signal.

2.4 Satellite Signal Anomalies

Satellite signals are transmitted at very low power (typically 25W [32]), and run into many interesting problems. Obvious limitations exist but the overall system is very sound, and would not have been implemented in such a widespread manner if it were totally unreliable. The challenges are most broadly considered as three major categories, namely clock errors, atmospheric errors and orbital errors. Each of these will be discussed in more detail below. Much of this data has been posted online by the various university departments that are studying these phenomena. [13, 33, 34]

Primary sources for GPS error:

- **Clock Errors**

Satellite's use highly accurate atomic clocks, yet even these do have time drift and require monitoring and resetting through the command centre. Receiver clocks are far less accurate. "*Where you are*" is highly dependant on "*what time it is*", with $speed\ of\ light \times time = distance$ to the satellite. Even a one microsecond drift of the satellite clock can be very significant in determining GNSS receiver position. Where this gets interesting is taking $c \times 1\mu s = distance$ representing a one microsecond timing error, giving $3 \times 10^8 m/s \times 10^{-6} s = 3 \times 10^2 m = 300m$ distance. By extension $300m \equiv$ one microsecond and $0.3m = 30cm \equiv$ 1 nanosecond or 10^{-9} seconds. In other words a single nanosecond error can alter posi-

tion by 30cm. Satellite clock accuracy is kept within range of a few nanoseconds (10^{-9} seconds). I recently discovered that the satellite clock updates are generated through a Kalman Filter.

- **Lower Atmosphere – up to 50km above the surface**

The Troposphere, Tropopause, and Stratosphere layers delay signal transmission. Satellites lower in the sky to the observer are much more affected by these lower atmospheric effects since the distance that signal traverses through these layers is much greater. Some researchers refer to delays as signal propagation errors.

- **Upper Atmosphere – 50-1000km above the surface**

Ionospheric Scintillation [33, 35, 36] can create a blockage or deflection of satellite signals. It is caused by charged particles (ions) which dynamically alter signal refraction. Solar activity (wind, flares, sunspots, etc.) can send charged particles that affect these Ions. Other issues (space weather for instance meteor showers, time of day, geomagnetic activity, etc.) can also affect the ionosphere.

- **Orbital Drift**

Orbit shapes are eccentric rather than circular, and they are constantly drifting. Velocity is also variable. GPS control provides updates to these trends through the ephemeris data.

- **Multipath**

satellites below 15° in the sky can reflect signals downwards causing airborne multipath errors. It is suggested best practise to disregard these satellites in calculations to avoid this error. Antenna designs that favour signal polarization (such as choke rings) naturally deselect these lower in the sky satellites which can contribute to multipath errors. [11]

2.5 Position Algorithms

Improvements to GPS Positioning Algorithms are a large topic for technical papers. The current techniques appear to be based on orthogonal transformations and iterative least

squares methods, and I was surprised to learn that some use variants of Kalman filtering (KF) (typically extended or unscented, and some use of complementary KF). It is not clear to me that KF used with Differential GPS (DGPS) is an area of interest often discussed, where a known fixed ground station is used in conjunction with the moving receiver to understand any satellite position errors, and isolate those errors to dramatically enhance the mobile receiver results. This approach could be implemented into selected DSRC Road Side Units (RSU's) pretty effectively, and should not be ruled out as a potential solution to the Urban Canyon (UC) issue.

A more detailed table describing how the SAE J2735 standards for Connected Vehicle handle Position Confidence is included in Chapter 3 [Table 3.7](#) on page 40.

2.6 Linear Quadratic Estimation

2.6.1 What is a Kalman Filter?

Numerical Analysis examines concepts with algorithms and methods to handle estimation problems. Two newer areas of interest based on numerical analysis techniques are Sensor Fusion and Signal Processing. Sensor Fusion specifically expects inputs coming from various unrelated (sensor) sources and can be used to reduce error in estimating the information in these systems. Signal processing has a similar goal of feature extraction based on noisy input data. Both of these areas rely on Kalman Filters as one of the techniques used to extract information.

Control systems, navigation, guidance and signal processing data streams all contain noise and features that can be extracted. Rudy Kálmán developed a technique to filter the noise and predict the t_{n+1} state for the data. He presented this to NASA in 1960 where it became very helpful for the Apollo program, and eventually has become the de facto standard used throughout navigation, radar, guidance and control systems.

Kalman's seminal paper "A New Approach to Linear Filtering and Prediction Problems" [20] contained a real breakthrough in handling the error.

The general Kalman Filter (KF) concept is that a type of weighted average will be calculated at each time step and loaded into a covariance matrix. Effectively it's designed as an optimal least-square-error recursive predictor corrector algorithm [25]. The other key concept in Kalman is which do you trust more, the predictions or the measurements? This is handled by the gain—an idea that you should have the ability to tune the results to favour whichever of these better suits your needs.

The covariance initial state for position tracking are the classical newtonian equations. The measured state at any time t_{n+1} is processed with the covariance (weights) from the prior t_n step which should improve the predicted state. The Kalman Filter algorithm is designed to build the cumulative result weights into the covariance matrix, to remove the noise, and to allow the gain to favour predictions (low gain) or measurements (high gain) depending on the particular need.

On-board navigational systems relying on GPS and vehicle sensors will notice a significant lag in external data—which will be described later.

The Extended Kalman Filter (EKF) and unscented Kalman Filter (UKF) are non-linear versions of Kalman Filters. Extended KF uses a Taylor function to approximate the system, and the unscented transform is used to estimate the system states (mean and sigma distributions). These both generate linear function so that the KF premise of normal gaussian distributions will work, and therefore the EKF or UKF can generate a workable (non-optimal) estimate of the KF.

2.6.2 Kalman Optimal Estimates

In following Kalman's [20] paper three types of **Optimal Estimates** were discussed.

"To have a concrete description of the type of problems to be studied, consider the following situation. We are given signal $x_1(t)$ and noise $x_2(t)$. Only the sum $y(t) = x_1(t) + x_2(t)$ can be observed. Suppose we have observed and know exactly the values of $y(t_0), \dots, y(t)$. What can we infer from this knowledge in regard to the (unobservable) value of the signal at $t = t_1$, where t_1 may be less than, equal to, or greater than t ? If $t_1 < t$, this is a data-smoothing (interpolation) problem. If $t_1 = t$, this is called filtering. If $t_1 > t$, we have a prediction problem. Since our treatment will be general enough to include these and similar problems, we shall use hereafter the collective term estimation." [20]

A case is made that the Urban Canyon has specific large distribution errors such as "interference effects, namely signal cross-correlation, multipath and echo-only signals" by Salman Syed et al. [15] and the erratic nature of these errors make modelling difficult. Again Syed [15] discussed map matching, which runs the risk of 'picking' the incorrect road segment to arbitrarily map onto. Inertial Navigation systems typically experience drift, and require a stable position fix to reset for this drift. GPS experiences loss-of-signal (urban canyon, other atmospheric effects) and needs to be carried through these outages.

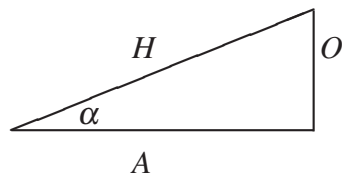
2.7 Road Grade or Slope

Road grade (or gradient) appears to be the conventionally used term to describe inclines, defined as “**Rise over run**” times 100%. In mathematical terms *rise / run* would be the $\tan \alpha$, representing the slope. A 45° incline would have a rise/run of 1/1, and therefore considered a 100% grade. This should become clearer below.

The steepest paved roadways have approximately 30% grade for short distances. Normal road grades are kept under 4% and normal highway maximum grade would be 8%. Mountainous roads like Pike’s Peak¹ have an average 6.7% grade and maximum grade of 10.5%. These translate into rise/run of 1/15 for 6.7% grade (3.8° incline) and 2/19 for 10.5%, a 6° incline.

Looking at an impossibly steep example road grade using the triangle below,

Figure 2.7: Road Grade



α represents the degree of slope,

$\tan \alpha$ is the $\frac{\text{rise}}{\text{run}} = \frac{3}{8}$

with opposite side (rise) 3, adjacent side (run) 8,

- giving $\tan(\alpha) = .375$, thought of generally as 37.5% grade (= 100% x rise/run)
- and $\angle \alpha = 20.6^\circ$

Using a familiar real world example, the Ambassador Bridge² structure has a 46m rise and half the 2286m span is 1143m. This means that the grade (gradient) is $100 \times (\frac{46}{1143})$ or 4% and solving for α we have the following:

$$\alpha = \arctan(\frac{46}{1143}) = \arctan 0.04 = 2.3^\circ$$

which is a 1 in 25 rise/run. This has the feel of being steep, and the Colorado highways that are 8% grade ($\approx 4.6^\circ$) also feel dramatically steep. Special consideration will be necessary for mountainous urban areas. San Francisco³ boasts 25 streets with 25%

¹https://parks.coloradosprings.gov/.../Pikes_Peak_Americas_Mountain/pphighway2009map.pdf

²<http://www.ambassadorbridge.com/intlcrossing/bridgefacts.aspx>

³<https://priceonomics.com/the-steepest-streets-in-/>

or steeper grade, Seattle⁴ has more than a dozen exceeding 20% grade. Urban canyon areas are generally not adjacent to these extreme grade roadways, and appear to have a better sky view than typical urban canyons. San Francisco has steeper grades than Seattle yet the later has more urban canyon problem areas. Extreme limits are in the range of 3 in 8, or a 37.5% grade (multiplying the grade by the linear ground distance covered will provide the correct limit). Altitude changes in hilly areas will be normally in the range of 1 in 4 rise/run, which is a 25% grade. Limits for typical reasonable altitude changes will be normally in the range of 1 in 12 rise/run, which is a 8.3% grade or $\approx 4.8^\circ$ incline.

2.7.1 Elevations

During review of GPS trails, it was noted that elevation information was inconsistent, as noted in Figure 4.7. Reference points can provide a context for the data. A few local points of interest are given in the table below:⁵

Table 2.6: GPS Position and Elevation References

Location	Latitude	Longitude	Elevation
Highest point in Detroit University District (west of Palmer Park)	42°25'37"N	83°8'23"W	204 m
Low Water Datum Detroit River (Fort Wayne station 9044036)	42°20'25"N	82°59'12"W	174m
Ambassador Bridge Towers rise 118m from water level	42°18'44"N	83°4'37"W	293m
Ambassador Bridge Roadbed Peak			220m
DTW Metro Airport	42°12'44.7"N	83°21'12.2"W	196.7 m
DET City Airport	42°24'33.5"N	83°00'36.6"W	190.8 m

⁴<https://www.seattlebikeblog.com/2013/01/17/the-steepest-streets-in-seattle/>

⁵<https://tidesandcurrents.noaa.gov/gldatums.html>

2.8 Literature Review

2.8.1 Annotation - Townsend 1995

Townsend, Bryan R and Fenton, Patrick C and Van Dierendonck, Keith J and van Nee, DJ Richard, *"Performance Evaluation of the Multipath Estimating Delay Lock Loop"*, 1995. [37]

The problem which the researchers/authors addressed:

"GPS pseudo-range and carrier phase measurements suffer from a variety of systematic biases. The sources of these are:

- (i) Satellite Orbit Prediction
- (ii) Satellite Clock Drift
- (iii) Ionospheric Delay
- (iv) Tropospheric Delay
- (v) Receiver Clock Offset
- (vi) Signal Multipath " [37]

The first four errors can be dealt with using modelling or differencing techniques, and 'receiver clock offset' is often the unknown variable solved for in determining position. Multipath presents a bigger challenges, degrades accuracy and increases signal processing time.

Previous work by others referred to by the authors:

The authors previously developed MEDLL (Multipath Estimating Delay Lock Loop) and appear to have pioneered these studies.

Shortcomings of previous work:

Delay Lock Loops (DLL) are discussed in general, including P-code DLL and of particular interest was the Early-Late DLL (E-L DLL). Previous GPS receiver code dedicated only two or possibly three correlators to each satellite tracking channel—while the MEDLL algorithm requires 10 or more.

The new idea, algorithm, architecture, protocol:

Multipath signals, whether bounced from buildings or the ground, always reach the receiver later than the primary signal.

Experiments and/or analysis conducted:

The authors appear to be principal researches at NovAtel. GPS equipment and software used to capture and record GPS system data was developed by NovAtel, specifically using antennas without choke rings. Experiments were done measuring signals from the NovAtel corporate rooftop in Calgary. Double difference residuals were calculated from the two receivers which allows removing satellite based errors. A 'short based' test and a simulator test were both described.

Results that the authors claim to have achieved:

The authors claim that they have demonstrated a 10 to 40% improvement using the MEDLL method on their ‘Short Based’ test, and yet state disappointment believing that the theoretical possibility is much higher. Apparently there was not enough multipath in that test to highlight the possible performance benefits. In the final results it is claimed that the proposed MEDLL receiver performance improves (reduces DLL multipath error) by up to 90% over the Narrow CorrelatorTM receiver.

Claims made by the authors:

Multipath signals tend to have one or two ‘strongest’ signals present at one time. It appears that their technique will accomplish the goal to improve position accuracy by determining satellites to exclude using their proposed method. This is seen by the authors as significantly useful for DGPS applications (those using a known fixed position ground based reference antenna in addition to any other mobile unit).

Citations to the paper by other researchers:

This was cited by 154 other papers according to Google Scholar.

2.8.2 Annotation - Psiaki 2001

Psiaki, Mark L, ”*Smoother-Based GPS Signal Tracking in a Software Receiver*”, Proceedings of ION GPS, 2001. [38]

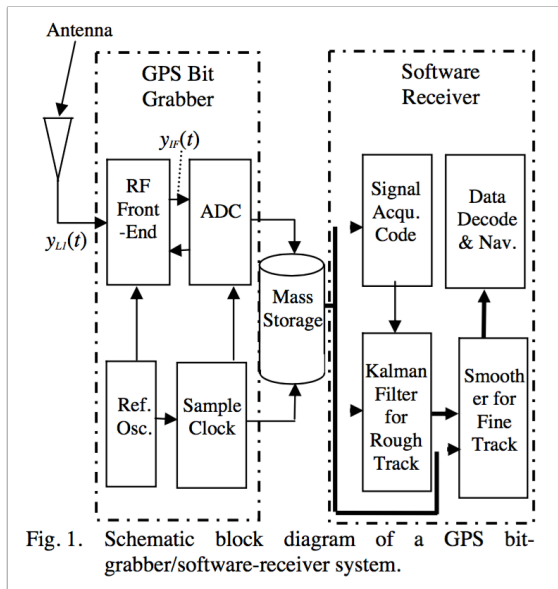
The problem which the researchers/authors addressed:

ABSTRACT: “Global Positioning System (GPS) signal tracking algorithms have been developed using the concepts of Kalman filtering and smoothing. The goal is to improve phase estimation accuracy for non-real-time applications. A bit-grabber/software-receiver has been developed for the GPS L1 coarse/acquisition signal. The bit grabber down-converts, digitizes, and stores the raw RF signal. The software receiver tracks each signal using a 2-step process. The first step uses phase-locked and delay-locked loops. The second step refines the tracking accuracy through the use of linear smoothing techniques. These techniques make optimal use of after-the-fact data.” [38]

The author says that this offline analysis work will help identify solutions for areas with low signal to noise ratio.

Previous work by others referred to by the author:

Figure 2.8: GPS Bit Grabber



GPS receivers all work as shown in their Fig.1 (my Figure 2.8) on the left.

“The Kalman Filter and Smoother modules in the software receiver implement functions like those of the DLLs and PLLs of a conventional real-time receiver: They estimate the phases of the code and the carrier. They also estimate the frequency and drift rate of the carrier. The main difference from a conventional receiver is that the smoother block uses correlations which extend past the time point of interest. These two blocks are the subjects of the remainder of this paper.” [38]

Shortcomings of previous work:

This is a new area using a software receiver to track GPS code signals.

The new idea, algorithm, architecture, protocol:

The author implemented his solution in software. Today we would call this ‘software defined radio’ (SDR). He cautions that his “use of the term ‘smoother’ and ‘smoothing’ do not imply the standard KF meanings.”

“The main contribution of this work is in the area of GPS signal smoothing, but there are two good reasons also to consider the subject of Kalman filtering of GPS signals. First, Kalman filtering is closely related to smoothing. Second, Kalman filters have been used to design a PLL for tracking carrier phase and a DLL for tracking code phase. The PLL and the DLL are needed in order to get the receiver’s replicas of the carrier and code to match closely with the received signal; otherwise, the linear models of this paper’s third section would not be valid for purposes of smoothing.” [38]

Experiments and/or analysis conducted:

The author noted that cycle slips and loss of signal lock are significant issues. He mentioned that the adaptation of these linear kalman filter methods using a non-linear KF might be required for these degraded signal areas.

Results that the author claims to have achieved:

The author has designed two software GPS receiver algorithms, which perform PRN code

phase tracking and carrier phase tracking, both acting on the civilian C/A signal, but only for offline processing. Both algorithms use KF's, the first one for a phase-lock loop (PLL) and the other for a delay-lock loop (DLL).

Claims made by the authors:

The author claims that using his method the results are quite good at tracking both code phase and carrier phase signals.

“The bottom line is that smoothers offer significant SNR improvements and the ability to track dynamic signals without introducing an estimation lag.” [38]

Citations to the paper by other researchers:

This was cited by 69 other papers according to Google Scholar.

2.8.3 Annotation - Groves 2005

Groves, Paul D and Long, Daniel C, *“Inertially-aided GPS signal re-acquisition in poor signal to noise environments and tracking maintenance through short signal outages”*, 2005. [18]

The problem which the researchers/authors addressed:

Poor signal to noise environments create GPS signal tracking gaps. This paper reviews a method using INS aiding to re-acquire GPS signal tracking in these degraded signal areas. The main focus is a technique for re-acquisition of GPS satellite signals in poor signal to noise environments using ‘aiding’ from the INS.

“This work is relevant to navigation in poor GPS signal to noise environments such as indoors, in some urban canyons and in the presence of jamming or interference. It is also relevant where frequent short-term blockage of GPS signals is a problem, such as navigation in urban areas and highly dynamic applications.” [18]

The authors say that integrated INS/GPS is required to operate in areas with degraded signal to noise, and that these following are the main three categories:

- (i) Unintentional interference;
- (ii) Weak signal applications;
- (iii) Deliberate jamming.

The Urban Canyon would be considered a ‘weak signal’ area.

Previous work by others referred to by the authors:

The authors referred to “controlled reception pattern antenna (CRPA)” systems, which is a solution that currently is restricted to military use due to size and cost. They state that

INS/GPS integration is necessary where GPS signals cannot be tracked. They also refer to their previous work *Combating GNSS interference with advanced inertial navigation* on increasing the number of Kalman filter states. They also discuss three general techniques for integrating INS and GPS which are i) loosely-coupled, ii) tightly-coupled and iii) deep. There is some discussion around these in several sections of the paper. QinetiQ's proprietary adaptive tightly coupled (ATC) technique is also discussed.

Shortcomings of previous work:

So called *AIDED RE-ACQUISITION THEORY* leverages all unused GPS tracking channels to reacquire the lost GPS signal. This assumes a 'well calibrated' INS/GPS system which implies that the user to satellite line-of-sight velocity is known. This means that only one satellite can be re-acquired at a time.

“Deep or ultra tightly-coupled integration techniques combine the GPS signal tracking and INS/GPS integration functions into a single estimation algorithm [e.g. 10, 11,12]. This increases the anti-jam margin by a few decibels over what ATC can achieve, but requires a redesign of the navigation system architecture, with a two- way 50 Hz interface between the GPS receiver and integration algorithm.” [18]

The authors also mention that “A problem with anti-jam integration techniques is that once a GPS signal is lost, it is very difficult to re-acquire it. This is because conventional signal acquisition algorithms require a very strong signal to noise environment” [18]

The new idea, algorithm, architecture, protocol:

The authors appear to be principal researchers at QinetiQ in Farnborough, UK, and they have developed a “proprietary prioritisation algorithm” for the situation where tracking multiple satellite signals have been lost, deciding which signal to reacquire first. Detailed statistical analysis are described in the theory section, including σ thresholds for evaluating results.

Experiments and/or analysis conducted:

QinetiQ Integrated Navigation Simulation (QINS) was used in their analysis. QINS incorporates “kinematic and INS models, a GPS receiver and satellite model and a reconfigurable set of ATC INS/GPS integration algorithms.” A simulated 10°/hour INS drift error was used similar to what the authors call a 'low-grade' INS, namely the Boeing Digital Quartz IMU (DQI). Five satellites were tracked using P(Y) code signals over four re-acquisition simulation scenarios. The duration of each simulation was four minutes, and one P(Y) signal was jammed for two minutes or more in each test. Multiple runs of these simulations were done at various threshold settings to create the data tables. “Tracking lock was detected by comparing the measured carrier power to noise density, C/N_0 against a threshold.”

Results that the authors claim to have achieved:

- inertially aided re-acquisition is practical for $C/N_0 > 10$ dB-Hz.

- results confirm that increasing the detection threshold increases the re-acquisition time
- results show that a quicker re-acquisition can be attained by reducing the correlator spacing
- the number of false re-acquisitions was significantly greater than predicted

Claims made by the authors:

“A simulation assessment of code tracking maintenance has been conducted using QINS. Again a P(Y) code GPS receiver and 10/hr INS were simulated. With a well calibrated INS and four satellites tracked with strong signal to noise ratios, the code tracking maintenance algorithm was able to operate for a 200 s outage with code tracking resuming almost immediately after the reintroduction of the signal. Longer outages have yet to be simulated.” [18]

Citations to the paper by other researchers:

This was cited by 7 other papers according to Google Scholar.

2.8.4 Annotation - Meguro 2009

Meguro, Jun-ichi and Murata, Taishi and Takiguchi, Jun-ichi and Amano, Yoshiharu and Hashizume, Takumi, *“GPS Multipath Mitigation for Urban Area Using Omnidirectional Infrared Camera”*, 2009. [5]

The problem which the researchers/authors addressed:

Can GPS Multipath Mitigation be implemented using an infrared camera image to determine areas that should be masked out, thereby including only LOS satellites while excluding the others?

Previous work by others referred to by the authors:

Fixed, floating and Differential GPS results are compared in static test as a baseline for understanding position results. It appeared to the authors that the popular multipath estimating delay-locked loops (MEDLLs) method could be improved upon by their method of excluding the satellites causing the position errors.

Shortcomings of previous work:

Positioning antennas away from buildings or specially designed (choke rings) are impractical for Connected Vehicle OBU's. Narrow-correlator technique from early 1990's noted as *“can eliminate multipath errors better than the former methods”* [5] although it is not discussed in depth.

The new idea, algorithm, architecture, protocol:

Create a dynamic infrared sky image model to use in determining which satellites are truly in LOS. The authors refer to NLOS satellites as ‘invisible’ satellites. They proposed a model to measure signal timing errors and use this data to infer the inherent error; thereby deciding whether or not a particular signal should be excluded.

Experiments and/or analysis conducted:

A very well provisioned test vehicle with triple GPS antennas, a specially constructed dual mirror (rather than fisheye) lensed omnidirectional infrared camera was fitted to a minivan test vehicle. Static and slow speed (under 20 km/h) tests were conducted to determine viability of the concept. GrafNav software from NovAtel was used to assist in recording live data streams and further in post processing. This greatly aided the efforts to understand exact satellite positions which is required to determine whether or not a satellite was ‘invisible’.

Results that the authors claim to have achieved:

A concept for measuring ‘dilution of precision’ (DOP) is used, and by excluding so-called ‘invisible’ satellites it is suggested that overall DOP may be reduced in many cases. The authors noted that their approach improved position accuracy even where it is clear from their results data (in the static position test specifically regarding the DGPS comparison) that DOP has in fact increased.

Claims made by the authors:

Multipath errors were measured to indicate 20-60m for the ‘invisible’ satellites. It appears that their technique will accomplish the goal to improve position accuracy by determining satellites to exclude using their proposed method.

Citations to the paper by other researchers:

This was cited by 94 other papers according to Google Scholar, which would indicate an active level of interest in this research.

2.8.5 Annotation - Khodjaev 2010

Khodjaev, Jasurbek and Park, Yongwan and Malik, Aamir Saeed, “Survey of NLOS identification and error mitigation problems in UWB-based positioning algorithms for dense environments”, 2010. DOI 10.1007/s12243-009-0124-z [39]

The problem which the researchers/authors addressed:

Non Line of Sight (NLOS) identification and mitigation techniques are each classified into tables for comparison purposes. NLOS identification classifiers are (i) range estimates, (ii) channel statistics, and (iii) 3-D building and terrain maps. Mitigation classifiers are

based upon direct path and statistical detection.

Previous work by others referred to by the authors:

NLOS identification comparison in their table 1 refers to seven papers. Mitigation techniques comparison in their table 2 refers to eleven papers.

Shortcomings of previous work:

Previous NLOS identification papers all seemed to either lack “*threshold selection for decision making*” [39] or had calculation troubles—too many calculations to be useful unless done in offline processing, or no comparison with previous work. Building map details must be very precise, and done in 3D the ray tracing is noted as complex and very time intensive to calculate. Earlier mitigation techniques are more interesting. The Least Squared method appears to be very computationally expensive. The single Linear Program method described only works if at least one LOS source is available. Direct Path detection attempts also take a long time to calculate. Only the Filtering techniques appear to have robust ability to adapt to real time use.

Claims made by the authors:

Channel statistics techniques appear to perform better than other methods of NLOS channel identification. Non-iterative methods are recommended to reduce computational complexity. NLOS mitigation techniques that are most widely used are statistics based. The authors believe that since the NLOS signal travels what they refer to as an ‘excess’ path, a statistics based method is the best solution.

Citations to the paper by other researchers:

This survey has been cited by 55 other papers according to Google Scholar.



Identifying & Mitigating GPS Anomalies

A research paper from the ITS 2014 World Congress [30] piqued my interest in the Urban Canyon issue. The authors stated in this would be the first Urban Canyon connected vehicle testbed in North America. They noted the following:

“To date, no Test Bed has been deployed in an urban canyon environment. During initial testing, the vehicles in the urban canyon area are not able to lock on to as many GPS satellites as they would in an open area. As a result, the accuracy of the in-vehicle GPS systems is significantly reduced, and some vehicles on the GUI appear to run into buildings or disappear due to these coverage gaps (especially in the tunnel section of the test bed where it takes significant time for the GPS system to re-lock onto the satellites). Significantly more research is going to be needed to determine the accuracy of GPS in urban canyons and to identify means to overcome those accuracy issues.”

p11, Detroit Builds First Urban Canyon Testbed [30]

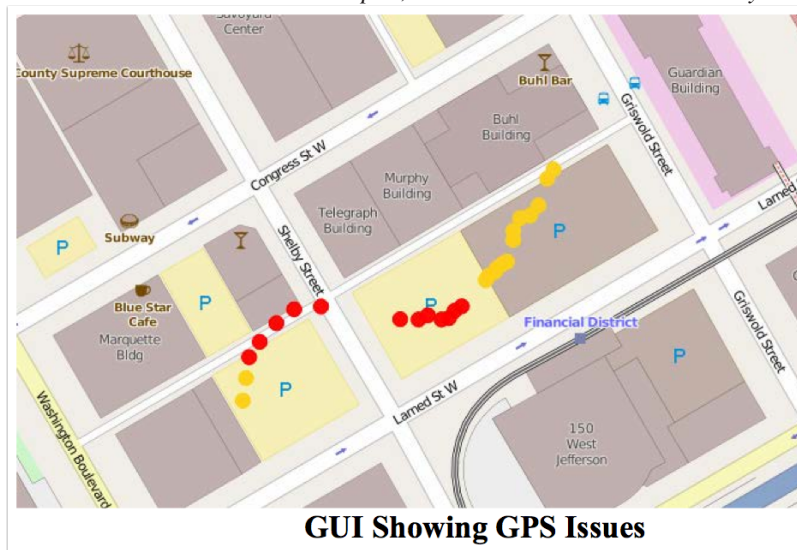


Figure 8, p11, Detroit Builds First Urban Canyon Testbed [30]

3.1 Research Methodology Overview

Reviewing data path traces containing GPS signals, it became apparent these position fixes are at best estimations. Details of this issue are shown in §4 **Results**. As signals are interpreted and processed, an estimate of position is given which should also contain an estimate of the precision. Commercial units like the Garmin which I tested provide latitude, longitude and altitude—yet position accuracy is not easy to determine. Their website [40] claims “extremely accurate” position fixes due to twelve specific parallel channel receivers and then states their equipment has an average accuracy of 15m. Bednarz [41] discusses generally excellent but not assured accuracy of GPS positions. The Arada Locomate On-Board Unit datasheet boasts a “less than 1 m” accuracy [42].

GPS position was available in all datasets, including the following fields:

- ϕ which represents the latitude,
- λ which represents the longitude, and
- h which represents the altitude

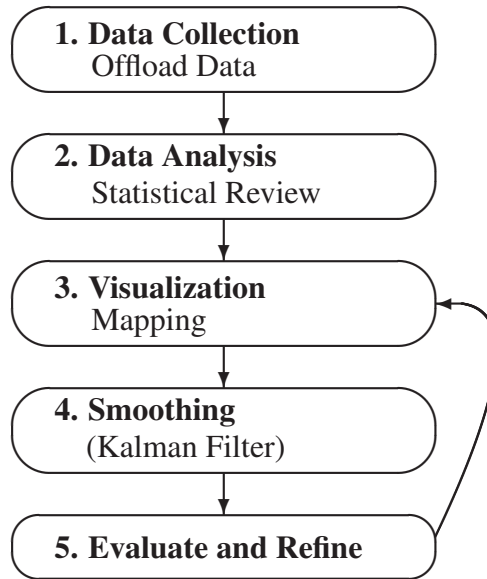
This becomes the initial state vector for GPS only calculations, and the beginning portion of the state vector for the full OBU calculations.

Reviewing the literature there were several options available to extract the best data fit, given that the data appears noisy (see [Figure 4.7](#)). Digital signal processing uses a variety of methods, with the top two most popular being the Viterbi algorithm and Kalman filtering. The most popular for navigation and guidance applications appears to be the Kalman filter, and this is where my research has focused.

3.2 Work Flow Diagram

The work flow to examine the Urban Canyon issue is shown here. A detailed explanation is given below in [section 3.3](#) “Steps Completed”. These steps were taken in gathering and processing data for the Urban Canyon project:

Figure 3.1: **Workflow Diagram**



3.3 Steps Completed

My approach to improving the Urban Canyon GPS location problem is as follows:

1. **Data Collection** of two distinct classes were collected and used to further my research. These are described below:
 - (a) As a first step I duplicated the issue noted in the “Detroit Builds First Urban Canyon Connected Vehicle Test Bed” report [30], using data from a commercial Garmin Nüvi GPS unit, mounted above the dashboard on the windscreen. Details are described in [subsection 3.4.1](#)
 - i) Collected GPS data, including date, time, latitude, longitude, and altitude.

- (b) A second dataset was gathered, containing DSRC OBU Data obtained from an Arada test vehicle. The GPS antenna mount was above the rear-view mirror. Details are given in [subsection 3.4.2](#)
 - i) Collected OBU data, including GPS and IMU information, such as utc (epoch) date, time, various GPS data such as latitude, longitude, and altitude, and IMU available data which also contains velocity and acceleration in the x, y and z directions.

The data gathering resulted in several datasets of both cases, which helped to gain perspective on the data elements that were available and those which would clearly not be helpful (a turn signal indicator or trunk light on would have no effect on the journey). Two additional OBU connected data files were also examined, one sampling at 200ms or five times a second (5 Hz), and the other containing fifteen IMU sensor input cycles per second, while the GPS remained constant on both of these until the *utc_time* changed each second. Many reports in the literature indicate that the GPS are commonly updated at 5Hz instead of the 1Hz cycle that was present in our data.

2. Data Analysis

Data analysis began with importing the csv data files into MS-Excel. Standard statistical measures were made included μ mean, and σ standard deviation for the state vector details. Calculations were performed to determine geographic distances between consecutive points. Details of these calculations are found in [subsection 3.5.2](#). I also examined the data with data mining (and statistical analysis) software from the University of Waikato, which is their “Waikato Environment for Knowledge Analysis” known widely by it’s acronym **Weka**. Weka results can be seen in Appendix B on page [84](#).

- 3. **Visualization** of data using comparison charts and 2-D maps to see the effects of the urban canyon data. An explanation of these calculations is found in [section 3.6](#). The two classes of data collected are as follows below:

- (a) GPS only data, and
 - (b) GPS including CAN bus data
 - i) Review detailed datasets and chart individual components, to understand where the data is becoming non-linear.
 - ii) Create comparison charts showing the cross-correlation between individual data columns, demonstrating how position error becomes visible
 - iii) Determine statistical details, including mean and standard deviation of the dataset.
4. **Smoothing using Kalman Filter** Python and iPython programs. Details of the Kalman Filter prediction correction cycle, and the python and iPython programs are found in [subsection 3.7.1](#).
- (a) Explore the use of various Kalman Filter algorithms
 - i) Standard KF expects linear functions and Bayesian distribution of noise.
 - ii) Extended KF and Unscented KF handle non-linear probability distributions.
5. To **Evaluate and Refine** the data, an iterative approach was taken, with descriptions and details found in [section 3.8](#) regarding the following data and calculations:
- i) Limited data from the preliminary GPS only (lat,long,alt) or (ϕ, λ, h)
 - ii) More extended data based on GPS (lat,long,alt) including additional IMU data
 - iii) Calculated various interpolations of data, including use of Haversine formula for great circle distances, IMU data for yaw and yaw rates.
 - iv) Visualize data on a 2-D map to see the effects of the urban canyon on data updated through KF filtering and smoothing.
 - v) Provide various outputs, charts, etc. to understand the components of the KF and related data.

3.4 Data Collection

3.4.1 GPS Data Collection

Since the Detroit DSRC Testbed had demonstrated problems as noted by the engineering paper [30], I decided to start there. This testbed contains several sections which are clearly Urban Canyons, blocking well over 50% of the sky view, and some sections are over 95% obscured. The buildings block enough of the satellite signals so that the effects are dramatic and it is therefore relatively easy to reproduce the Urban Canyon problem.

Preliminary data was gathered by using a conventional Garmin nüvi 2798 receiver, travelling through the ITS Detroit DSRC testbed. The GPS receiver was mounted inside the windshield using a suction cup mount, and powered through the auxiliary power port. I then drove through the Testbed area on two separate occasions, establishing that the GPS was operating well and capturing data. On both trips I was able to collect GPS path data. Once the driving tests were complete, the GPS unit was removed from the vehicle and data was then offloaded for further processing.

The Garmin hardware and software were closely coupled, and this integration allowed a very nice 2-D map visualization of the path driven. These following are the software tools for the Garmin: (i) Basecamp, (ii) Connect, (iii) Express, (iv) MapInstall and (v) MapManager.

Basecamp is the most useful software, specifically handling maps and also allowing data export in several formats. I used the GPX format for initial 2-D map visualization which proved quite useful. I also exported using CSV (comma separated values) format file, which highlighted some of the interesting details behind Garmin's data collection philosophy.

The Garmin nüvi recorded trip waypoints which can be used to provide post trip mapping. The interval is not user controllable, it seems to vary from each second to about 45 seconds on the primary run examined, with an average of just over 7 seconds. During periods of imprecise position details the intervals were very close together (one second), and longer gaps when at steady speed on relatively straight roadways. This was adequate

to determine the position, direction of travel and the velocity. In addition to the latitude and longitude I also was able to obtain the altitude data. This data provided a working set sufficient as the initial proof of concept for the proposal. These details are visible in §4 **Results** under [Figure 4.3](#) and [Figure 4.5](#).

What became immediately apparent during the driving test was the behaviour of the on-screen map display, which worked fine in normal highway operation yet became ‘indefinite’ and jumpy in the degraded signal areas. It was visible that a problem was occurring for the system to display and update the on-screen image, but it seemed to be solved by the system lagging and updating when new info became available. After the driving test, the data was offloaded for further analysis. More details on this are shown in §4 **Experiment Results** on page [52](#), the main point here is that I noticed from most 2-D mapping applications altitude is ignored, almost as if it were considered to be ‘noise’. The nüvi GPS data clearly grabbed what it had calculated for altitude, and a noticeable difference was visible in altitude (as seen in [Figure 4.7](#) on page [56](#)). Position path data was pretty consistent, until the ‘Urban Canyon’ was encountered – Altitude appeared to contain quite a bit of the error.

3.4.2 OBU Data Collection

DSRC data was a bit more difficult to obtain. Negotiating US DOT paperwork with University of Windsor Office of Research (ORIS) commenced in November 2014, and the agreements were not completed until May 2015. During May 2015 a data recording was done on a driving test thorough the Detroit DSRC Urban Canyon and surrounding area. This data set was representative of the current state of technology in DSRC OBU’s, and included GPS position and altitude, and many (over eighty) CAN bus details, several of these are interesting to this proposal.

I was fortunate to have access to On-Board Unit (OBU) data from an Arada test vehicle. This included Inertial Measurement Unit (IMU) data in addition to the GPS receiver data. The GPS antenna was mounted inside the windshield above the rear view mirror using double faced tape. The OBU was powered directly through the vehicle CAN Bus operating on vehicle power. The test driver drove through the Testbed and

surrounding area capturing data. This included portions of the Detroit DSRC testbed as well as two adjacent tunnels on Atwater street. Tunnels are considered a ‘degraded’ GPS environment. [32] Once the driving tests were complete, the data was made available and offloaded for further processing.

3.5 Data Analysis

3.5.1 Questions on Data from the Urban Canyon

Regarding the preliminary GPS trace, I posed the following questions: (i) what may be learned from the data gathered? (ii) Will it be possible given only the *a priori* path data (say for instance beginning at t_0 over the last several measurements) to predict or estimate based on current time t the possible or probable position at time $t+1, t+2, \dots, t+10$? (iii) Moreover, if looked at from the standpoint of error estimation can we predict when a reading is in fact outside of allowable limits? (iv) What would help us determine and set acceptable limits? (v) And if, having identified this erroneous reading, is there a method to mitigate this error and restore the system accuracy? Let’s begin with the preliminary GPS data.

3.5.2 Statistical Analysis of OBU Data

Path trace data from the Urban Canyon was gathered by the Arada OBU. GPS and CAN Bus data are combined at the OBU and the information was offloaded for analysis. The data included many useful fields which are outlined in Appendix C Table C.9 on page 86. Our Arada OBU research data originally contained 4,920 rows with 84 columns, and upon review 39 of these were duplicate columns. The CSV formatted data file shows that the GPS was only being updated once a second, although the other CAN Bus data became available more frequently. Depending on which dataset was being reviewed the CAN Bus data could often be updated fifteen to twenty-two times per second. GPS updates were very consistently given each second, which may be by design or firmware setup in the Locomate [42] OBU.

My first thought with nearly 5,000 rows of data and 45 remaining unique columns was to look at a machine learning technique, to see what statistical correlations could be drawn. From looking at the data I noticed the Altitude was a significant factor, since altitude varied widely most of the time when the track was jumping on the 2-D maps. I used Weka and attempted many combinations that proved unsatisfactory in drawing any real correlations. I have included results from the Arada Test data which can be seen in Appendix B on page 84, which correlated nicely on the altitude using the popular ‘random forest’ statistical grouping technique.

Graphs depicting GPS latitude, longitude and altitude were correlated so that the errors could be visualized (see Figure 4.7). Vertical line segments show clear indications of where the data shows discontinuity. A statistical analysis of the raw data demonstrated to me that these large vertical steps were demonstrating a large variance from the mean in each individual aspect. Let h represent altitude [21] and the mean be \bar{h} . When considering the mean altitude \bar{h} , considered over the last five, ten, or twenty seconds, it was found that the last 20 seconds gave far better results. The standard deviation between the was much greater than I would have anticipated. In one dataset the altitude standard deviation was 7.8m, and another it was as much as 22m. For the standard deviation $\sigma=7.8\text{m}$, 68% of the altitudes should fall into $\pm 1\sigma$, 95% should fall into $\pm 2\sigma$ from the mean (\bar{h}). It appeared that the largest data set possible for these measurements would yield better results, and that the statistical outliers specifically for altitude were considerable when in the area of the urban canyon.

Although the time interval is quite small (changing each second), the GPS distance and directions were calculated at every step using the Haversine formulas. The “spherical law of cosines” provides poor accuracy when points are too close together, due to computational results from rounding errors. Haversine calculations appear to provide better accuracy over these shorter distances which are consistently involved in V2V research. These distance comparisons were done to determine where the most likely disagreement was between GPS signals compared to the IMU results.

3.5.3 DSRC Details for On-Board Unit Data

My review focussed on the direct data collection of first GPS only path details, and second OBU direct output of integrated data, including GPS and CAN Bus information. This was convenient for my analysis, and is the way that any correction to the OBU will be implemented. Examining these data files it was not readily apparent what the intent was for several fields. Looking beyond the direct OBU data, I noticed some interesting items when reviewing DSRC OBU data specifications. Several details that were not clear in the IEEE 802.11p [43] standard specifications, but were more clearly shown in the 1609 specs [44, 45]. The 1609.2 [44] describes how GPS positions are stored as shown below:

The latitude and longitude fields contain the latitude and longitude as an sint32 type, encoding the latitude and longitude in 1/10th integer micro-degrees relative to the World Geodetic System (WGS)-84 datum as defined in NIMA Technical Report TR8350.2.

Data Example (from IEEE 1609.3 specs, p123(2010-August)):

3D Location And Confidence

Hex Example	WAVE Element
06	ID=6
0F	Length=15
01 7A 12 AC	latitude (4 octets): + 24.777388°
07 36 F8 BB	longitude (4 octets): +121.043131°
03 E8	elevation (2 octets): +100.0m
36	* <u>position confidence (4 bits): 3=100m</u>
	* <u>elevation confidence (4 bits): 6=10m</u>
FF FF FF FF	positional accuracy (4 octets): unavailable

* Table 3.7 below describes half-byte fields for Position and Elevation Confidence

Table 3.7: **Position / Elevation Confidence**

unavailable (0)	B'0000		Not Equipped or unavailable
a500m (1)	B'0001	500m	or about $5 * 10^{-3}$ decimal degrees
a200m (2)	B'0010	200m	or about $2 * 10^{-3}$ decimal degrees
a100m (3)	B'0011	100m	or about $1 * 10^{-3}$ decimal degrees
a50m (4)	B'0100	50m	or about $5 * 10^{-4}$ decimal degrees
a20m (5)	B'0101	20m	or about $2 * 10^{-4}$ decimal degrees
a10m (6)	B'0110	10m	or about $1 * 10^{-4}$ decimal degrees
a5m (7)	B'0111	5m	or about $5 * 10^{-5}$ decimal degrees
a2m (8)	B'1000	2m	or about $2 * 10^{-5}$ decimal degrees
a1m (9)	B'1001	1m	or about $1 * 10^{-5}$ decimal degrees
a50cm (10)	B'1010	0.50m	or about $5 * 10^{-6}$ decimal degrees
a20cm (11)	B'1011	0.20m	or about $2 * 10^{-6}$ decimal degrees
a10cm (12)	B'1100	0.10m	or about $1 * 10^{-6}$ decimal degrees
a5cm (13)	B'1101	0.05m	or about $5 * 10^{-7}$ decimal degrees
a2cm (14)	B'1110	0.02m	or about $2 * 10^{-7}$ decimal degrees
a1cm (15)	B'1111	0.01m	or about $1 * 10^{-7}$ decimal degrees

SAE J2735 detailed specifications on PositionConfidence

3.6 Visualization

Each record represented many data fields and recorded these at a specific time. Using MS-Excel allowed interpolation of information relating each (row) record to the prior record. This was very useful in determining and double checking heading, velocity, and rates of change for latitude, longitude, altitude, etc. Breaking up these individual columns to provide charts allowed comparisons of the specific positions details, latitude, longitude and altitude at specific times which provides a great visualization tool. The results of these charts are presented in [Figure 4.7](#) on page 56.

The Arada Locomate “Classic” OBU [42] uses standard UTC time format and synchronizes the internal clock to the signal gathered from the satellite transmissions. This records date and time as the number of seconds from January 1, 1970, known as the “Unix epoch” (literally ‘the beginning of something’). Excel counts time from January 1, 1900, and the Unix epoch begins at 25569 for Excel. UTC time is converted for spreadsheets as `utc_time + 25569 + time zone adjustment` using our local time zone correction of

GMT-4 hours, which for this calculation is therefore $TZ = -4/24$. The Excel formulas for this is given as follows (assuming the `utc_date` is in cell B2):

$$\text{Excel Date number} = B2 / (60 * 60 * 24) + 25569 - (4/24) \quad (3.1)$$

$$\begin{aligned} \text{Excel Date (human readable)} = & \text{TEXT}(B2 / (60 * 60 * 24) + 25569 - (4/24), \\ & "yyyy - mmm - dd hh : mm : ss") \end{aligned} \quad (3.2)$$

Excel dates are adjusted to the local computer time zone, where the GPS `utc_time` is aligned with GMT. Our standard time adjustment of GMT-05:00 becomes GMT-04:00 during daylight savings time for this area, explaining the use of “-(4/24)” in these equations. [Equation 3.1](#) leverages Excel’s internal formatted dates to allow all date calculations, and [Equation 3.2](#) is most convenient for visual comparisons.

The Arada OBU recorded two distinct velocity fields which are the GPS speed and the CAN Bus speed. The measurements are provided in km/h. A consistent lag between these allows comparison of the current second GPS speed with the prior second CAN Bus indicated speed, illustrated on the chart in on [Figure 4.6](#) on page 54. As we will see in §4 when these speeds vary by more than a **1.5 km/h threshold** there is normally something happening to indicate a position issue especially when a group of records all exhibit this condition, although in isolation it could also be caused by uneven braking or acceleration forces.

Haversine calculations between consecutive positions were are calculated. This allows comparison of great circle distances travelled. The calculations give results in metres travelled, and the conversion formula $3.6 * x \text{ m/s} = y \text{ km/h}$ allows convenient further comparison to OBU speeds.

MS-Excel does run into some problems with proper calculation of the haversine formulas, notably the required ACOS function displays an error code in five of the five thousand (or 0.1%) of the Locomate OBU records, where the calculation cell results are shown as #NUM!. The ACOS is expecting values in the range between [-1,1] and depending on the precision chosen and rounding, you can inadvertently pass a $\pi/2$ (≈ 1.57) to be sent to the function, which is outside of this allowable range.

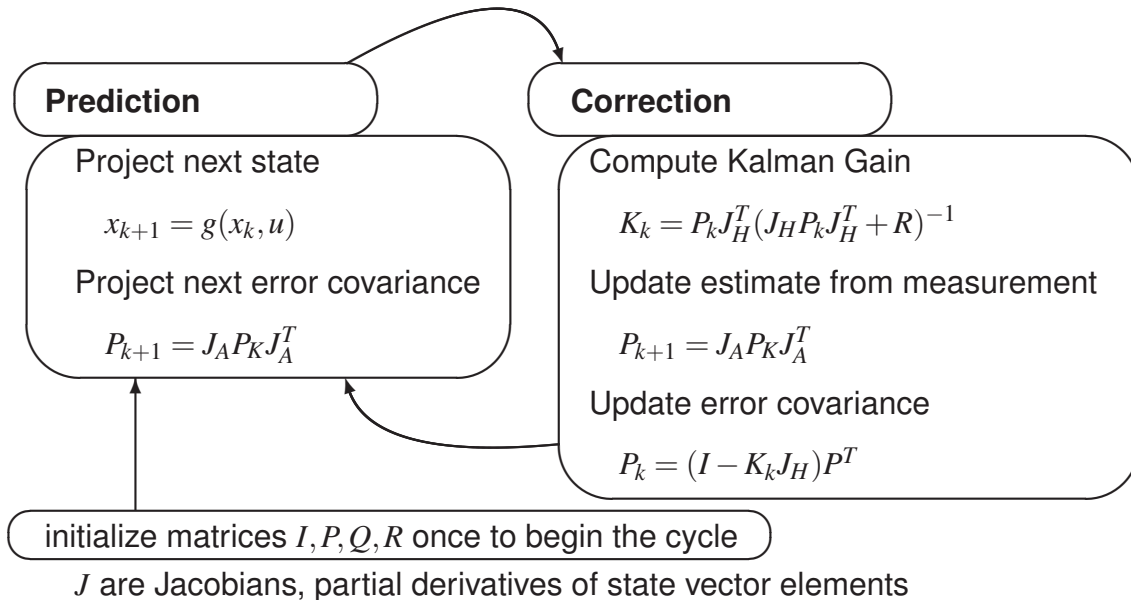
3.7 More Kalman Filter Information

3.7.1 Kalman Prediction Correction Cycle

As you will recall the standard Kalman Filter is designed for the specific case of linear filtering. With many natural processes behaving strictly in a non-linear fashion, non-linear functions have been developed. The Unscented Kalman Filter (UKF) is used in Robotics for SLAM (System Localization and Mapping). The UKF has higher computational complexity, on the order of n^3 [26] where n is the number of elements in the state vector. For the special case of state estimation the Extended Kalman Filter (EKF) is on the order of n^2 [26], and it is the defacto standard for navigational and guidance systems use.

The method for using Kalman Filters is to follow a process of of initializing the known states, by building a training data set, and using this a starting point for loading the formulas and equations, to begin running through the KF algorithm. The details of the algorithm are as follows:⁶

Figure 3.2: Kalman Filter-Smoother Prediction-Correction Cycle



⁶<http://bilgin.esme.org/BitsAndBytes/KalmanFilterforDummies>

3.7.2 Kalman State Vectors

For each case of Kalman Filter the state vector changed.

Standard kalman filter using only GPS data:

$$\text{state vector } x_t = (\phi, \lambda, h)$$

Standard kalman filter using GPS and CAN Bus data:

$$\text{state vector } x_t = (\phi, \lambda, h, \text{x-vel}, \text{y-vel}, \text{z-vel}, \text{x-accel}, \text{y-accel}, \text{z-accel})$$

Extended kalman filter using GPS and CAN Bus data:

$$\text{state vector } x_t = (x, y, \text{yaw } \psi, \text{vel}, \text{yaw-rate } \dot{\psi})$$

Unscented kalman filter using GPS and CAN Bus data:

$$\text{state vector } x_t = (\phi, \lambda, h, \text{yaw}, \text{yaw-rate})$$

3.7.3 Kalman Filter Explanation

The Kalman “Prediction-Correction” cycle with Jacobians can be a confusing, and so it seems the EKF and UKF are a bit more intuitive to me. State estimation of a system at time t evolves from the prior state at time $t - 1$ according to the following equation:

$$\mathbf{x}_t = \mathbf{F}_t \mathbf{x}_{t-1} + \mathbf{B}_t \mathbf{u}_t + \mathbf{w}_t \quad (3.3)$$

where

\mathbf{x}_t is the state vector containing the terms of interest for the system at time t and time $t - 1$ (i.e. position, velocity, heading)

\mathbf{F}_t is the state transition matrix, used to apply the effect of each system state parameter at time $t - 1$ to the system state at time t
(i.e. position and velocity at time $t - 1$ both affect position at time t)

\mathbf{B}_t is the control input matrix which applies the effect of each control input parameter in the vector \mathbf{u}_t on the state vector
(i.e. applies the effect of the throttle, brake, steering settings on system velocity and position)

\mathbf{u}_t is the control inputs vector (in our case containing steering angle, throttle setting, braking force and other CAN bus data)

\mathbf{w}_t is the vector containing process noise terms for each parameter in the state vector. Process noise is assumed to be taken from a zero-mean multivariate normal distribution with covariance matrix \mathbf{Q}_t

Measurements of the system can also be performed according to the model

$$\mathbf{z}_t = \mathbf{H}_t \mathbf{x}_t + \mathbf{v}_t \quad (3.4)$$

where

\mathbf{z}_t is the measurement vector

\mathbf{H}_t is the transformation matrix which maps the state vector parameters into the measurement domain

\mathbf{v}_t is the measurement noise vector corresponding \mathbf{z}_t . Like the process noise, the measurement noise is assumed to be zero-mean Gaussian white noise with covariance \mathbf{R}_t .

3.8 Evaluate and Refine

Once the first dataset (GPS only) was available I used the Garmin BaseCamp software to produce 2-D maps (shown in [Figure 4.3](#) and [Figure 4.5](#) on page 52) of the two driving paths taken. Results showed that as suspected the GPS worked perfectly outside of the Detroit DSRC testbed, and also I was able to duplicate the issue noted by the testbed engineering paper. [\[30\]](#) With this information plotted, I then examined the spreadsheet to correlate the mapped points with the detailed records, paying special attention to the ‘interesting’ areas. I found that the delta differences between consecutive records were very interesting.

The OBU provided both GPS and IMU sensor data, and comparing the current time *GPS velocity(t)* with the CAN Bus *velocity(t-1)* it was clear that a difference of more than 1.5 metres per second was indicative of a GPS error.

Comparing GPS velocity with the haversine calculated velocity I also found that a difference of more than 1 metre per second was a good indicator that an issue existed.

Altitudes in a group of records were recorded as below 175 m (the GPS reports these as metres above sea level). From the local chart datum [Table 2.6](#) this is clearly below water and not possible for our driving test.

Looking at the Δh (altitude) a difference of more than 1 metre per second was also highlighting a problem.

$\Delta\phi$ (latitude) and $\Delta\lambda$ (longitude) did not independently show much information, however the haversine calculations did shed some light on what was happening. The distance calculation provides the meters separation and this converts quickly to kilometers per hour allowing comparison back to the IMU data. Speeds in excess of 120 km/h are not common in this urban area, with speed limits of 50 km/h. A bound can be established that Δspeed must not exceed 120km/h or 33.3 m/s. When the urban canyon issue surfaces this is quite often the case.

With these calculations completed I began to put the data through the Kalman Filters to establish how they worked, would they do the job, and what else could be done.

3.9 Urban Tunnels

GPS requires direct line of sight (and it's interesting also GPS does not work under water). It's worthwhile noting that within 500m of the Detroit testbed area there are six significant tunnels. The tunnels are as follows: on US-10 southbound under Cobo Hall, either taking (i) exit 1A 310m to West Jefferson exit or (ii) exit 1B 285m onto Larned, through (iii) 86m under the Millender Center, or on (iv) Atwater street 174m under Hart Plaza or (v) 150m under Cobo Arena from the DSRC test drive, and also the (vi) 1.68km Detroit-Windsor Tunnel). These long tunnels are GPS shadow areas and special treatment will need to be taken to make their position reports accurate.

3.10 GPS Position Discard Threshold Method

Examining the various standard correction routines it became clear that another approach was necessary. Normally some version of the Kalman Filter, whether standard, extended or unscented should have been able to deal with the inconsistencies. The standard kalman filter was not useful to smooth the urban canyon data, and the extended (EKF) did a much better job. Altitudes from the GPS are clearly not correct much of the time, and the statistical details for intermediate points led to seeing a bigger picture of pre-filtering or combing the data. Given these various parameters how would a pre-cleanup of the data improve position accuracy? These are several parameters that appeared interesting to consider:

1. Altitude Change

- (a) Recalling our road grade example from page 20 in Figure 2.7 the most extreme grade in the world would be less than a rise/run of 3/8, ($\angle \alpha 20.6^\circ$) meaning that the altitude cannot ever change by more than 37.5% of the distance covered.
- (b) Hilly area road grades nearby extreme grade areas (some urban San Francisco or Seattle streets) should normally be less than 1 in 4, or 25% grade. This would be indicated by a limiting factor of 1/4 (=0.25) of the linear distance covered for these areas.
- (c) Maximum city and highway road grades outside of the mountainous or hilly areas should normally be less than 1 in 19, or 5.3% grade or 3° . This would be indicated by a limiting factor of 1/19 (=0.053) of the linear distance covered.

2. Distance Travelled

- (a) OBU velocity is given in metres per second. $36 \text{ km/h} = 10 \text{ m/s}$, giving us a 3.6 factor to compare for impossible or improbable velocities (our data noted speeds an order of magnitude higher when the GPS position reports were compromised)

3. Direction Travelled

- (a) Haversine calculations provide GPS vector details including direction and magnitude

4. Direction Changes (yaw rate ψ)

- (a) turns should be ‘normal’ or standard rates, commonly <0.1 rads/sec [46]

5. Hysteresis effects on velocity from OBU vs. GPS

- (a) A one second hysteresis effect (lag) in GPS ‘catching up’ with the IMU data was noted above in [Figure 4.6](#)
- (b) When the IMU indicated the vehicle was stopped, GPS reported velocity and position were inaccurate.
- (c) The Garmin nüvi internal database provides a speed limit warning display. This valuable resource should be integrated into all CV systems.

6. Path Prediction using IMU Data

- (a) IMU data will confirm slowing or speeding up, and help us understand whether or not a turn is reasonable.
- (b) Considering the last five seconds of travel do the GPS and IMU roughly indicate the same track? Believing the IMU data is fine, do we trust the GPS vector (haversine direction and distance)?
- (c) where the ‘predicted’ path and the GPS path diverge, can we use strictly CAN data until the GPS regains enough data to provide an accurate fix?
- (d) identifying that our GPS cannot be trusted has been established in all these preceding steps. How will we establish when can we trust our GPS position again?

The Garmin nüvi databases were loaded with relevant traffic zone information, including school zones, speed limit details which were very accurate, and the emergency notification system that can help alert drivers to public service messages about impending slowdowns, it appears that we should expect more information will be available to assist

drivers. Also for these purposes I'd like to see Altitude included so that impossible low or high altitudes would be bounded, and erroneous data thereby discarded. It would be good to know if we should use the 'extreme', merely 'hilly' or 'normal' grade constraints. In the specific case of our data I used the [Table 2.6](#) elevation data as a 'floor' and would not allow 'below water' altitudes.

3.10.1 GPS Position Discard Threshold – Pseudocode

To elaborate on the GPS Position Discard Threshold (PDT) Method from [section 3.10](#), I have included a pseudocode routine. The pseudocode should better describe how the limiting criteria are chosen, and how they are proposed to be applied. The outcome after applying the GPS PDT method are given in [section 4.5](#), with the key result mitigation illustrated in [Figure 4.12](#).

GPS Position Discard Threshold – Pseudocode

' GPS PDT Method for Eliminating Erroneous GPS path data

' **INITIALIZE** (and regularly re-initialize):

' General factors to consider before implementing the PDT Method

' Run these steps to initialize processing and re-run every 5 or 6 minutes, sooner

' if there are sustained altitude changes (more than one minute at a significantly

' [± 25 m] changed altitude) or speed limit changed (such as highway to city streets).

1. Altitude Change limits are based on terrain, as described in [Figure 2.7](#)

Determine which geographic area best suits conditions from the following:

(a) **Extreme Grade:**

the most extreme grade in the world would be less than a rise/run of 3/8, ($\angle \alpha 20.6^\circ$) meaning that the altitude cannot ever exceed 37.5% of the distance covered. Limiting factor (**=0.375**)

(b) **Hilly Grade:** (should cover San Francisco, Seattle, etc.)

This should be less than 1 in 4, or 25% grade.

A limiting factor of 1/4 (**=0.25**) of the linear distance covered for these areas.

(c) **Normal Grades:**

maximum grades outside of the extreme or hilly areas should normally be less than 1 in 19, or 5.3% grade or 3° .

a limiting factor of 1/19 (**=0.053**) of the linear distance covered.

★ Set the current **altitude change limit factor** based on the above selection.

2. Speed Limit

★ Set expected 'current' **speed limit maximum** — allowing an anticipated commonly occurring over speed $< \approx 30\%$.

Also load warnings for school zones, construction, incidents.

3. Direction Changes (yaw rate $\dot{\psi}$)

★ set 'normal' or standard **turn rate limit**, based on history or known driving style... typically < 0.2 rads/sec, more comfortably < 0.1 rads/sec [\[46\]](#)

' **ITERATION STEPS:**

1. Read GPS once per second.

2. Simultaneously read OBU CAN Bus (IMU) data.

3. Calculate changes Δ from prior step at time $t - 1$ to current time t

GPS Position Discard Threshold – Pseudocode (continued)

4. Process GPS data vector {latitude, longitude, altitude} for 'great circle' distance and heading using Haversine calculations.
5. Determine altitude change from prior step using **Equation 3.5**

$$\Delta \text{ Altitude} = \text{Altitude}_t - \text{Altitude}_{t-1} \quad (3.5)$$

6. Calculate haversine distance travelled using **Equation 3.6**.
(Use earth radius 6378000m to return distance in metres)

$$\Delta \text{ distance} = \text{distance}_t - \text{distance}_{t-1} \quad (3.6)$$

7. Calculate haversine heading, *hdg*, (measurement is in radians)
Convert $\Delta \text{ distance}$ from m/s to km/h using **Equation 3.7**

$$\text{vel} = 3.6 * \Delta \text{ distance} \quad (3.7)$$

$$\Delta \text{ heading} = \text{heading}_t - \text{heading}_{t-1} \quad (3.8)$$

' PROCESS:

1. **Altitude:**

Compare does $\Delta \text{ Altitude}$ exceed **altitude change limit factor** * $\Delta \text{ distance}$?
IF TRUE THEN discard position data, break from process

2. **Velocity:**

Is IMU velocity zero?
IF TRUE THEN discard position data, break from process
Does IMU velocity at prior step differ from GPS velocity by 5km/h ?
IF TRUE THEN discard position data, break from process

3. **Direction Travelled:**

Compare does $\Delta \text{ heading}$ exceed **turn rate limit**?
IF TRUE THEN discard position data, break from process

' NOTES:

' All Breaks from this procedure highlight enough measurable difference to believe that an Urban Canyon or other GPS position anomaly has been detected

' End Threshold Discard Method pseudocode



Experiment Results

4.1 Steps Completed

Preliminary data gathering using the Detroit DSRC testbed was done in October 2014 using a commercial Garmin GPS. The Detroit testbed contains seventeen ARADA Locomate Roadside Units (RSU's). The map of this Detroit testbed is shown in [Figure 4.1](#). Traffic on Larned is eastbound only, and Congress is also a one-way street westbound. Two driving tests were done to gather preliminary data (see [Figure 4.2](#) and [Figure 4.4](#)). GPS position data was plotted (in [Figure 4.3](#) and [Figure 4.5](#)) and used to understand these initial results. These results should be typical of any other urban canyon environment and would be easily reproduced anywhere that shows significant loss of direct satellite view.

Figure 4.1: ITS 2014 Detroit DSRC Urban Canyon Test Track

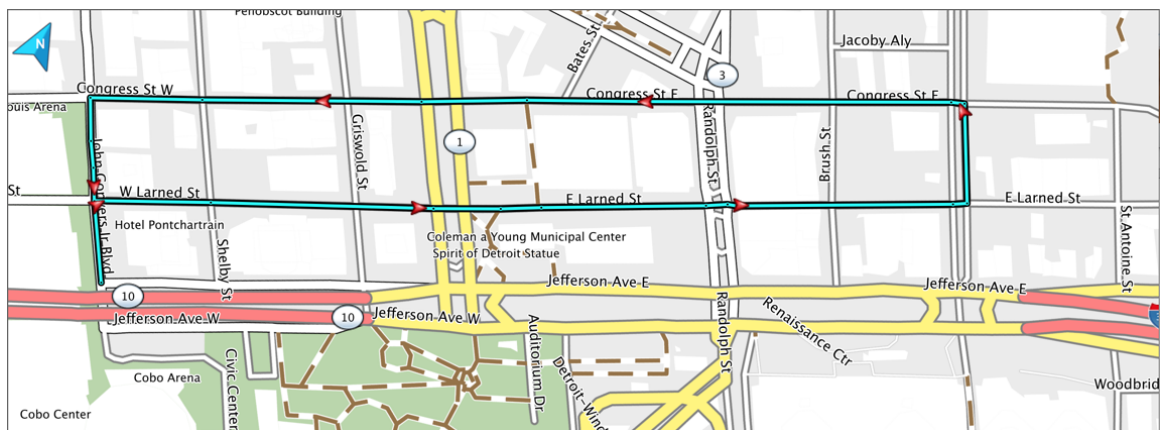


Figure 4.2: Test Run 1 - True Track

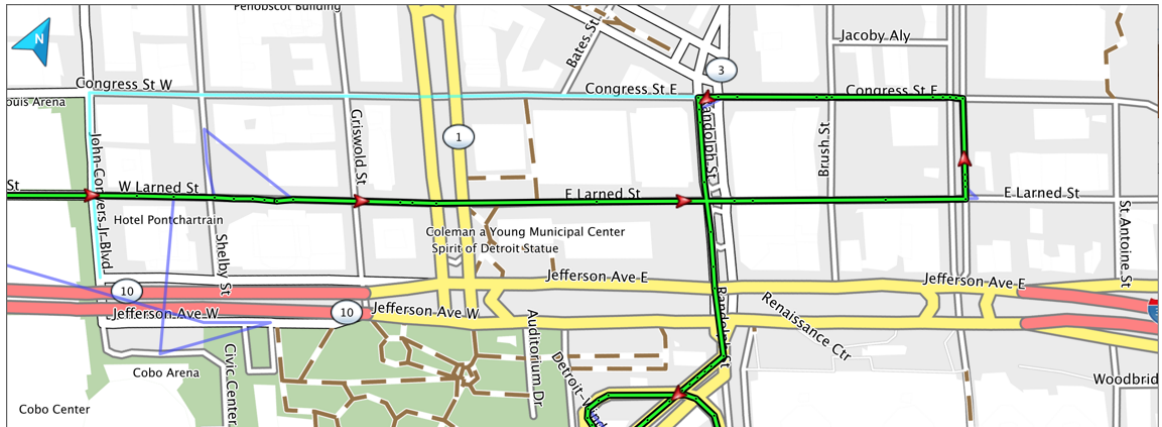


Figure 4.3: Test Run 1 - GPS Track showing Urban Canyon Issue

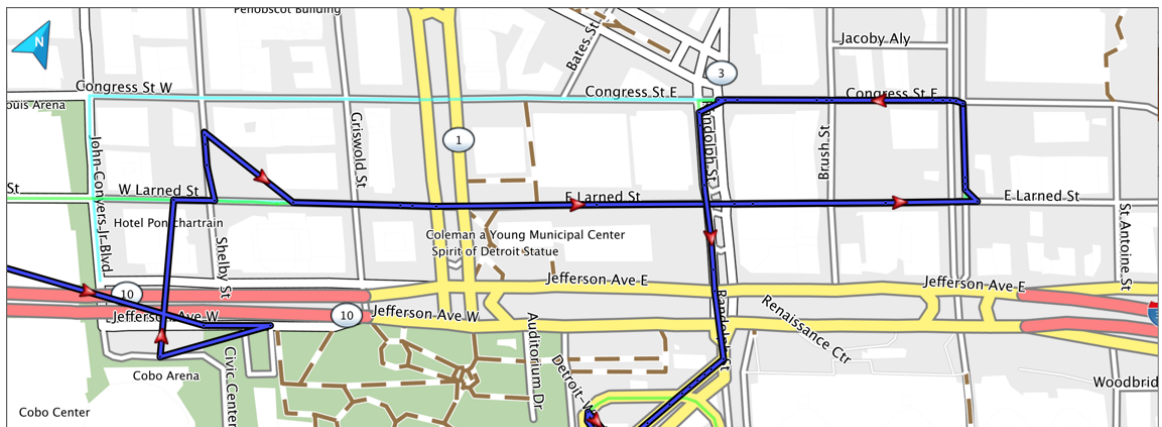
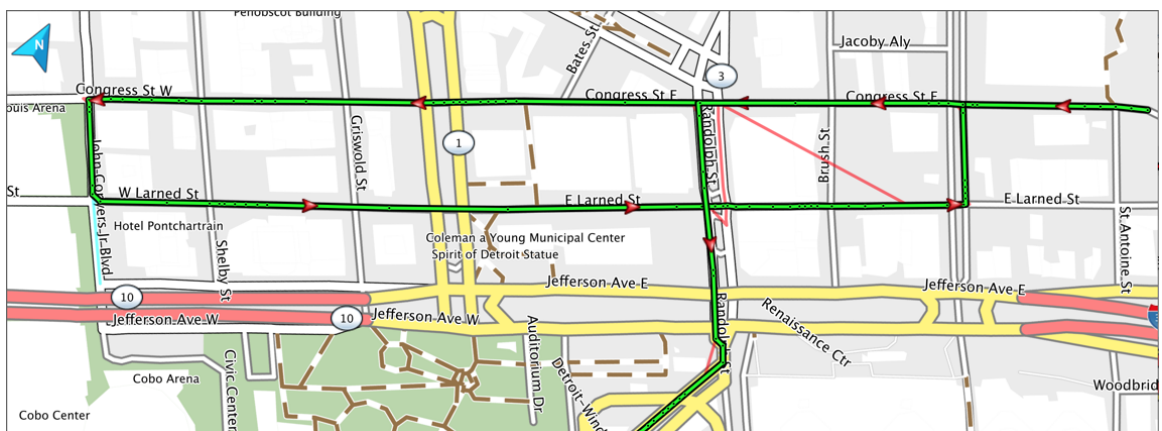


Figure 4.4: Test Run 2 - True Track



It was apparent that the GPS tracks were not aligning with the navigational ‘track made good’, showing trails appearing to jump. Prior results with the GPS in car map display never demonstrate these anomalies. The onboard GPS map display while the vehicle is in motion is not adjusting in this way – the track continues in a more or less ‘normal’ fashion, sometimes ‘shuddering’ briefly. When the detailed data capture is reviewed it shows a great deal of confusion, as can be seen in both the [Figure 4.3](#) and [Figure 4.5](#).

After seeing the initial (GPS only) problem was repeatable, a more reliable dataset was required. An ARADA systems vehicle, fitted with on-board units (OBU's) was run through the Detroit DSRC testbed. The data was aggregated and a Wireshark dissection of this data was completed. The data was transferred for further offline processing.

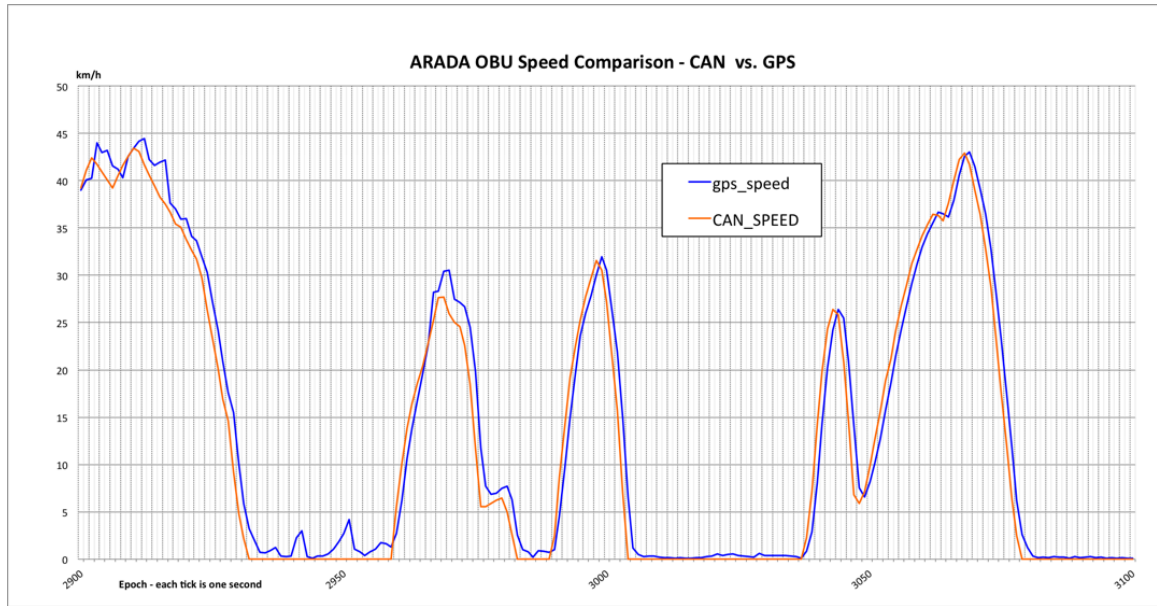
Real world data comes with ‘interesting’ variations. Velocity and acceleration are DSRC OBU inputs and I would have expected some better consistency in that sensor data. The velocity and acceleration data both appear highly variable. It is not trivial to adjust these to provide reference relative to geographic measurements. The sensors are set up to provide x, y, z axis information. These reference from the vehicle, x represents left (negative) and right (positive) turning, y represents forward (positive) and reverse (negative), z represents (positive) hill climb or (negative) descent. Motion terms are typically used to describe these as roll, pitch, and yaw. Roll would hopefully not happen in cars as it’s rotation centered on the y axis, which would involve tires not remaining in contact with the pavement. Pitch is rotation centered on the x axis, think of this as the nose up or down

orientation. Yaw is rotation around the z axis, which is how the vehicle normally turns.

Altitude information in the datasets was examined. It quickly became apparent that compared to [Table 2.6](#) on page 21 and it was noted that dramatic changes in Altitude were inevitable, and some drastic adjustments in latitude and longitude were also noted. These are visible in [Figure 4.7](#) on page 56.

One point of interest became clear originally from another researcher's thesis (Gaurav Sood) [47] who noted the GPS speed lagged behind the more direct readings from the CAN Bus sensors. I also charted the GPS velocity vs. CAN Bus velocity, and the results are shown in [Figure 4.6](#) on page 54. It can be clearly seen here that the GPS speed measurements lag roughly one second from the CAN Bus measurement. This is interesting as a comparative feature in the datasets to determine discrepancies. Since this hysteresis normally remains very constant at one second, we will also be able to use this as a key feature threshold.

Figure 4.6: **ARADA OBU Velocity Comparison CAN Bus vs. GPS**



4.2 Method

Passing GPS test data through Weka shows results which indicate a high correlation when used with specific filtering techniques. Various filters were applied, and ultimately a Ran-

dom Forest technique showed 99.88% correlation. Weka is an amazing tool for machine learning and statistical data mining, and this leads to an offline, after the fact data analysis. To be of practical in vehicle use we would need dynamic in vehicle processing, which means that Weka will not be the correct software tool.

Urban Canyon's have been demonstrated to have a significant effect on the accuracy of a vehicle determining its' own position based on satellite signals. By reviewing a capture of GPS track details in this environment, it became evident that a data driven method of detecting this error could be possible. Further, a method for predicting position could also be possible. Using well defined signal processing techniques on the given data we should have enough information to determine what position is possible.

4.3 Charts and Graphs

A diagram of GPS position data is shown and several areas of interest are discussed. Time scale measures from 2600 to 3600 representing 1001 seconds (16 minutes and 41 seconds) elapsed. Horizontal lines denote position unchanged over time with respect to each of these variables (latitude, longitude, and altitude) and are anticipated due to stops (traffic lights, etc.). The expectation is that each of the lines would be smooth with DSRC measuring data five times a second (capture rate $5Hz$), however that is the stated rate for DSRC BSM message transmissions. The OBU Data that we collected was only updating the GPS data at 1Hz (once per second). Vertical lines are only expected when data gaps occur or other errors have been observed.

I assigned an arbitrary four digit time stamp to correspond to each second in the Arada test, to make my comparison chart references and analysis easier. I began at step 1000, and progressed through the end of the drive test which was 4986, total trip duration one hour, six minutes and 42 seconds. The [Table 4.8](#) shows significant time stamps for comparison with all the figures [4.7](#) through [4.12](#) using the same time epochs.

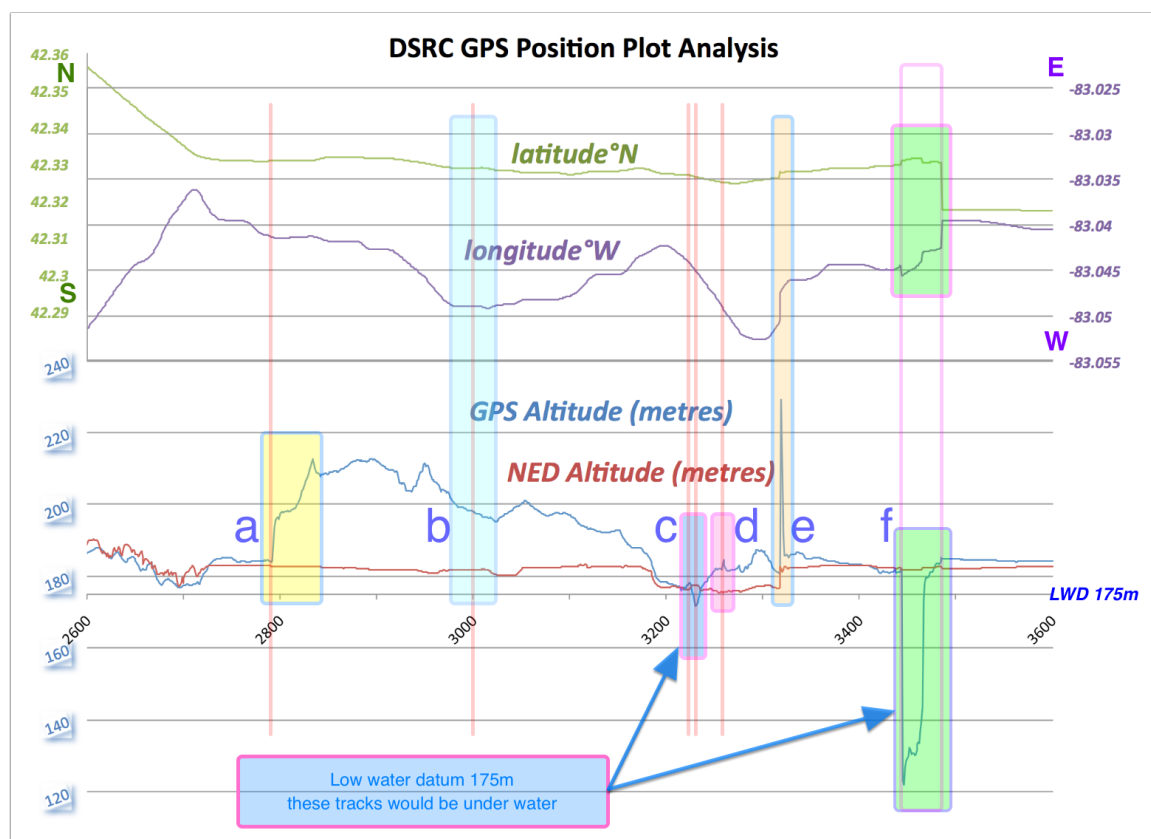
The Longitude scale (on right in magenta) uses negative numbers denoting degrees west (east is up).

Table 4.8: Epoch Cross-Reference for GPS Map Position Figures

Hdg.	Location - Interest	Epoch	Colour	Fig:
WB	Altitude spike 10m 2sec	a - 2792 - 2794	yellow	4.7
NB	Brush St. - traffic stopped	a - 2790 - 2832	yellow	4.8
WB	turning corner much too wide	b - 2972 - 3018	cyan	4.8
WB	Atwater under Hart Plaza	c - 3212 - 3230	light blue	4.8
WB	Atwater under Cobo	d - 3246 - 3259	pink	4.8
EB	US-10 S exit 1A under Cobo	e - 3313 - 3321	amber	4.8
EB	Larned–Urban Canyon path jump	f - 3443 - 3466	green	4.8
EB	EB Jefferson under Cobo exit 1A	E - 3318 - 3320	amber	4.9
EB	EKF improved	F - 3443	green	4.9
SB	SB Randolph–SKF mess	G - 3464		4.9

Epoch(1000) = utc(1434580013) which represents 2015-Jun-24 18:36:53 Eastern Time.
Heading abbreviations are West-bound (WB), North-(NB), East-(EB) and South-(SB)
Colour references chart area on [Figure 4.7](#)

Figure 4.7: DSRC GPS Position Analysis



Six areas are noted on this [Figure 4.7](#). The lettered areas correspond to the later figures in this section specifically figs: [4.8–4.12](#). The chart breaks out latitude (green

line above) and longitude (magenta line underneath) at the top, and the altitude separately (blue line) at the bottom. Another reference line for the National Elevation Database (NED Altitude) which is accurate to one arc-second ($\approx 30\text{m}$), has been included (dark red) to compare and contrast with the GPS Altitude. Several types of error situations are discussed as follows:

- (a) (altitude in yellow) North bound on Brush, the red reference line crosses latitude and longitude shown as smooth curves, with no disruption even when the altitude shows a dramatic jump. The jump observed here is (184.5 to 189.3) 4.8m in 200ms, (184.5 to 196.5) 12m in 1s (which is 43.2km/h upward velocity—clearly impossible), while the recorded vehicle surface velocity is 2.3m/s (8km/h)
- (b) (cyan) this area is the corner of Congress and Washington—which looks normal on this chart and becomes significant in later figures
- (c) (light blue) Hart Plaza tunnel. Detroit River Low Water Datum (LWD) at 175m provides a great reference threshold for impossibly low altitudes. Both areas (c) and (f) shown are not possible for DSRC vehicles since they must operate above water. For GPS underwater is considered a “degraded” environment. [32] Not shown is the upper threshold, for Detroit is the University District (elevation 204m) [Table 2.6](#) on page 21.
- (d) (pink) Cobo Hall–Atwater tunnel
- (e) (amber) The US-10 SB exit 1A tunnel to Jefferson, which clearly shows vertical lines in all three areas, latitude, longitude and Altitude. The altitude jumps from 180.6 to 229 (48.4m) in one step, and within one second is back into the 185m range. Longitude shows a change of 321m in a single step, where a local average around this shows $\approx 15\text{m}$. Latitude is least affected, shows a change of 179m in a single step, where the local average is $\approx 2\text{m}$.
- (f) (green) the dramatic effect of vertical jumps in altitude (below) help highlight the latitude and longitude areas that are also showing errors. Altitude changes (182 to 123.3) 58.7m in one step, and recovers over four seconds later to $\approx 179\text{m}$.

These dramatic spikes will allow us to flag these erroneous GNSS positions. This should be sufficient for noticing the problem is present, and allowing error correction to be invoked.

4.4 Position Maps

Several web resources worth mentioning that are very helpful for visualizing navigational details are Open Street Maps⁷, All Trails⁸, Thunderforest⁹ and of course Google Earth¹⁰. The OpenStreet Map site has quite a few features and uses crowd sourced information. It allows anyone to sign up, validates your account through email and allows you to add to the crowd sourced details on the map. Alltrails provides routes for hiking, walking and scenic trails, and a “Pro” option to create your own trail maps, push data to a smartphone and integrate with your GPS. Thunder Forest caters to people needing a way to integrate custom cartography and map hosting into their apps and web sites. Google Earth allows 3-D visualization, compatibility with every imaginable format of input and output, plus the local Google Earth application to install on your local machine.

I’ve made frequent use of the GPS Visualizer¹¹ website. This site integrates the most popular mapping backgrounds from Google, Open Street Maps, World Streets, Open Sea Maps, National Geographic and the USGS, with many backgrounds from several other providers ultimately featuring over fifty background types. I found this has better tools to visualize the maps that I’m interested in producing. It also has the least cluttered, most useable and straight forward design. Control over background opacity is given at 10% increments from 0 through 100%. Waypoints are fully controllable including interactive descriptions. Another great feature is that it will import many file formats and produce overlay maps with up to nine tracks by incorporating a layering concept. The site was designed using Perl scripts.

⁷<https://www.openstreetmap.org>

⁸<https://www.alltrails.com>

⁹<https://www.thunderforest.com>

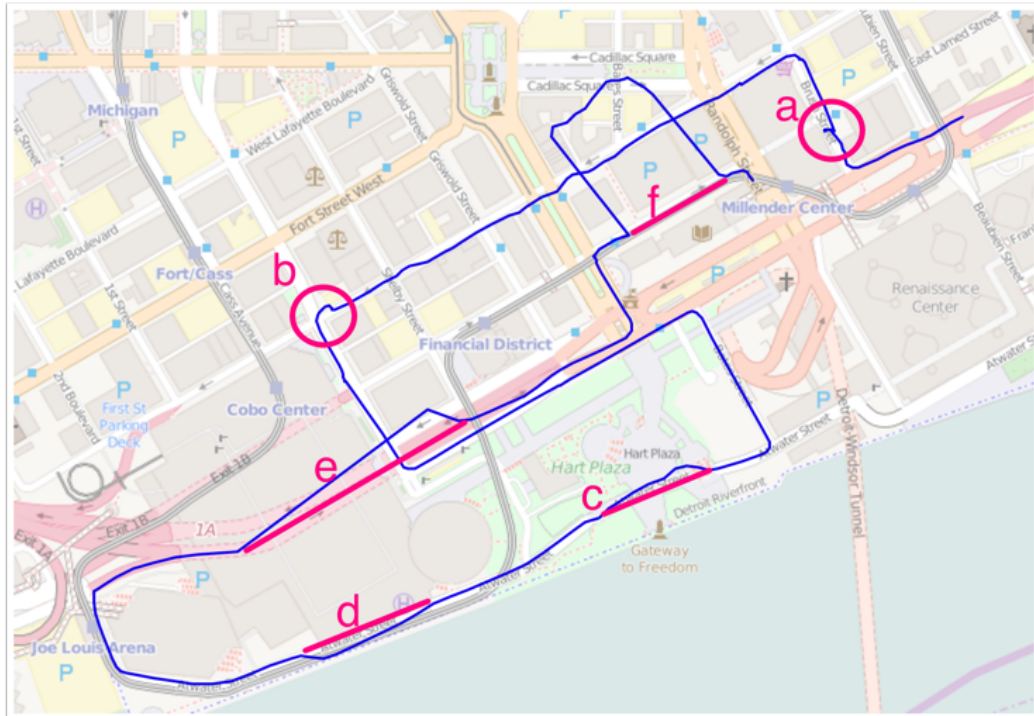
¹⁰<https://www.google.ca/intl/en/earth/>

¹¹<http://gpsvisualizer.com>

4.4.1 Arada Raw OBU Data

The original Garmin GPS raw data is shown above in Figure 4.3 and Figure 4.5. The Arada test data was taken from the Arada offices in Southfield, MI with a destination of University of Windsor. It includes a drive through the Detroit DSRC Testbed and surrounding area. I have truncated the data to focus on the specific urban canyon problem in the Detroit DSRC test area. The raw unprocessed GPS data collected from the Arada OBU is shown below:

Figure 4.8: Arada OBU Raw Nav Data



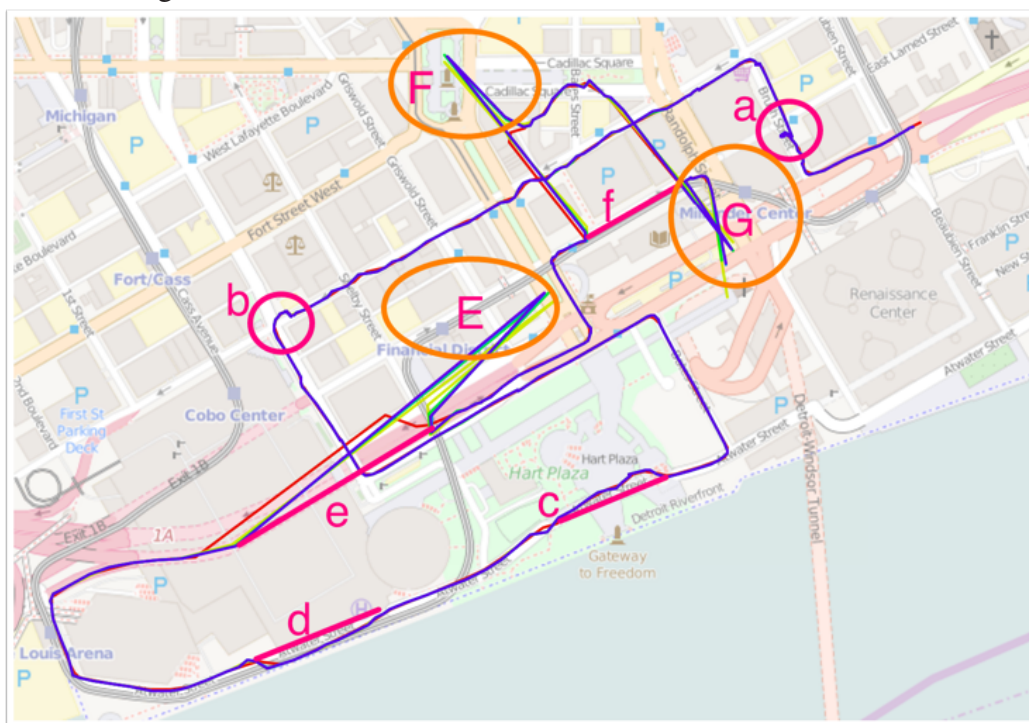
There are several areas of interest in this raw OBU navigation track. The direction of driving was entering from the east (right hand side) heading west-bound on Jefferson, turning north onto Brush. The areas marked on the next few figures (figs:4.8-4.12 match the chart in Figure 4.7 for comparison purposes. The first area of interest is area (a) which is during a brief traffic stop the GPS was already experiencing position difficulties only a few metres south of Larned-Brush. Heading down Congress there are quite a few interesting wavers to the GPS version of the course. At Congress-Washington (b) there is another odd mis-correct noted. Continuing south to Jefferson and south again on Bates, the Arada test was in full sky view and the resulting data for this section of the

driving test was quite good. Turning west-bound on Atwater, the vehicle once again encounters GPS shadow heading through the two tunnels (c) under Hart Plaza and (d) Cobo Hall. Emerging the track stabilizes heading around Joe Louis Arena, until the final drive through (e) the US-10 exit 1A onto Jefferson back east-bound. The trail does stabilize on Jefferson and follows along Woodward to Larned, and demonstrates the classic urban canyon (f) track displacement for 21 seconds before recovering. The Arada vehicle track was stopped at this point on Randolph before entering the Detroit-Windsor tunnel.

4.4.2 Arada OBU Data - Standard Kalman Filter Smoother

For the following diagrams, I've used upper case letters (E,F,G) to designate the areas containing wild position errors. Lower case letters (a,b) show small odd position errors. Lower case (c,d,e,f) lines designate the true path that was taken. The **GPS only** data was run through my standard Kalman Filter smoother python program and the trace results are shown below:

Figure 4.9: Arada Nav Data - Standard Kalman Smoother

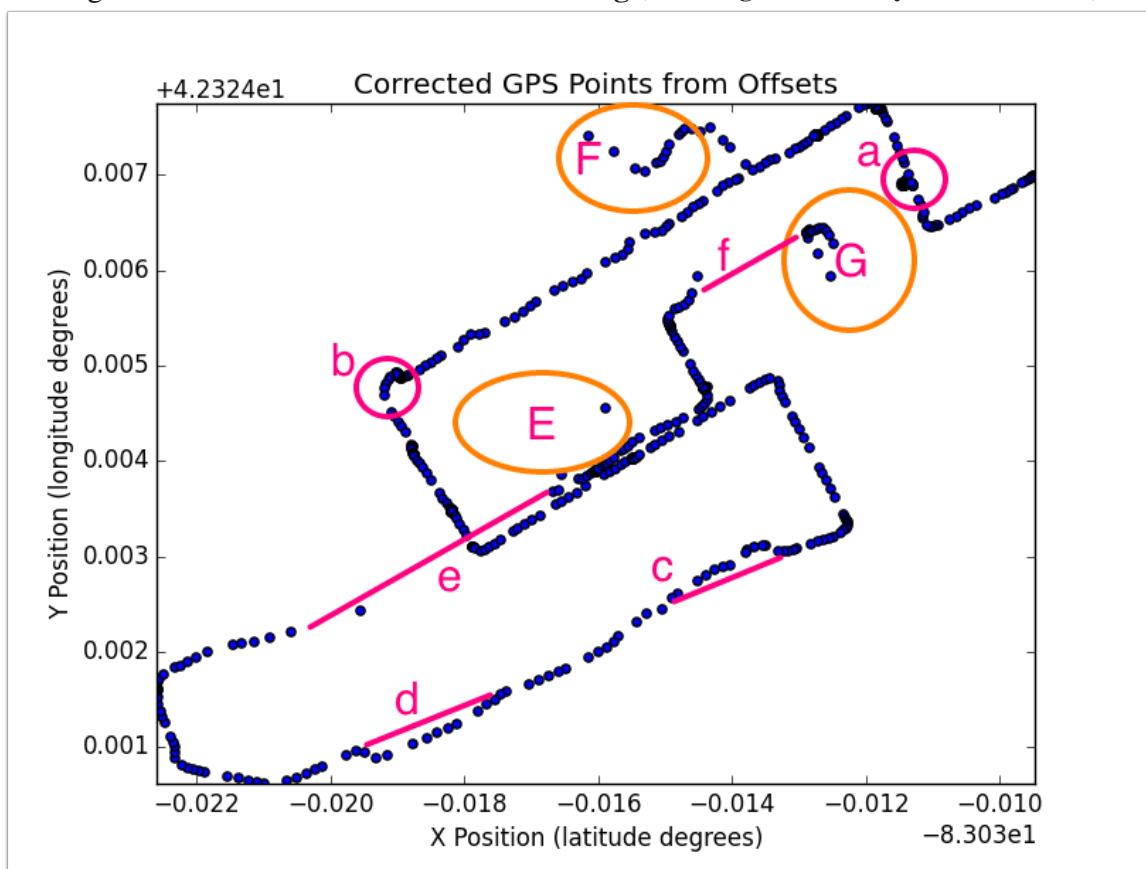


As visible in Figure 4.9 with this attempt using the state vector of (ϕ, λ, h) only our results were not very good. Significant areas of new problems occurred, these are

labelled as in capitals where (E), (F) and (G) experienced a much worse track than anticipated. The standard Kalman Filter relies on a bayesian normal distribution, and this is clearly not the case given these results.

Additional IMU data, including the x-, y-, and z- velocity and acceleration was available, and the state vector of $(\phi, \lambda, h, x\text{-vel}, y\text{-vel}, z\text{-vel}, x\text{-accel}, y\text{-accel}, z\text{-accel})$ was used through my standard Kalman filter smoother python program (which can be found in Appendix A on page 75) resulting in the following output:

Figure 4.10: **Standard KF after smoothing** (including IMU velocity & acceleration)



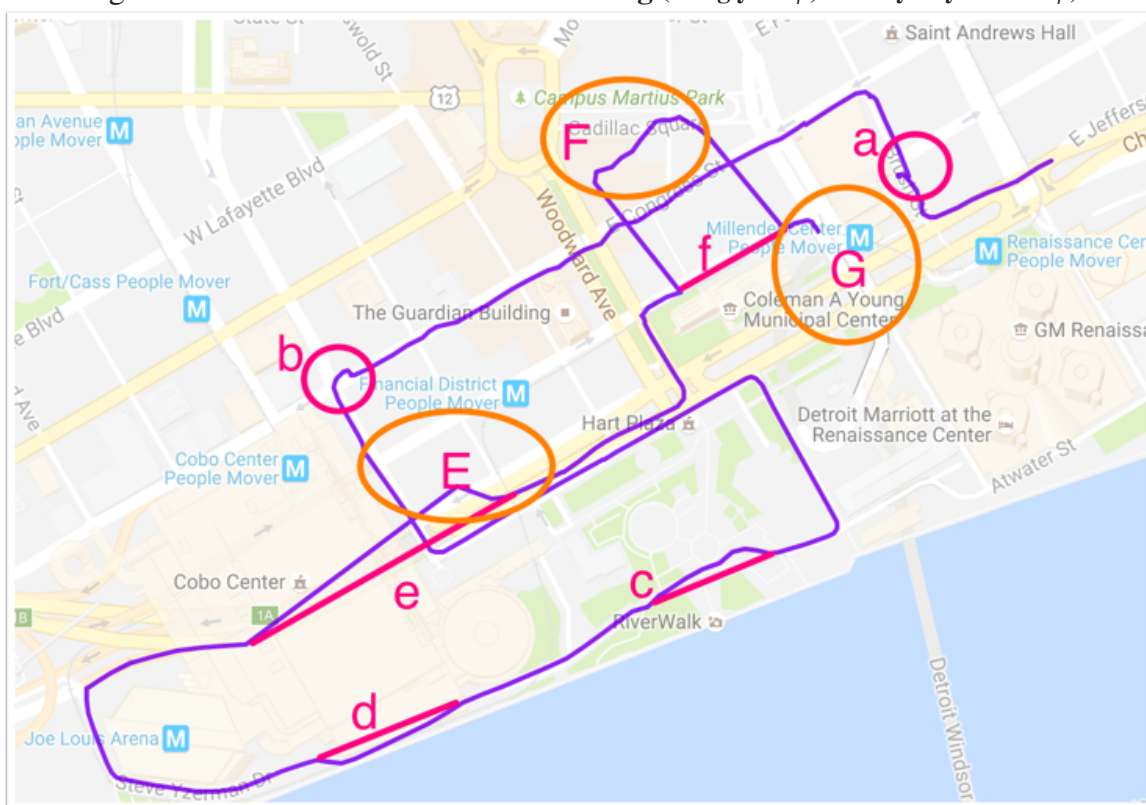
As visible in Figure 4.10 with this attempt using the full IMU data for our state vector of $(\phi, \lambda, h, x_{vel}, y_{vel}, z_{vel}, x_{accel}, y_{accel}, z_{accel})$, results were still not very good. Significantly areas labelled as in capitals where (E), (F) and (G) show only very limited improvement over the **GPS only** data. The areas of (a) where the GPS wandered although the vehicle was stopped, the *ab* track down Congress showed no improvement at all, and neither did the recovery from the exit 1A tunnel (*e* and *E*). It appears that the limitations

of the standard Kalman filter (SKF), which requires data under a normal bayesian distribution is clearly not able to deal with this non-linear input. Since the SKF is not enough to overcome these problems, and the other forms of KF must be explored.

4.4.3 Arada OBU Data - Extended Kalman Filter

The Extended Kalman Filter allows a non-Bayesian distribution to be handled. Our state vector of $(x, y, \psi, v, \dot{\psi})$ provided a bit of additional information, namely the yaw (ψ) and the yaw rate ($\dot{\psi}$). The effects of these adjustments are dramatic compared to the failure of the standard Kalman filter.

Figure 4.11: Extended KF after smoothing (using yaw ψ , velocity & yaw rate $\dot{\psi}$)

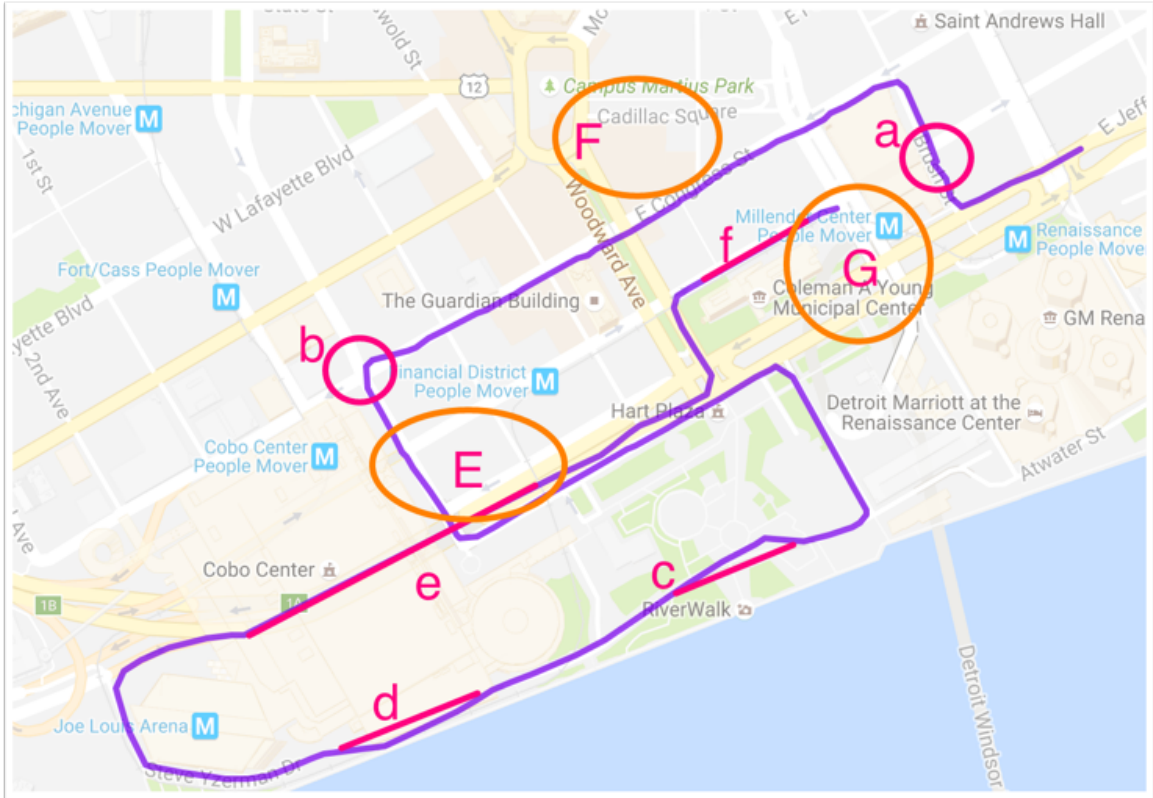


Areas noticed on the raw data in Figure 4.8 like (a), (b), (c), (d), and (e) were not affected by the EKF, but clear improvements were seen in the areas marked (E) and (G). Area (F), our classic urban canyon, is still unaffected.

4.5 Position Discard Threshold Method Results

After setting rules and applying the criteria to our testbed data, the results of using a threshold to discard erroneous data provides the following output:

Figure 4.12: Test Drive results after Position Discard Threshold method



It can be seen here that most of the really horrible problems from the previous KF examples have been eliminated. The wavering problems from GPS position floating (a) while the IMU indicates zero velocity are reduced. The oddly wide corner (b) at Congress and Washington is still visible although slightly reduced. Both Atwater street tunnels (c) and (d) still appear slightly off course. The very long (275m) exit 1A tunnel (e) to Jefferson has a significant gap between reported points, but a much smoother appearance than (E). And the wild urban canyon phenomena (F) that was originally present at on Larned between Woodward and Randolph has been eliminated (f), as well as the confusing tracks from (G).

While this does not particularly solve all the issues, it accomplishes getting rid of the gross errors introduced by the loss of satellite signals. The larger gaps between

markers highlight areas where data was discarded. These results should however provide a much better idea of the ultimate track followed, and with a small adjustment our IMU data could now be enough to assist in the main goal — sending far more accurate DSRC BSM messages, including a much more accurate idea of position.



Conclusions and Future Work

The Intelligent Transportation System is working towards the ultimate mobile device — your automobile. Connected Vehicles offer the promise of increased road safety for all travellers. Accident prevention tops the list of reasons to adopt this system. Location reporting shared through BSM broadcasts rely on highly accurate calculations of every vehicle's position. Urban roadways are often shadowed from the best position source currently available, which must have clear line of sight to accurately interpret GNSS signals.

Urban Canyon's, tunnels and other degraded GPS signal areas present a real challenge to accurate position reporting given our current operating conditions. Satellite signal transmissions have equivalent energy to a twenty-five watt light bulb shining from 20,000 kilometres altitude.¹² It's amazing that from such a low power output we can receive and correctly process these transmissions.

MEMS (micro-electro mechanical sensors) chips have made the costs affordable and the implementation easier, and all current OBU's are using these. IMU's have the reputation of 'drift' and will need frequent calibration to provide accuracy. GPS systems by design are reliant on the external satellite signals and although they work great for the majority of cases, it is critical for connected vehicles to have both systems in place to correct for the critical outages caused by Urban Canyon settings.

The GPS PDT method for path analysis has been developed and shown as a solution for flagging large-scale erroneous GNSS position calculations. This will be sufficient for noticing when large Urban Canyon position problems are present, and allowing error corrections to be invoked. GNSS position calculations provide latitude, longitude and altitude results. Altitude has proven to be most useful in getting a clear picture of where the problem exists — it appears that current mapping techniques focus on producing two-

¹²Expected signal strength for earth-based GPS receivers is $\approx 1.6 \times 10^{-16}$ watts [32]

dimensional (or surface based) maps, which drastically underestimates the importance of elevation in providing an accurate position.

5.1 Conclusions

Conventional GPS reliance on line-of-sight from receiver to satellite works well outside of so-called ‘degraded signal’ areas. Error mitigation expects proper acquisition and tracking of four or more satellite carrier PRN codes, and recovery after any signal tracking lock problem. Operation of connected vehicles in large urban canyon areas cannot provide accurate GPS positions without this initial signal lock-in, and will not be accurate in many urban areas that have significantly blocked sky views.

In the US, the FAA has been working on enhancing navigational systems for many years, and encouraged Loran use prior to the widespread availability of GPS. Their first enhanced GPS system was called WAAS, a **wide-area augmentation system**, essentially offering a differential GPS signal for all of north America. For further enhancement in accuracy their **local area augmentation system** (LAAS), recently renamed more descriptively as Ground Based Augmentation Systems (GBAS) will provide much finer accuracy (typically $<1\text{m}$). Fortunately there are equivalent systems to WAAS both in Europe (EGNOS) and Japan (MSAS).

Knowing that the OBU data can provide us with enough information to predict a GPS signal anomaly based on the threshold values determined in §4, we can interpolate positions during brief outages. What becomes more difficult is vehicles parking within these urban canyon areas will likely be blocked from establishing their initial GPS position fix. Our real world non-gaussian data directed our look at Kalman filters to the Extended (EKF) and Unscented (UKF), yet continued investigation using more input would be required to use these effectively.

Differential GPS offers a very practical solution to overcome the urban canyon issue. Surveys of these areas can be done without sophisticated equipment, such as I have done with the Garmin GPS receiver in the Detroit testbed area. You may recall from [section 3.9](#) that the six significant tunnels nearby the Detroit testbed are all GPS

shadow areas and special treatment will be necessary to accommodate these. In my opinion the best technical solution in the long term will be widespread DGPS rollout to cover tunnels and urban canyons. This could be done in or beside current selected RSU's during deployment. Until that can be accomplished our most practical solution for now appears to be using threshold conditions to discard erroneous GPS positions.

5.2 Possible Areas for Future Work

Exploring use of IMU data as a primary source for navigation should be a direction for the future. While IMU/INS units are well known to experience drift over time, it is a small enough measurable amount to be considered highly reliable, while using GPS data can help reset for this known drift factor. Position confidence is a major requirement going forward, some way of determining the magnitude of errors is imperative. GPS researchers are often quite happy to consider $<5\text{m}$ as 'accurate' — but standard highway lanes are 3m wide, and so clearly we need $<1\text{m}$ to provide safety of life services. Careful readers will recall that one nanosecond (10^{-9} seconds) is $\approx 30\text{cm}$, and so very small errors in satellite signal timing can really throw off positioning. Fortunately IMU and other APNT methods will be relatively simple calculations and well within the capacity of the current OBU computational power.

Python scripts are provided showing the standard kalman filter, and pseudo-code for applying my position threshold limits based method to discard erroneous GPS position reports. The way forward must include the ability to achieve the following:

- i) determine if the GPS signals can be trusted,
- ii) use integrated OBU (MEMS) sensors to determine accurate position when GPS is unreliable or unavailable
- iii) use path prediction,
- iv) economically integrate sensors for other systems beyond the GPS receiver.

The alternate position navigation and track (APNT) efforts currently being studied by the FAA [48] use an existing distance measuring equipment (DME) network and

provide a new data frame that very closely matches current GNSS satellite communications, while providing an alternative for times when satellites are not available. This "NextGen" technology is mandated for many aircraft by 2020, and planned adoption coverage reaches through North America and Europe. Connected vehicle research should be considering the extension of these APNT NextGen efforts as well as expansion of selected RSU's with integrated DGPS for the automotive segment of the Intelligent Transportation System.



Bibliography

- [1] Gibbons Media and LLC Research. Global Satellite Navigation Systems. <http://www.insidegnss.com/node/3265>, Accessed: 2015-02-18. [xvi, 11]
- [2] ITS America. Intelligent Transportation System. <http://itsa.org/aboutus/25-years-of-its>, Accessed: 2016-03-24. [xvii, 1]
- [3] Josep Aulinas, Yvan Petillot, Joaquim Salvi, and Xavier Lladó. The SLAM Problem: A Survey. In *Proceedings of the 2008 Conference on Artificial Intelligence Research and Development: Proceedings of the 11th International Conference of the Catalan Association for Artificial Intelligence*, pages 363–371, Amsterdam, The Netherlands, The Netherlands, 2008. IOS Press. [xviii, 5]
- [4] US DOT. US Department of Transportation. <https://www.transportation.gov>, Accessed: 2016-03-24. [1]
- [5] Jun-ichi Meguro, Taishi Murata, Jun-ichi Takiguchi, Yoshiharu Amano, and Takumi Hashizume. GPS Multipath Mitigation for Urban Area Using Omnidirectional Infrared Camera. *Intelligent Transportation Systems, IEEE Transactions on*, 10(1):22–30, 2009. [1, 2, 27]
- [6] Andrey Soloviev and Frank Van Graas. Utilizing Multipath Reflections in Deeply Integrated GPS/INS Architecture for Navigation in Urban Environments. In *Position, Location and Navigation Symposium, 2008 IEEE/ION*, pages 383–393. IEEE, 2008. [1, 2]
- [7] Sébastien Peyraud, David Bétaille, Stéphane Renault, Miguel Ortiz, Florian Mougél, Dominique Meizel, and François Peyret. About Non-Line-Of-Sight Satellite Detection and Exclusion in a 3D Map-Aided Localization Algorithm. *Sensors*, 13(1):829–847, 2013. [1]
- [8] François Peyret, David Betaille, Carolina Piñana-Díaz, Rafael Toledo Moreo, Antonio Gomez-Skarmeta, and Miguel Ortiz. GNSS Autonomous Localization: Non-Line-Of-Sight Satellite Detection Based on Digital Maps of City Environments. *IEEE ROBOTICS and AUTOMATION MAGAZINE*, 21(1), 2014. [1]
- [9] Kegen Yu and Y Jay Guo. Improved Positioning Algorithms for NonLine-of-Sight Environments. *Vehicular Technology, IEEE Transactions on*, 57(4):2342–2353, 2008. [1]

- [10] Paul D Groves, Ziyi Jiang, Morten Rudi, and Philip Strode. A Portfolio Approach to NLOS and Multipath Mitigation in Dense Urban Areas. *ION*, 2013. [1, 2, 10]
- [11] Paul D Groves and Ziyi Jiang. Height Aiding, C/N_0 Weighting and Consistency Checking for GNSS NLOS and Multipath Mitigation in Urban areas. *Journal of Navigation*, 66(05):653–669, 2013. [1, 2, 16]
- [12] Lei Wang, Paul D Groves, and Marek K Ziebart. GNSS Shadow Matching: Improving Urban Positioning Accuracy Using a 3D City Model with Optimized Visibility Scoring Scheme. *Navigation*, 60(3):195–207, 2013. [1, 10]
- [13] Dept. of Geography Penn.State. Ionospheric Scintillation. <https://www.e-education.psu.edu/geog160/node/1924>, Accessed: 2016-03-18. [1, 15]
- [14] ARADA Systems. LocoMate RoadSide Unit. <http://www.aradasystems.com/locomate-rsu/>, Accessed: 2016-03-24. [1]
- [15] Salman Syed and ME Cannon. Fuzzy Logic-Based Map Matching Algorithm for Vehicle Navigation System in Urban Canyons. In *ION National Technical Meeting, San Diego, CA*, volume 1, pages 26–28, 2004. [1, 10, 19]
- [16] Marcus Obst and Gerd Wanielik. Probabilistic Non-Line-Of-Sight Detection in Reliable Urban GNSS Vehicle Localization Based on an Empirical Sensor Model. In *Intelligent Vehicles Symposium (IV), 2013 IEEE*, pages 363–368. IEEE, 2013. [1]
- [17] Li-Ta Hsu, Shau-Shiun Jan, Paul D. Groves, and Nobuaki Kubo. Multipath Mitigation and NLOS Detection using Vector Tracking in Urban Environments. *GPS Solutions*, 19(2):249–262, 2015. [1, 2, 10]
- [18] Paul D Groves and Daniel C Long. Inertially-Aided GPS Signal Re-Acquisition in Poor Signal to Noise Environments and Tracking Maintenance Through Short Signal Outages. *Institute of Navigation GNSS*, 2005. [1, 2, 25, 26, 27]
- [19] Diana Cooksey. Understanding the Global Positioning System (GPS). *Online Report, Montana State University-Bozeman*, 2013. [3, 14]
- [20] Rudolph Emil Kalman. A New Approach to Linear Filtering and Prediction Problems. *Journal of Fluids Engineering*, 82(1):35–45, 1960. [4, 18, 19]
- [21] Alonzo Kelly. A 3D State Space Formulation of a Navigation Kalman Filter for Autonomous Vehicles. Technical report, DTIC Document, 1994. Defense Tech Info Center, Carnegie Mellon University, Report Number CMU-RI-TR-94-19. [5, 38]
- [22] Paul Balzer. Talk at PyData Berlin Conference. <https://github.com/balzer82/PyData-Berlin-2014-Kalman>, Accessed: 2015-12-14. [5]
- [23] Roger Labbe. Kalman and bayesian filters in python. <https://github.com/rllabbe/Kalman-and-Bayesian-Filters-in-Python>, 2014. [5]

- [24] Yuanxi Yang and Weiguang Gao. Comparison of Adaptive Factors in Kalman Filters on Navigation Results. *Journal of navigation*, 58(03):471–478, 2005. [5]
- [25] BL Malleswari, IV MuraliKrishna, K Lalkishore, M Seetha, and P Hegde Nagaratna. The Role of Kalman Filter in the Modelling of GPS Errors. *Journal of Theoretical & Applied Information Technology*, 5(1), 2009. [5, 18]
- [26] Rudolph Van Der Merwe and Eric A Wan. The Square-Root Unscented Kalman Filter for State and Parameter-Estimation. In *Acoustics, Speech, and Signal Processing, 2001. Proceedings.(ICASSP'01). 2001 IEEE International Conference on*, volume 6, pages 3461–3464. IEEE, 2001. [5, 42]
- [27] World Health Organization et al. *WHO Global Status Report on Road Safety 2013: Supporting A Decade of Action*. World Health Organization, 2013. [6]
- [28] John B Kenney. Dedicated Short-Range Communications (DSRC) Standards in the United States. *Proceedings of the IEEE*, 99(7):1162–1182, 2011. [7]
- [29] US DOT. Dedicated Short Range Communications. http://www.its.dot.gov/factsheets/pdf/JP0-034_DSRC.pdf, Accessed: 2015-05-16. [7]
- [30] Sunny Jacob, Prasad Nannapaneni, Colleen Hill-Stramsak, Beata Lamparski, and Greg Krueger. Detroit Builds First Urban Canyon Connected Vehicle Test Bed. http://itswc.conferencespot.org/polopoly_fs/1.1512254.1416424806!/fileserver/file/308191/filename/13549.pdf, Accessed: 2014-11-24. [10, 30, 32, 35, 44]
- [31] Vincent Drevelle and Philippe Bonnifait. Global positioning in urban areas with 3-d maps. In *Intelligent Vehicles Symposium (IV), 2011 IEEE*, pages 764–769. IEEE, 2011. [10]
- [32] George T. Schmidt. Navigation Sensors and Systems in GNSS Degraded and Denied Environments. *Chinese Journal of Aeronautics*, 28(1):1–10, 2015. [15, 37, 57, 65]
- [33] Stanford University Space Based Augmentation Systems, Ionospheric Working Group. Effect of Ionospheric Scintillations on GNSS? A White Paper. http://waas.stanford.edu/papers/IWG/sbas_iono_scintillations_white_paper.pdf, 11 2010. [15, 16]
- [34] Jonathan S Abel and James W Chaffee. Existence and Uniqueness of GPS Solutions. *Aerospace and Electronic Systems, IEEE Transactions on*, 27(6):952–956, 1991. [15]
- [35] Bureau of Meteorology Commonwealth of Australia. What is Ionospheric Scintillation? <http://www.sws.bom.gov.au/Satellite/6/3>, Accessed: 2016-03-18. [16]
- [36] NOAA. Ionospheric Scintillation. <http://www.swpc.noaa.gov/phenomena/ionospheric-scintillation>, Accessed: 2016-03-18. [16]

- [37] Bryan R Townsend, Patrick C Fenton, Keith J Van Dierendonck, and DJ Richard van Nee. Performance Evaluation of the Multipath Estimating Delay Lock Loop. *Navigation*, 42(3):502–514, 1995. [22]
- [38] Mark L Psiaki. Smoother-Based GPS Signal Tracking in a Software Receiver. In *Proceedings of ION GPS*, volume 2001, pages 2900–2913, 2001. [23, 24, 25]
- [39] Jasurbek Khodjaev, Yongwan Park, and Aamir Saeed Malik. Survey of NLOS Identification and Error Mitigation Problems in UWB-based Positioning Algorithms for Dense Environments. *annals of telecommunications-Annales des télécommunications*, 65(5-6):301–311, 2010. [28, 29]
- [40] Garmin. What is GPS? <http://www.garmin.com/aboutGPS/>, Accessed: 2015-04-22. [31]
- [41] Sean Bednarz and Pratap Misra. Receiver Clock-Based Integrity Monitoring for GPS Precision Approaches. *Aerospace and Electronic Systems, IEEE Transactions on*, 42(2):636–643, 2006. [31]
- [42] ARADA Systems. LocoMate On-Board Unit. http://www.aradasystems.com/wp-content/uploads/2015/06/Arada_datasheet_obu_2.01_2015.pdf, Accessed: 2015-07-14. [5, 31, 37, 40]
- [43] IEEE Standard for Information Technology– Local And Metropolitan Area Networks– Specific Requirements– Part 11: Wireless LAN Medium Access Control (MAC) And Physical Layer (PHY) Specifications Amendment 6: Wireless Access In Vehicular Environments. IEEE Std 802.11p-2010 (Amendment to IEEE Std 802.11-2007 as amended by IEEE Std 802.11k-2008, IEEE Std 802.11r-2008, IEEE Std 802.11y-2008, IEEE Std 802.11n-2009, and IEEE Std 802.11w-2009), pages 1–51, July 2010. [39]
- [44] IEEE Standard for Wireless Access in Vehicular Environments Security Services for Applications and Management Messages. *IEEE Std 1609.2-2013 (Revision of IEEE Std 1609.2-2006)*, pages 1–289, April 2013. [39]
- [45] Daniel Jiang and Luca Delgrossi. IEEE 802.11 p: Towards an International Standard for Wireless Access in Vehicular Environments. In *Vehicular Technology Conference, 2008. VTC Spring 2008. IEEE*, pages 2036–2040. IEEE, 2008. [39]
- [46] Motoki Shino and Masao Nagai. Yaw-moment control of electric vehicle for improving handling and stability. *JSAE review*, 22(4):473–480, 2001. [47, 49]
- [47] Gaurav Sood. Architecture for Extracting Data from Vehicular Sensors. Master’s thesis, University of Windsor, 401 Sunset Avenue Windsor, Ontario, Canada N9B 3P4, 8 2015. [54]
- [48] Eldredge, Leo and Enge, Per and Harrison, Mike and Kenagy, Randy and Lo, Sherman and Loh, Robert and Lilly, Robert and Narins, Mitch and Niles, Rick. Alternative Positioning, Navigation & Timing (PNT) Study. In *International Civil Aviation*

Organisation Navigation Systems Panel (NSP), Working Group Meetings, Montreal, Canada. Citeseer, 2010. [67]

Appendices

Python Kalman Smoothing

```
1  #!/usr/local/bin/python
2  # coding: latin-1
3  '''
4  # <codecell>      pykal5.py      Kalman Filter for GPS trail (path trace)
   smoothing
5  #
6  #      Ian Douglas      forked 2015–August updated through 2016–August
7  '''
8
9  from pykalman import KalmanFilter
10 import matplotlib.pyplot as plt
11 import numpy as np
12 import random
13 from mpl_toolkits.mplot3d import Axes3D
14 from haversine import haversine
15 from pyproj import Proj
16 from os import mkdir
17
18 '''
19 #      Haversine computes great circle distance from latitude and
   longitude angles
20 #      = 1/2 * versine = (1 - cos(Δσ))/2 or sin²(Δσ/2)
21 '''
22 def do_hav(source_lat,source_lng,dest_lat,dest_lng):
23     return haversine((source_lat,source_lng),(dest_lat,dest_lng),miles =
        True)
24 #      END do_hav
25
26 '''
27 #      Convert [latitude, longitude, altitude] to UTM cartesian values
28 #      GPS uses WGS84 standard
29 '''
30 def gps_to_cartesian(coordinates,inverted=False):
31     p = Proj(proj='utm',zone=17,ellps='WGS84')
32     converted = []
33     for point in coordinates:
34         lat = point[0]
35         lng = point[1]
36         alt = point[2]
37
38         if inverted:
39             x,y = p(lng,lat,inverse=True)
40         else:
41             x,y = p(lng,lat)
42
43         converted.append((y,x,alt))
44
45     return converted
46 #      END gps_to_cartesian
47
48 def calculate_point_variance(measurements,corrected,absolute=True):
```

```

49     if len(measurements) != len(corrected):
50         print "* 50 * calculate_point_variance: Lists must be same
51             size", len(measurements), len(corrected)
52         exit(1)
53
54     variance = []
55     for i in range(len(measurements)):
56         if absolute:
57             variance.append((abs(measurements[i][0] - corrected[i][0]),
58                                 abs(measurements[i][1] - corrected[i][1]),
59                                 abs(measurements[i][2] - corrected[i][2])))
60         else:
61             variance.append((measurements[i][0] - corrected[i][0],
62                             measurements[i][1] - corrected[i][1],
63                             measurements[i][2] - corrected[i][2]))
64     return variance
65 # END calculate_point_variance
66
67 def calculate_offsets(measurements, cartesian=False):
68     offsets = []
69     for i in range(len(measurements)):
70         if i == 0:
71             continue
72         offsets.append(
73             (measurements[i][0] - measurements[i-1][0],
74              measurements[i][1] - measurements[i-1][1],
75              measurements[i][2] - measurements[i-1][2]))
76     return offsets
77 # END calculate_offsets
78
79 def positions_from_offsets_additive(ref, offsets):
80     positions = []
81     current_pos = list(ref)
82     positions.append(current_pos)
83     for offset in offsets:
84         for i in range(3):
85             current_pos[i] += offset[i]
86         positions.append(tuple(current_pos))
87     return positions
88 # END positions_from_offsets_additive
89
90 def positions_from_offsets_corrective(refs, offsets):
91     if not (len(refs) == len(offsets)):
92         print "ERROR 96: positions from offsets not matching in length.
93             %d vs. %d" % (len(refs), len(offsets))
94         exit(1)
95     positions = []

```

```

103     for j in range(len(refs)):
104         current_pos = list(refs[j])
105         for i in range(3):
106             current_pos[i] += offsets[j][i]
107         positions.append(tuple(current_pos))
108
109     return positions
110 # END positions_from_offsets_corrective
111 '''
112 # measurements must be in the format
113 # [ (x, y, z), ... ]
114 '''
115 def run_kal(measurements, training_size=60):
116     ' count the number of measurements
117
118     num_measurements = len(measurements)
119     ' find the dimension of each row
120
121     dim = len(measurements[0])
122
123     if dim == 3:
124         simple_mode = True
125     elif dim == 9:
126         simple_mode = False
127     else:
128         print "Error: Dimensions for run_kal must be 3 or 9"
129         exit(1)
130     print("run_kal 127 - run_kal smooth —> simple_mode={0}".format(
131         simple_mode))
132
133     training_set = []
134     for i in range(min(training_size, len(measurements))):
135         training_set.append(measurements[i])
136
137     if simple_mode:
138         ' run the filter
139
140         kf = KalmanFilter(initial_state_mean=[0,0,0], n_dim_obs=3)
141         (smoothed_state_means, smoothed_state_covariances) = kf.em(
142             training_set).smooth(measurements)
143     else:
144         kf = KalmanFilter(initial_state_mean=[0,0,0,0,0,0,0,0,0],
145             n_dim_obs=9)
146         (smoothed_state_means, smoothed_state_covariances) = kf.em(
147             training_set).smooth(measurements)
148
149     ' means represent corrected points
150
151     return smoothed_state_means, smoothed_state_covariances, simple_mode
152 # END run_kal
153 '''
154 # Process input [latitude, longitude, altitude] vehicle path

```

```

151 '''
154 def load_gps_trail(extra_fields = False):
155     i = 0
156     measurements = []
157     header=True
158     print("load gps trail 152 – Failure here means the input gps file
        did not have correct Unix LF endings...")
159     for line in open('threshold-discard-no-epoch.csv'):          #'
        nazeer_trip.csv'):
160         if header:
161             header=False
162             continue
163         i += 1
164         fields = line.strip().split(',')
165
166         if i % 6 == 0:      # Display every sixth entry, plus first and
            100th
167             print i, fields
168             continue
169         elif i == 1:
170             print i, fields, '\n Length of line = ', len(fields), "
                extra_fields = ", extra_fields
171             continue
172         elif i == 100:
173             print i, fields, "\n extra_fields = ", extra_fields #,
                measurements
174             continue
175
176 #     print "Past the 6 – 1 and 100 clause", i, fields
177     px = float(fields[0])
178     py = float(fields[1])
179     pz = float(fields[2])
180
181     if len(fields) == 3:
182         extra_fields = False
183         measurements.append((px, py, pz))
184         continue
185     else:
186         extra_fields = True
187         px_vel = float(fields[3])
188         py_vel = float(fields[4])
189         pz_vel = float(fields[5])
190         px_accel = float(fields[6])
191         py_accel = float(fields[7])
192         pz_accel = float(fields[8])
193         measurements.append((px, py, pz, px_vel, py_vel, pz_vel,
            px_accel, py_accel, pz_accel))
194         continue
195
196     return measurements
197 #     END load_gps_trail
198
199 '''
200 #     Plot [latitude , longitude , altitude ]

```

```

201 #    (need to flip lat & long to get x & y looking right on the output
202 ', ',
203
204 def plot_first_dim(data, flip_xy=True, xlabel="", ylabel="", title="", filename=
None):
205     x_axis = []
206     y_axis = []
207
208
209     if flip_xy:
210         minx = maxx = data[0][1]
211         miny = maxy = data[0][0]
212     else:
213         minx = maxx = data[0][0]
214         miny = maxy = data[0][1]
215
216     print "plot_first_dim 208 — length data = ", len(data)
217
218     for x,y,z, in data:
219
220         if flip_xy:
221             t = x
222             x = y
223             y = t
224
225         x_axis.append(x)
226         y_axis.append(y)
227
228         minx = min(x, minx)
229         miny = min(y, miny)
230
231         maxx = max(x, maxx)
232         maxy = max(y, maxy)
233
234     plt.xlim((minx, maxx))
235     plt.ylim((miny, maxy))
236     plt.xlabel(xlabel)
237     plt.ylabel(ylabel)
238     plt.title(title)
239     plt.scatter(x_axis, y_axis)
240
241     if filename:
242         plt.savefig(filename)
243     else:
244         plt.show()
245
246     plt.close()
247 # END plot_first_dim
248
249 def plot_all_dimensions(positions, file_prefix):
250     split = [list(), list(), list()]
251
252     j = 0
253     for position in positions:
254         for i in range(3):
255             split[i].append((j, position[i], 0))

```

```

256         j+=1
259
260     x_vals = split[0]
261     y_vals = split[1]
262     z_vals = split[2]
263
264     plot_first_dim(x_vals, flip_xy=False, xlabel="Time (s)", ylabel="WGS84
        Latitude", title="WGS84 LATITUDE OVER TIME", filename=output_dir +
        file_prefix + "all_dim_x")
265     plot_first_dim(y_vals, flip_xy=False, xlabel="Time (s)", ylabel="WGS84
        Longitude", title="WGS84 LONGITUDE OVER TIME", filename=output_dir
        + file_prefix + "all_dim_y")
266     plot_first_dim(z_vals, flip_xy=False, xlabel="Time (s)", ylabel="Altitude "
        , title="Altitude OVER TIME", filename=output_dir + file_prefix + "
        all_dim_z")
267 # END plot_all_dimensions
268
269 '''
270 ## MAIN Program here
271 '''
272 output_dir = "output5/"
273
274 try:
275     mkdir(output_dir)
276 except OSError, e:
277     pass
278
279 #EXTRA
280 parse_extra_fields = True
281
282 # Load the path in from the file
283 measurements = load_gps_trail(extra_fields = parse_extra_fields )
284 print "Main 274 - measurements ", measurements, "\n274
        parse_extra_fields = ", parse_extra_fields
285
286 # Convert these measurements to meters
287 cart_measurements = gps_to_cartesian(measurements)
288 print "Main 278 - cart_measurements ", cart_measurements
289
290 #####
291 ## Positions ##
292 #####
293
294 # Run the Kalman filter on the measurements
295 print "Main 285 - Run KF - measurements", measurements
296 ss_means_meas, ss_covariance_meas, cart_simple = run_kal(measurements,
        training_size=60)
297 filtered_measurements = ss_means_meas # from smoothed_state_means
298 print "Main 288 - Run KF - filtered_measurements ", len(
        filtered_measurements)
299 f = open(output_dir + 'skf_trail_filtered.csv', 'w')
300 f.write('lat,lng,alt,xdot,ydot,zdot,xddot,yddot,zddot\n')
301
302 #if cart_simple == True:

```

```

303 for lat,lng,alt in filtered_measurements:
304     f.write("{0},{1},{2}\n".format(lat,lng,alt))
305 #else:
306 #     for lat,lng,alt,xdot,ydot,zdot,xddot,yddot,zddot in
307         filtered_measurements:
308 #         f.write("{0},{1},{2},{3},{4},{5},{6},{7},{8}\n".format(lat,lng,
309             alt,xdot,ydot,zdot,xddot,yddot,zddot))
310 f.close()
311
312 print "Main 299"
313 #####
314 ## OFFSETS ##
315 #####
316
317 # Find the offsets between points
318 offsets = calculate_offsets(measurements)
319
320 # Run the Kalman filter on the offsets
321 ss_means, ss_covariance, offset_simple = run_kal(offsets)
322 corrected_offsets = ss_means # from smoothed_state_means
323
324 # Reconstruct the cartesian positions from the offsets and a reference
325     point
326 #corrected_cart_positions = positions_from_offsets_additive(
327     cart_measurements[0],corrected_offsets)
328 corrected_positions = positions_from_offsets_corrective(measurements[1:
329     len(measurements)], corrected_offsets)
330
331 # Convert the positions back to wgs84
332 corrected_gps_positions = gps_to_cartesian(corrected_positions,inverted=
333     True)
334
335 ### OFFSET Results ##
336 print "Main 319 - plot_first_dim - measurements[:3] "
337 plot_first_dim(
338     measurements[:3],
339     title = "Measurements in WGS84 - Lat/Long/alt",
340     xlabel = 'X Position (latitude degrees)',
341     ylabel = 'Y Position (longitude degrees)',
342     filename = output_dir + "measurements.png" )
343
344 print "Main 327 - plot_first_dim - corrected_positions "
345 plot_first_dim(
346     corrected_positions,
347     title = "Corrected GPS Points from Offsets ",
348     xlabel = 'X Position (latitude degrees)',
349     ylabel = 'Y Position (longitude degrees)',
350     filename = output_dir + "corrected_fromoff.png")
351 #     filename = output_dir + "filtered_fromoff.png")
352
353 print "Main 336 - plot_first_dim - filtered_measurements "
354 plot_first_dim(
355     filtered_measurements,
356     title = "Corrected WGS84 Points from Positions",

```

```

353     xlabel = 'X Position (latitude degrees)',
354     ylabel = 'Y Position (longitude degrees)',
355     filename = output_dir + "filtered_frompos.png")
356
357 print "Main 344 — length corrected_positions = ", len(
358     corrected_positions), " length measurements = ", len(measurements)
359 plot_first_dim(corrected_positions,"fromFilteredOffset—")
360 plot_first_dim(measurements,"fromMeasurements—")
361
362 # Split the variance into it's three components and draw it
363
364 # Calculate the variance between the two offset lists
365 offset_variance = calculate_point_variance(measurements[1:],
366     corrected_positions)
367
368 offset_variance_split = [[],[],[ ]]
369 for x,y,z in offset_variance:
370     offset_variance_split[0].append(x)
371     offset_variance_split[1].append(y)
372     offset_variance_split[2].append(z)
373
374 plt.plot(offset_variance_split[0])
375 plt.plot(offset_variance_split[1])
376 plt.plot(offset_variance_split[2])
377 plt.xlabel('Measurement ID')
378 plt.title('Variance Between Measurement and Filtered (Offsets)')
379 plt.ylabel('Measurement offset_variance')
380 #plt.show()
381 plt.savefig(output_dir + "offset_variance.png")
382
383 # calculate variance for positions
384 position_variance = calculate_point_variance(measurements,
385     filtered_measurements)
386 position_variance_split = [[],[],[ ]]
387 for x,y,z in position_variance:
388     position_variance_split[0].append(x)
389     position_variance_split[1].append(y)
390     position_variance_split[2].append(z)
391
392 plt.plot(position_variance_split[0])
393 plt.plot(position_variance_split[1])
394 plt.plot(position_variance_split[2])
395 plt.xlabel('Measurement ID')
396 plt.title('Variance Between Measurement and Filtered (Positions)')
397 plt.ylabel('Measurement offset_variance')
398 #plt.show()
399 plt.savefig(output_dir + "point_variance.png")
400
401 # output corrected file
402
403 f = open(output_dir + 'gps_trail_filtered.csv', 'w')
404 f.write('lat ,lng ,alt\n')
405 i = 0
406 for x,y,z in corrected_gps_positions:

```

```

406     i += 1
409     f.write("{0}, {1}, {2}\n".format(x,y,z))
410     if i % 6 == 0:      # Display every sixth entry, plus first and 100th
411         print i,x,y,z
412         continue
413     elif i == 1:
414         print i,x,y,z, '\n Length of line = ', len(
            corrected_gps_positions)
415         continue
416     elif i == 100:
417         print i,x,y,z #, "\n extra_fields = ", extra_fields #,
            measurements
418         continue
419
420 f.close()
421
422 '''
423 ##    GPS Trail input and standard Kalman filter smoothing...
424 ##
425 ##    End <pykal5.py>
426 '''

```

Weka Random Forest

=== Run information ===

```
Scheme:      weka.classifiers.trees.RandomForest -I 100 -K 0 -S 1 -num-slots 1
Relation:    DSRC_ALL_FIELDS
Instances:   4920
Attributes:  19
             utc_time
             lat
             lng
             gps_speed
             CAN_SPEED
             AccelPedalPosition
             AmbientTemp
             BrakeSwitch
             BrkSw2Stat
             ESP_BrakeSwitch
             BrkSwOnStat
             YawRate
             SteeringWheel_Angle
             Odometer
             BATT_VOLT
             LAT_ACCEL
             EngineSpeed
             alt
             Altitude-nominal
Test mode:   split 66.0% train, remainder test
```

=== Classifier model (full training set) ===

Random forest of 100 trees, each constructed while considering 5 random features.
Out of bag error: 0.1381

Time taken to build model: 1.62 seconds

=== Evaluation on test split ===

Time taken to test model on training split: 0.09 seconds

=== Summary ===

Correlation coefficient	0.9988
Mean absolute error	0.1568
Root mean squared error	0.5056
Relative absolute error	2.3326 %
Root relative squared error	5.1409 %
Total Number of Instances	1673

Arada OBU Data Fields

ARADA OBU Data Fields - highlighted fields denote duplicates

utc_time	lat	lng
alt	YawRate	speed
VEH_SPEED	AccelPedalPosition	BrakeSwitch
FrontWiperInUse	ESP_Lamp_Req	BrkEnbl_LCM
FullBrk_Actv	AccelPedalPosition	ABS_Lamp_Req
SteeringWheel_Angle	ETC_LmpFlash	ESP_TrqRqEnabl
ESP_PD_EN	Odometer	REF_VEH_SPEED
BarometricPressure	ABS_BrkEvt	EngineSpeed
ESP_AVL	BRK_ACT_LCM	ABS_PRSNT
BRK_ACTIVE	BrkSw2Stat	HighBeams
ESP_PRSNT	BarometricPressure	ESP_TrqRqEnabl
ESP_PD_EN	FullBrk_Actv1	TSC_SUPP
ESP_AVL	PARK_BRK_EGD	ESP_BrakeSwitch
TSC_SUPP	PrkBrake_Indicator	BRK_ACT_LCM
AmbientTemp	VEH_SPEED	BrakeSwitch
FullBrk_Actv1	REF_VEH_SPEED	LowBeam
ETC_LmpOn	BRK_ACTIVE	TSC_EN
utc_time	lat	ESP_Disabled
PrkBrake_Indicator	ABS_PRSNT	BrkSw2Stat
ESP_PD_SUPP	LAT_ACCEL	FrontWiperInUse
ESP_Disabled	ESP_Lamp_Req	ESP_BrakeSwitch
ETC_LmpOn	BrkEnbl_LCM	lat
FullBrk_Actv	AccelPedalPosition	TSC_EN
ABS_Lamp_Req	ETC_LmpFlash	BrkSw1Stat
YawRate	SteeringWheel_Angle	Odometer
BATT_VOLT	ESP_PD_SUPP	PARK_BRK_EGD
BATT_VOLT	ABS_BrkEvt	LAT_ACCEL
EngineSpeed	BrkSw1Stat	ESP_PRSNT

Table C.9: Arada OBU Data Fields

Colophon

This thesis was typeset using Leslie Lamport's \LaTeX , an advanced development of Donald Knuth's \TeX . The body text is set in 12 point Computer Modern Roman, designed in METAFONT by Knuth.

Document class for a UWindsor thesis requires `\documentclass[oneside,12pt,letterpaper]{book}` using a single sided, double spaced environment. Margins are one inch top, bottom and right, one and one-half inches on the left to allow for binding.

Tables (and pseudocode) were shown using Computer Modern Sans Serif adjusted to various point sizes to suit the contents. The CMSS font is effectively a variant of Helvetica.

Vita Auctoris

Ian Douglas completed his (four year) Honours Bachelor in Computer Science (*with honours*) at the School of Computer Science, University of Windsor in 1985. His interests were assembly language programming (IBM s/370) and Computer Graphics (in C on the Tektronics Unix O/S known as TNIX using a Tek 4105 Graphics terminal). Since then he has worked in Information Technology for several manufacturing businesses. A private pilot license followed in 1987 through the Windsor Flying Club. This led to an interest in navigation, which becomes valuable when sailing.

In 2014 Ian joined IEEE in during his master's studies, and he will complete his Master's Science in Computer Science degree in 2016, also through the University of Windsor School of Computer Science.

Ian resides in Tecumseh with his wife, Mary, and enjoys spending time with his family, including two delightful grandchildren.