OPTIMIZING GPS TRACKING FOR A SPACE ROVER VIA KALMAN FILTERING

Thesis · May 2022

DOI: 10.13140/RG.2.2.28730.36809

1 author:

Takoua Bejaoui
San Jose State University
2 PUBLICATIONS **0** CITATIONS

SEE PROFILE

OPTIMIZING GPS TRACKING FOR A SPACE ROVER VIA KALMAN FILTERING

A ME 295B Final Report

Presented to

The Faculty of the Departments of Mechanical Engineering and Computer Engineering

San José State University (SJSU)

In Partial Fulfillment

of the Requirements for the Degree

Master's of Science

by

Takoua E. Bejaoui

May 25, 2022

ABSTRACT

The overall purpose of this study is to optimize the derivation of a linear Kalman filter to establish a simplified and integrable navigation system for a mobile agent. A Kalman filter, with respect to sensor fusion, and the implementation of Robotics Operating System (ROS) are the main methodologies of this study. The agenda of this report will contain a brief background on ROS and the associated sensors used for the estimators, literary review, mathematical concepts used, methodologies completed, Matlab results and ROS updates, comparisons between different estimators, as well as future works. The dynamic and static data of the sensors (particularly the magnetometer, GPS and accelerometer) were collected and preprocessed to establish the inputs and measured outputs for the estimators. The conclusion states that the Kalman filter had an average tracking performance higher than that of the low pass filter by 0.3 percent in the forward (Y) direction but 9 percent in the X direction. The low pass filter's accuracy came with a 3 second delay. The final values for Q and R were 0.15 m/s^2 and 4 m, respectively. The selected sensors and the methods used in how they were pre-processed played a major role in the high tracking performance of the Kalman filter.

ACKNOWLEDGMENTS

I would like to express my deepest gratitude for the following people, for if it were not for their collaboration, this project would not have been possible:
Dr. Agarwal, Dr. Nicole Okamoto and Dr.Viswanathan for maintaining the ME establishment, at SJSU, to continue providing opportunities for interested students on a global scale.
Dr. Saeid Bashash, for supporting and maintaining the role as Committee Chair throughout the duration of this project, as well as, the many students, including myself, that were fortunate to benefit from his tutelage and mentorship. The extent of the depth of this project would not have been possible if it weren't for his consistent guidance and support.
Dr.Wencen Wu and Dr. Stas Tiomkin, for their roles as Committee Members throughout the duration of this project, for their constant support and collaboration, as well as, asking the right questions to better guide the direction of the project.
Nelson Wong and Gaurav Kupta (SJSU Graduate students from the Computer Engineering (Comp.E) and Computer Science (C.S) departments, respectively), leaders of the SJSU Robotics Team who initiated the first machine learning and ROS summer 2020 workshops, which has established my foundation in the realm of intelligent systems and ROS.
Mason Sage (ME student and president of the SJSU Robotics Club) and the rest of the SJSU Robotics Club (Lattapol Wongpiya, Khalil Estell, Nathanael Garza, Yosef Mirsky and Maxim Vovenko) for their feedback on questions that ranged from hardware resources (in relation to ROS) to electrical schematics, as well as, their support and patience throughout the duration of this project.
Joshua Ramayrat, SJSU Alumni Grad student (and a member of the Robotics Team at SJSU) who's insight and experience using sensor fusion and mobile robotics, along with sharing Matlab/Mathworks concepts was time saving and valuable.
Stanley Krzesniak, AE Grad student at SJSU, for his mentorship on GPS/GNSS receivers and how it all works.
Reihan Ashraf and Sumaid Mahmood for their efforts in providing technical support during the ROS debugging phases.
Aycan Hacioglu and Ronal George from Mathworks (Matlab), whose insight into Matlab's libraries and ROS1 helped in the efficiency of the validation processes and methodologies.
Last, but not least, my mother, Sallouha El Bejaoui, who continues to be a constant supporter and advisor throughout my academic and career pursuits.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS & SYMBOLS

ROS = Robotics Operating System
SLAM = Simultaneous Localization and Mapping (found in ROS)
VSLAM = Visual SLAM
ORB-SLAM = Oriented FAST (Features from Accelerated Segment Test) & Rotated BRIEF (Binary Robust Independent Elementary Features) feature detector-Simultaneous Localization and Mapping.
VO = Visual Odometry
OGM = Occupancy Grid Map
RTLS = Real Time Locating Systems
INS = Inertial Navigation System
AHRS = Attitude Heading Reference System
MARG = Magnetic Angular Rate and Gravity
MEMS = MicroelectroMechanical System
MOT = Multi-Object Tracking
RGB-D = Red,Green,Blue Depth (Sense Camera)
TOF = Time of Flight
LiDAR = Light Detection and Ranging
CNN = Convolutional Neural Networks
AI = Artificial Intelligence
CV = Computer Vision
AR = Augmented Reality
RNN = Recurrent Neural Networks
ICP = Iterative Closest Point
A-GPS = Assisted GPS
GPS = Global Positioning System
DOD = Department of Defense
UAV = Unmanned Air Vehicle
MPU = Motion Processing Unit
IMU = Inertial Measurement Unit (interchangeably used with MPU, if it has a geo-magnet.)
PID = Proportional Integral Derivative (Controller)
SMC = Sliding Mode Control (nonlinear estimator)
ASMC = Adaptive Sliding Mode Control (nonlinear estimator)
KF = Kalman Filter (linear estimator)
EKF = Extended Kalman Filter (nonlinear estimator)
UKF = Unscented Kalman Filter (nonlinear estimator)
EnKF = Ensemble Kalman Filter (nonlinear estimator)
LQR = Linear Quadratic Regulator
DLQR = Discrete LQR
DLQRE = DLQR Estimator
DOF = Degrees of freedom
UTM = Universal Transverse Mercator (grid)

MIMO = Multi-Input Multi-Output
SISO = Single Input Single Output
μ = gravitational parameter
m = meters
h = specific angular momentum
v = velocity
t = time
θ = angle (degrees)
θ' = angular velocity

θ'' = angular acceleration
d = distance
l = length
x = position in the x-axis (linear)
y = position in the y-axis (linear)
z = position in the z-axis (linear)
x' = velocity along the x-axis (linear)
y' = velocity along the y-axis (linear)
z' = velocity along the z-axis (linear)
x'' = acceleration along the x-axis (linear)
y'' = acceleration along the y-axis (linear)
z'' = acceleration along the z-axis (linear)
q_n = Quaternion vector with regards to the 4d complex-real plane and rotation/orientation
p = Position within and outside the quaternion sphere

Chapter 1 - Introduction

1.1 Motivation

1.1.1 Background

Autonomous robots, or mobile agents, such as rovers, unmanned air vehicles (UAVs) and small indoor robotic house cleaners, are dependent on sensor integration, as well as filtering of the associated sensor noises, to complete any basic scripted task. Some of the common sensors used are: LiDAR, RealSense cameras, gyroscopes, magnetometers, accelerometers and GPS/GNSS. A combination of some of these sensors yield a different type of navigation or stability system for the physical/dynamic system to use, such as: IMU/MPU (yields raw data from the gyroscope, magnetometer, accelerometer and (sometimes) barometer sensors); AHRS (fusion of IMU and magnetometer or MEMS) to yield orientations: roll, pitch and yaw angles and may fuse with odometry sensor(s) as well; MARG (fusion of magnetometer, gyro and accelerometer); and INS (fusion of IMU/MPU and GPS). AHRS and IMU/MPUs can be GPS-Assisted, but not fused. Fusion only occurs when a Kalman filter is introduced into the system. Thus, the navigation industry pathway with these systems usually starts from IMU/MPU, then to AHRS and eventually to INS. Due to the sensor noises, dead reckoning and drift, estimators (or filters) are integrated into the closed control systems design's feedback loop , to remove noise and improve tracking accuracies. A few common examples of such are: Linearized Kalman, Unscented Kalman, Ensemble Kalman, LQR (along with its derivatives: DLQR and DLQE), complementary filters, and the basic low/high pass filters.

A common software tool used for sensor fusion and autonomous mobile agents is the Robotics Operating System (ROS). It is a software package developed by the Stanford Artificial Intelligence Laboratory in 2007. It allows an operator to use a single computer to control different autonomous operations for multiple physical systems. It allows ease of communication between multiple processes, and is referred to as "complexity via composition"[1]. For instance, the rover is connected to a web server that is responsible for establishing a link between the sensors for the hardware (i.e motor drivers, IMUs for wheels and joints, respectively) and the codes/scripts (preferably in either C++ and Bash/Shell) that are manually operated. ROS can then 'wrap' them and establish them as drivers within itself. Thus, ROS 'GET' commands are now linked to the webserver and associated sensors and hardware.

There are different libraries that embody flexible algorithms, thus building/compiling (via 'rosbuild' or 'catkin'), writing and reading code are all possible actions for a variety of mobile robotic applications. Some of the common algorithms are "navigation, mapping, and motion planning"[1]. All proprioceptive and exteroceptive sensor data, as well as other 'messages', can be recorded and played back via the ROS-bagging feature ('rosbag'), which allows optimization techniques to be implemented. It comes with a simulator (turtle sim or Turtlebot) to test various robotic functions in a virtual space; thus alleviating production costs. Other virtual 2-D and 3-D simulators are Rviz and Gazebo [2], respectively. ROS is not the only software package that has all the aforementioned characteristics, but it does offer a large community support that aids in the debugging and setup process. ROS is not a programming language, Integrated Development Environment (IDE), nor a stand-alone library. It has many distributions (hydro, groovy,

indigo...etc.) and may have slight modifications to the way topic names and message types (also referred to as ‘type of topic’) are called.

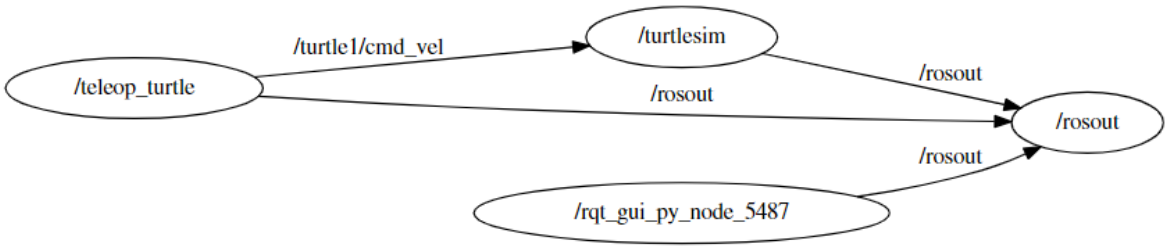


Figure 1: *A sample diagram of how messaging between nodes through various topics are visualized in ROS. [1] The ovals are the nodes, and the nodes either publish (send/transmits) or subscribe (receives) to topics using messages. The topics are the text written on the arrows in between nodes (i.e. /rosout).*

Simultaneous Localization and Mapping (SLAM) is a feature hosted within ROS’s framework. One of ROS’s navigation packages is “g_mapping”. It accepts odometry and/or Global Positioning System (GPS), as well as, Light Detection and Ranging (LiDAR) based data. This data, through ‘ros-bagging’, is then used to create, either, a 2-D or 3-D grid or mapping of the local area and trajectories of the mobile robot. These active sensors are also used with Inertial Measurement Units (IMUs). This fusion is sometimes referred to as Navigational System (NS) or Inertial Navigation System (INS), when equipped with a computer [3]. The recent ‘Tango Project’, a collaboration between JPL and Google, used versions of SLAM (Visual SLAM (VSLAM)) and Visual Odometry (VO) to 3-D map their surroundings using smartphones [4]. The Perseverance Mars rover project, also from JPL, primarily depends on VSLAM technology for localization due to the absence of GPS in space. Fortunately, future launches of low orbiting interplanetary satellites, along with changing the mode of communication links from radio to laser, will change the navigational capabilities in deep space.

Global Positioning System (GPS), originally known as NavStar GPS, is a U.S owned radio-navigation satellite system. It was an assigned collaboration between NASA and the Department of Defense (DOD) in 1965 [5]. It was initially called TIROS then later renamed to Defense Meteorological Satellite Program (DMSP), once the two collaborators split [5]. These 30 weather satellites orbit about LEO, and are managed by the U.S Space Force. GPS satellites are one of the many different types of satellites that make up the overall Global Navigation Satellite System (GNSS). Data collection on space and terrestrial weather, navigation, networking and communications, for both civilian and military purposes, are some of the general tasks that these satellites are used for. Satellites send radio signals that contain a navigation message (which carries key information on when the satellite broadcasted the signal and where it is in space) and a navigation signal (a pseudo-random code [6])[7].

The commercial GPS sensors are receivers with antennas and are used in vehicles, mobile apps and in robotics. They depend on the NavStar GPS for localized navigational information, such as velocity, position and timing. The GPS sensor contains a quartz clock, instead of the atomic clock found in a satellite, and measures the amount of time it takes for it to receive a

signal from four satellites [6]. If the receiver (GPS sensor) had an atomic clock, the length of the delay is equal to the length of that signal's travel. The receiver, then, multiplies this delay (seconds) by the speed of light to determine how far the signal has traveled. However, with the quartz clock, the receiver instead calculates the intersection zone between the four satellites. There are two processes that determine one's location via GPS receiver and the intersecting circles from ground receivers and they are triangulation and trilateration [6]. Despite the internal corrective algorithms in place, the accuracy ranges within a few meters, and better performance is usually achieved through differential GPS[7]. A differential GPS requires a setup of two stations and the mobile robot, or the rover in this study, must navigate relative to the reference receiver [7]. Furthemore, time delay due to the atmosphere also accounts for the different clock readings. Smart phones and devices depend on a configuration called Assisted GPS (or A-GPS) where the data acquisition from both satellites and receivers are faster and have higher sensitivity [7].

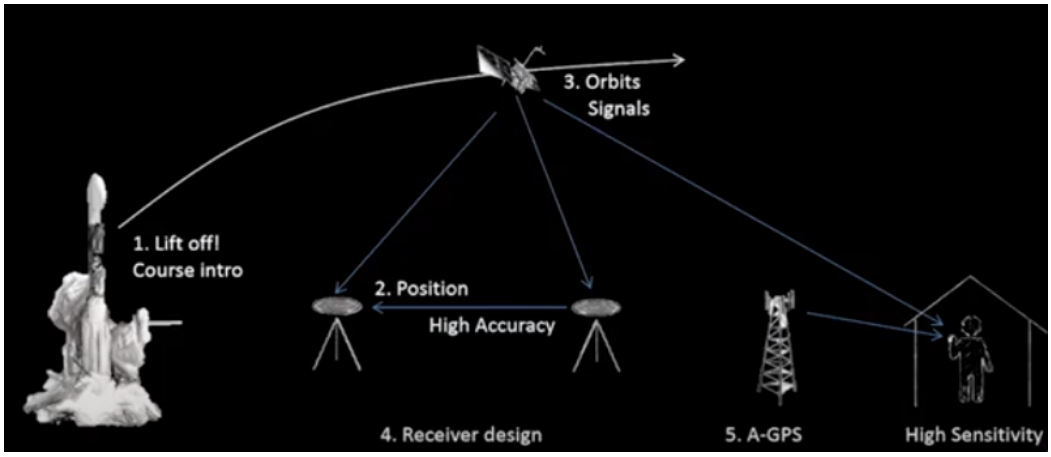


Figure 2: *GPS diagram in different scenarios. [7]*

The precise measurement in arrival time, of the receiving signal from the satellite, is called pseudo ranging [7]. Satellites move at 2 miles/second, thus knowing the time and the location of the GPS satellite with respect to signal transmission is key. However, due to the difference in time with respect to reception and transmission, orbital determination in real time, or within one meter of accuracy, is difficult to attain. Another component is the signal speed, which is close to the speed of light, but is slowed down due to Earth's atmosphere, buildings, tunnels, underground bunks…etc. GPS receivers have a different clock than the atomic clocks that are aboard satellites, thus there is a time mismatch due to the delay in the radio signal (from the satellite(s)), as well as the different clock measurements. Furthermore, GPS signals travel at a lower frequency than other sensors, thus when fused with MPU/IMUs create instances known as 'IMU dead reckoning', where the sensor with the higher frequency (in this case the IMU/MPU) acts as an input to predict the tracking steps of a robot where the GPS receiver's low frequency is unable to capture.

MPUs or IMUs behave as internal compasses (if 9 axial with the addition of the magnetometer) and stabilizers (due to the incorporation of gyro and accelerometer, each are

3-axial) for mobile vehicles, such as drones (UAVs) and rovers. They contain an accelerometer (measures gravitational and linear acceleration in the x, y, and z axis), a gyroscope (measures angular velocity and orientation along another x, y and z axis), and (sometimes) a magnetometer (another 3-axial DOF) that measures the push and pull (how close or how far) the magnetic north is with respect to each axis (compass). Oftentimes, the 'heading' is referred to how far or close the azimuth angle is to the magnetic north with respect to the magnetometer's readings; while 'attitude' refers to the orientations read by the gyroscope.

Visual mapping, tracking and localization typically required the combination of cameras and LiDAR, regardless if indoors or outdoors. LiDAR is popularly used for outdoor applications since cameras are light sensitive, thus information gathered from the pixelated images can be distorted; whereas, LiDAR is laser based and measures depth of the surrounding 3-D environment. Other visual sensors are augmented reality (AR) and visual reality (VR), but ROS1 versions prefer computer vision (or OpenCV) algorithms.

The demonstrations of ROS integration have been implemented in the following categories: navigation, localization and mapping of mobile robots (i.e. autonomous vehicles, drones, rovers…etc.) and smart grid and renewable energy implementation at the commercial and industrial scales. Some of the common evaluation metrics for such studies centered about enhancing tracking and mapping accuracy, error reduction in sensor noise, object avoidance and collision free, as well as, object detection (and segmentation).

1.1.2 Significance

The significance of this project is to establish the foundation for a simple and integrable navigation system for a mobile system. The results found within this study will impact the controller design for localization and mapping for the SJSU Robotics Club's space rover. Most applications found in literature used multiple sensors (especially visual/depth based sensors), used a nonlinear filter, and/or the navigation system established is constrained to the type of mobile agent used. The mathematics used aren't often clearly explained or cumbersome due to the many parameters involved in the state space representations nor is the transition from theory to simulating real-time sensor data in ROS adequately presented. Furthermore, there is a lack of research done towards creating a control system design that is flexible and can be integrated into ROS. Thus, the contents of this study will pave a way towards a simplified, but integrable approach towards autonomous navigation for a professional, hobbyist or student to adopt to use for their terrestrial agent or UAV that is in constant cruise altitude and operating at small accelerations and within a single geodetic/UTM zone.

1.2 Literature Review

Most research applied SLAM/ROS to inertial based systems and cameras to achieve optimal tracking. Thale et al., used ROS to compare different virtual simulation platforms and nonlinear

controllers [2]. Menna et al, conducted field testing using a Pioneer P3-AT mobile robot that had an integrated NS and INS via ROS, and will be used for underwater drones [3]. Kim et al., created a hybrid system by combining GPS and IMU sensors within a Kalman filtered controller, which results in low tracking error between actual and calculated [8].

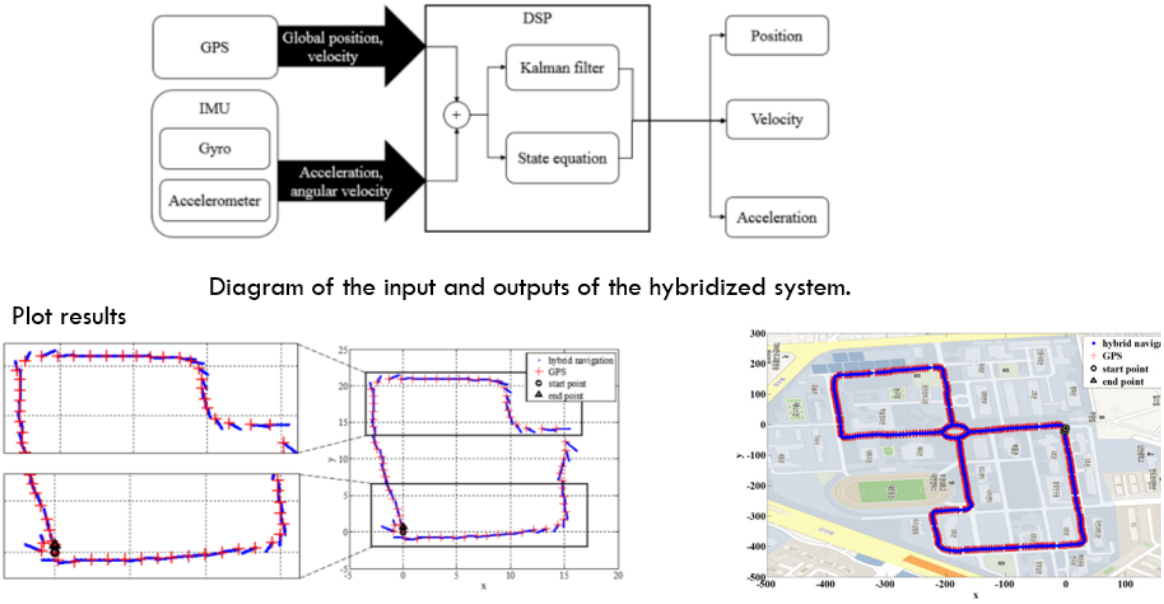


Figure 3: *A simplified hybrid approach developed by Kim et al.[8]. The results compared GPS to sensor fused (GPS + IMU) tracking.*

Motlagh et al., used ROS and a fusion of IMU and VSLAM for an autonomous aerial vehicle in scenarios without GPS, and used EKF to reduce local navigational errors [9]. Kellalib et al., conducted a sensor fault detection approach, via SLAM/EKF and equipping the mobile robot with proprioceptive and exteroceptive sensors, to provide a solution in a situation where the mobile robot encounters faulty hardware [10] .

Slowak et al., added Iterative Closest Point (ICP) algorithm and Lidar technology to the VSLAM approach in a GPS/GNSS denied environment for their mobile platform [11]. Park et al., conducted an integrated approach of using internal vehicular sensors with a Kalman filtered controller to optimize state estimation [12]. Jin et al., used a nine axial IMU with an EKF in a ROS system for navigation in a GPS and Lidar denied environment[13]. The argument was to provide cost effective solutions thru reducing the amount of sensor technology used while preserving the navigation algorithm's accuracy.

Oh et al., improved navigation performance via the insertion of a moving average filter between the signal tracking error estimator and the integration Kalman filter (to avoid other measurement errors); their approach combined INS, MEMs and GPS sensors [14]. Cheng et al., conducted three testing scenarios to compare localization results that involved a PID and Kalman filtered controllers with ROS/SLAM integration [15]. Kiss-Illes et al., wanted to test using the ORB/GPS-SLAM approach with slow frame rated data, since the ORB-SLAM produces tracking issues due to lack of continuity[16]. Cho et al., replace camera sensors with

IMU to compare tracking response plots [17]. The argument was to provide alternative solutions in addition to the commonly used technology in localization and mapping.

Other works focused on providing new incoming researchers, such as Yousif et al., in the mobile robotics field with the basic foundation needed to understand how to implement the more basic localization techniques (wheel odometry) to the more current (V-SLAM and OV) techniques to their navigational algorithms [18].

While still others focused on developing an estimator model with its associated state space representation model. Studies besides Kim et al. [8], helped provide the basis for this report. Kumalasari et al. developed a Kalman filter (for UAV) to better estimate GPS receiver position (long., lat.) by reducing noise. Their team fine tuned their Q and R matrices for their estimator through the accumulation of datasets with varying features with respect to weather and speed of the mobile agent.

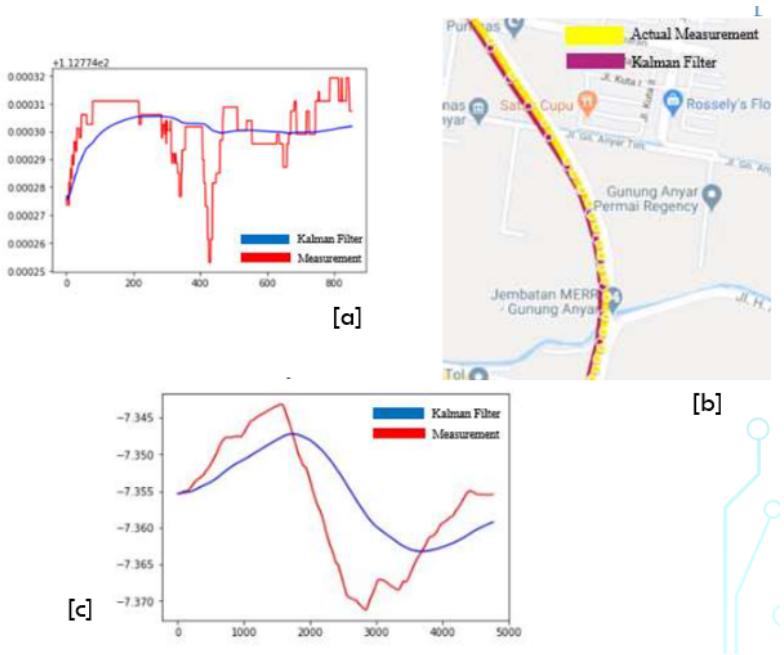


Figure 4: *Simulated trajectory onto Google maps and fine tuned the estimator via its Q (co-variance noise of the inputs/state) and R (co-variance measured noise of the output - GPS) matrices. Plot [a] is the longitudinal raw data vs the Kalman filtered one and the same goes for plot [c] but with respect to latitude instead. [19]*

Song et al. developed a Kalman filter to improve the inaccuracies (esp. due to signal masking) of the in-built NS (GPS and vision) aboard the UAV [20]. The Kalman filter improved the positioning accuracy by 0.8m.

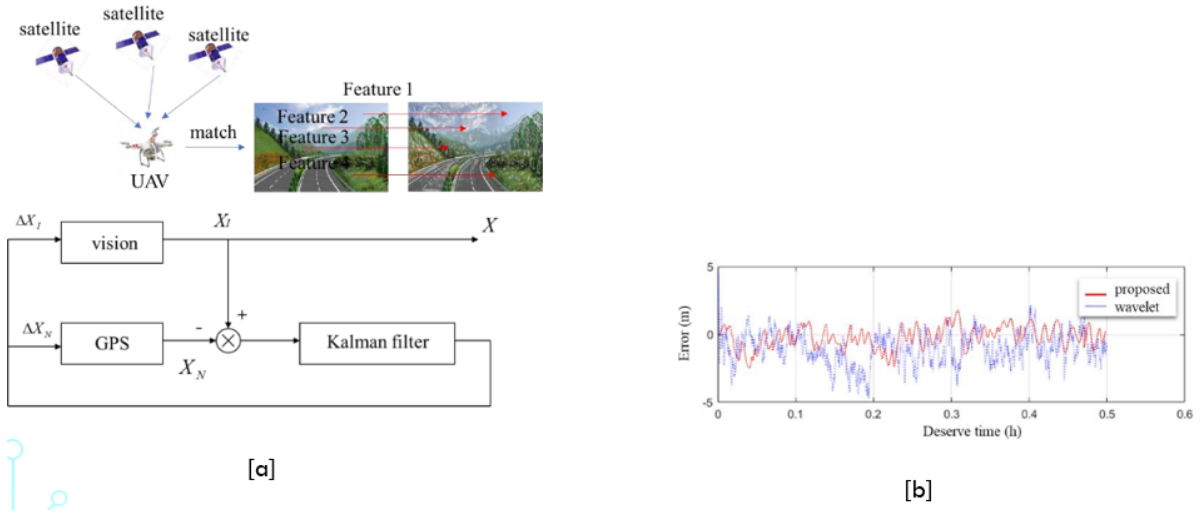


Figure 5: *Introduced a combination of navigation and path smoothing. The model switched between the primary/main navigation vs auxiliary navigation mode for either the camera or GPS with respect to the ambient environment of the vehicle. For instance, if the weather was observed to be cloudy, then the GPS would be configured to be the primary navigation mode. [b] presents the simulated results in Matlab where camera was the primary and GPS was the auxiliary navigation mode [20].*

Al-Khatib et al. developed an EKF based model with input/output feedback for tracking a robot [21]. The robot was equipped with an encoder, compass, IMU and GPS and were mapped to test the performance of three different approaches.

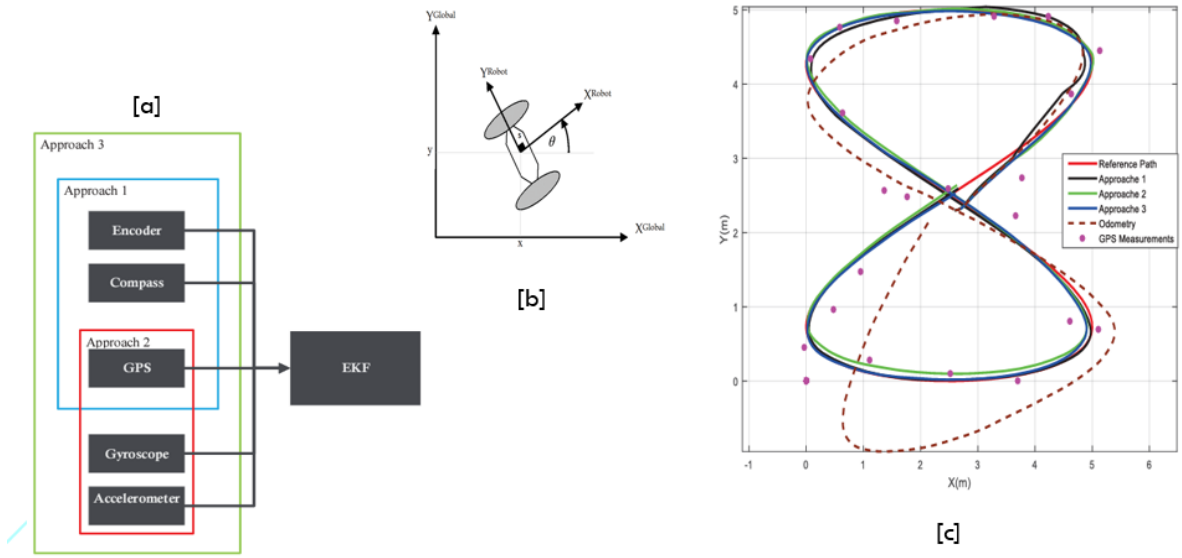
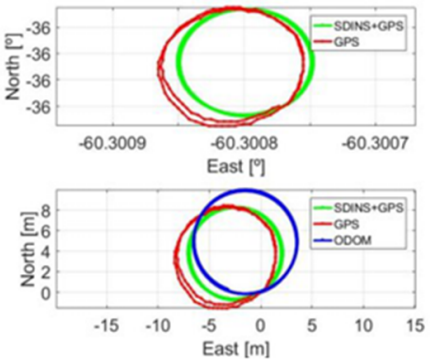


Figure 6: *Al-Khatib's et al. model and the three different approaches for comparison tracking [21]. Tested the approaches on 8 different paths and found that approach 3 had the best while approach 1*

had the worst tracking performance.

Another recent research was done by Menna et al., where the team's study focused on developing a Navigation System (NS) via GPS/INS and an extended-Kalman filter (EKF) [22]. Testing involved simulations in ROS with real sensor data, and on the field, on a flat track, with a Pioneer P3-AT mobile robot. Their research shed light on converting sensor data (GPS and gyro) to NED frame (local true/absolute geographic coordinate frame). Their conclusion is that the NS benefited from the hardware integration provided by ROS, and that the NS can be integrated onto other mobile agents such as Autonomous Surface and Underwater Vehicles (ASV and AUV) .



[a]

[b]

Figure 7: *Comparison plots [a] between GPS vs. NS-GPS fusion vs. odometry lead tracking model with respect to the robot[b] [22].*

1.3 Project Proposal

1.3.1 Objective

The objective of this project is to determine and compare modern non/linear estimators with the administration of MATLAB/Simulink and ROS packages to establish the foundation for a robust tracking, navigation and localization unit for a space rover. Optimization of tracking and error characteristics are performance parameters that are used to determine final selection for the future setup of the terrestrial agent's autonomous control system's design. The sensors selected to be outputs and inputs are a GPS receiver and an MPU (specifically the accelerometer and magnetometer), respectively. Optimization will be done via sensor fusion and Kalman filtering. A comparison will then be made between the linearized KF established via theory to a low pass filter, as well as, (if time permits) to the one embedded in ROS's robot_localization package: 'ekf_localization_node'; which uses EKF to locally track the movement of the mobile agent. The information in this study will be shared with the SJSU Robotics Team for future implementation within the Intelligent Systems Division.

1.4 Methodology

1.4.1 Approach

This project will be carried out primarily in the following three stages:

- **Establishment of the state space model :** via free body diagram and control systems design theory of a single rotation (yaw or psi angle) about the z-axis of the relative rigid body frame of a mobile agent. This model contains the necessary parameters for the estimator to be established: which are the A, B, G (noise), C, D, Q (variances of the accelerometer) and R (variance of the GPS measurements) matrices. Quaternions and Euler rotation matrices will also be used to convert certain states, from the absolute to the body frame.
- **Usage of a smart phone's sensors (Matlab Mobile app), as well as the individual GPS receiver and MPU sensor itself to collect data sample(s):** The data collected from the phone's sensors will be used for the first test runs in establishing the kalman filter. Once the kalman filter has been established, the actual sensors will be used to test its performance before implementing it in ROS. QGIS, Matlab and ROS will be used to simulate the necessary plots.
- **ROS implementation and comparisons:** will provide insight into how precise either method, linearized KF and the embedded EKF, is with respect to the tracking and navigation of the mobile agent.

First few steps are to establish the state space representation matrices through a free body diagram and the mathematics involved in the Euler rotation matrices and quaternion calculations. Next, is to collect all smartphone sensor data (GPS(λ, ϕ, h, v), accelerometer(x'', y'', z'') , orientation (φ, θ, ϕ), angular velocity (ω_x, ω_y, ω_z) and magnetometer ($X_{\mu T}$, $Y_{\mu T}$, $Z_{\mu T}$)) and plot the static noise for each (to determine whether or not the noise is of Gaussian distribution), as well as the parameters themselves. Then, once that's achieved, do the necessary transformations of the critical parameters to simulate and tune the kalman filter in Simulink/Matlab using the procured data. Last, but not least, use the actual sensors (GPS and MPU9250) and connect them to the simulated kalman filter via Matlab/PC/Simulink, as well as ROS's localized tracking library (localization_ekf) to compare plots.

1.4.2 Theory and Principles

Modern control systems design theory, mathematics with Euler transformations, and telecommunication theory of GPS modules would be covered more in depth in later chapters. Please, check Chapters 2 and 3 for a more in depth look at the aforementioned topics.

1.4.3 Resources Needed

Table 1. Potential resources and its expected acquisition date for each.

Item	Expected Acquirement Date (and Cost)
MATLAB/Simulink/Machine Learning (also used the free MATLAB online offered to SJSU students. Used version R2022a online to build the Kalman filter; but downloaded R2021b to connect with ROS1, Melodic setup.)	Acquired ($100.00)
Rover	Being built by Robotics Club
Raspberry Pi (Multiple) (may not use)	Provided by SJSU Robotics Club
iMac (Ubuntu 18.04 OS) [but will use a work laptop, Dell, when collecting outdoor real time sensor data.]	Acquired
Smart phone	Acquired
GPS/GNSS Module (BU-353S4)	Provided by SJSU Robotics Club
IMU/MPU-9250	Acquired ($10.00)
Arduino IDE	Free Download
ROS (Melodic)/Gazebo - need Linux OS	Free Download
QGIS (Long Term Release)	Free Download

The rover will not be functional throughout the duration of this study, but instead will be used as a point of reference with respect to how ROS is implemented within its paradigm. The information in this study will be shared with the SJSU Robotics Team for future implementation for both the Intelligent Systems and the Control Systems Design Divisions.

Figure 8: *Rover under construction by SJSU Robotics Club.*

1.4.4 Preliminary Work

The first step is understanding the overall framework of ROS, figure out whether to use Ubuntu Virtual Machine (VM) or have Ubuntu as an OS, and then download ROS. The Virtual Machine is basically Ubuntu in cloud and is supported by VMWare Fusion (for Mac OS), VMWare Workstation (for Windows OS) and/or VirtualBox. Familiarize oneself with the packages and how nodes and launch files work. Memory requirements for any machine running ROS is eight to 20 Gigabytes (GB), and Ubuntu Linux is the recommended OS (avoid using Windows, if possible).

The ROS Melodic Morenia and ROS Kinetic Kame are compatible with Ubuntu Artful (17.1) or Ubuntu Bionic (18.04 LTS), and Debian Stretch (among others) [22].

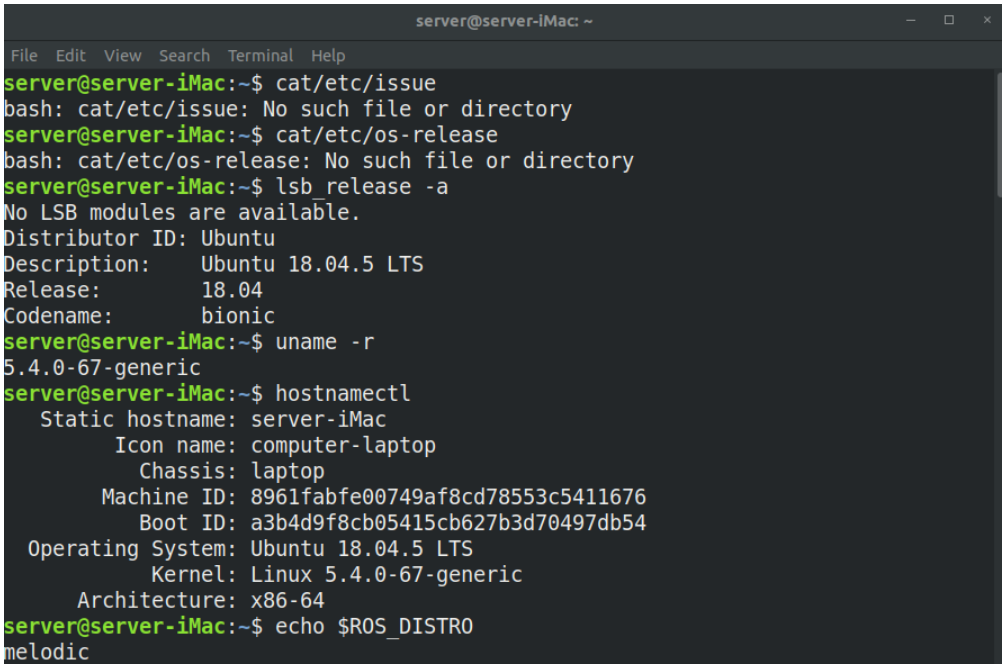


Figure 9: *Ubuntu terminal, using Bash, showcasing ROS distribution successfully installed.*

1.4.5 Anticipated Challenges and Alternative Strategies

1. Will use MATLAB full version and its featured Navigation, Sensor Fusion and Tracking, and UAV toolboxes; however, if not acquirable, will use the Matlab Online platform offered by Mathworks for SJSU students to use.
2. Another COVID lockdown:
    a. No outdoor real-time tracking, thus only simulated results will be done .

3. Time:
    a. Complete comparisons between the linearized Kalman filter and low pass filter with respect to smartphone's sensor data (only). ROS's robot localization package will have the setup completed but comparisons of the response curves between the Kalman filter and ROS's ekf_localization_node will be done as part of future works. The github repo stated in Appendix B will upload the ekf_localization_node's response plots, ROS's code and setup by September of 2022.

1.4.6 Deliverables

The final executable files for this project will be as follows:

1. MATLAB (.m) code for control design and system response

Software to be used to display the linearized kalman filter and associated sensor data plots.

2. Gain values for system response
   This is done to compare gain values that the kalman filter requires to reach optimal error with respect to the output state variables of the sensor data.

3. . xlsx or .mat files for storing captured data
   If data sets were to be used, Microsoft Excel is an option that will be used for data collection and uploaded to a MATLAB workspace.

4. .png/.jpeg images of plots created by either ROS or Matlab
   Images of the plots with respect to sensor data and tracking performances within the context of the kalman filter selected (linear kalman filter in Matlab and EKF in ROS).

1.4.7 Evaluation Metrics

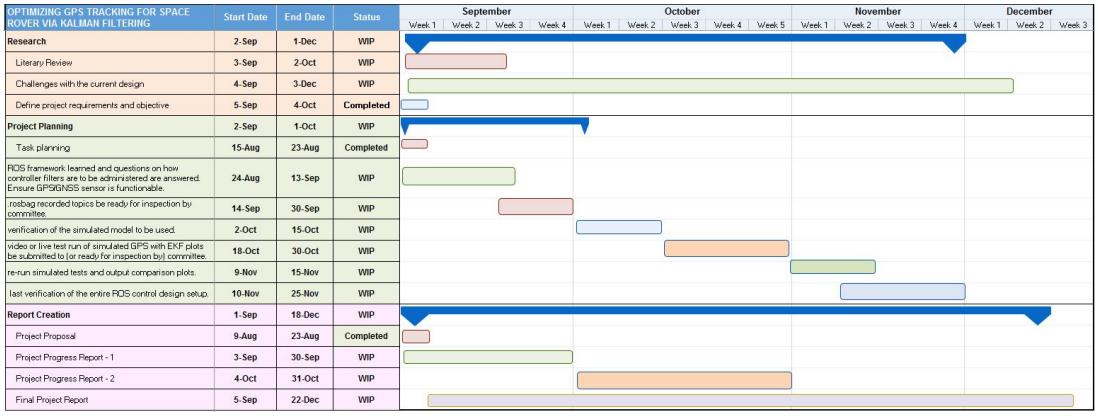
The associated evaluation metrics for this project will be as follows:

1. Lower tracking error (± 0.2 %)
   Minimum error with respect to the reference signal and the output response of the state variable. Tracking error response for navigation would include, but not limited to, the following parameters: distance [m] vs. time [ms].

2. Higher tracking performance (25mm accuracy)
   Tracking performance will be measured, initially, via simulated plots, established via Matlab/Simulink and ROS. Plots will display the error comparisons due to the kalman filter selected with respect to navigational tracking performance.

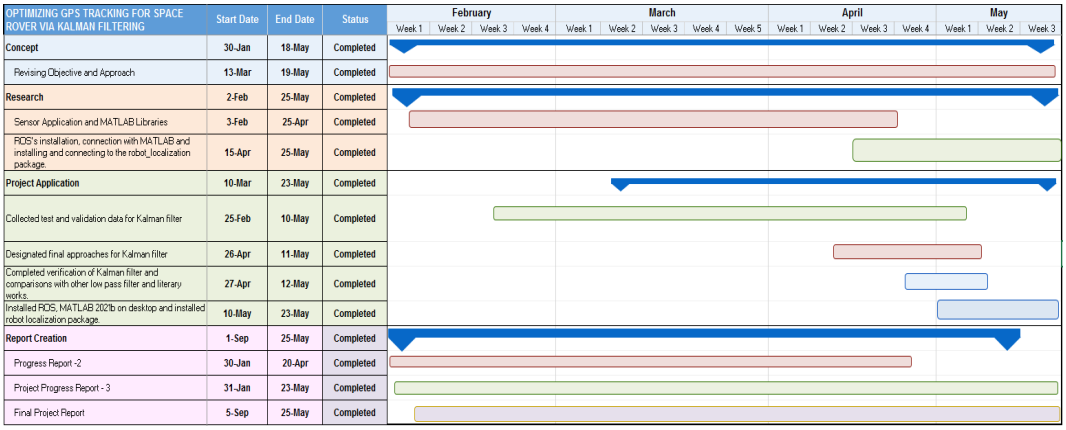
1.4.8 Timelines

Timelines are conventionally shown using a Gantt chart. The Gantt charts are seasonal, Hence, Phase A is Fall 2021. Next, is Phase B, which is for the Spring of 2022.



OPTIMIZING GPS TRACKING FOR SPACE ROVER VIA KALMAN FILTERING	Start Date	End Date	Status
Research	2-Sep	1-Dec	WIP
Literary Review	3-Sep	2-Oct	WIP
Challenges with the current design	4-Sep	3-Dec	WIP
Define project requirements and objective	5-Sep	4-Oct	Completed
Project Planning	2-Sep	1-Oct	WIP
Task planning	15-Aug	23-Aug	Completed
ROS framework learned and questions on how controller filters are to be administered are answered. Ensure GPS/GNSS sensor is functionable.	24-Aug	13-Sep	WIP
.rosbag recorded topics be ready for inspection by committee.	14-Sep	30-Sep	WIP
verification of the simulated model to be used.	2-Oct	15-Oct	WIP
video or live test run of simulated GPS with EKF plots be submitted to (or ready for inspection by) committee.	18-Oct	30-Oct	WIP
re-run simulated tests and output comparison plots.	9-Nov	15-Nov	WIP
last verification of the entire ROS control design setup.	10-Nov	25-Nov	WIP
Report Creation	1-Sep	18-Dec	WIP
Project Proposal	9-Aug	23-Aug	Completed
Project Progress Report - 1	3-Sep	30-Sep	WIP
Project Progress Report - 2	4-Oct	31-Oct	WIP
Final Project Report	5-Sep	22-Dec	WIP

Figure 10: *Gantt chart for phase A.*



OPTIMIZING GPS TRACKING FOR SPACE ROVER VIA KALMAN FILTERING	Start Date	End Date	Status
Concept	30-Jan	18-May	Completed
Revising Objective and Approach	13-Mar	19-May	Completed
Research	2-Feb	25-May	Completed
Sensor Application and MATLAB Libraries	3-Feb	25-Apr	Completed
ROS's installation, connection with MATLAB and installing and connecting to the robot_localization package.	15-Apr	25-May	Completed
Project Application	10-Mar	23-May	Completed
Collected test and validation data for Kalman filter	25-Feb	10-May	Completed
Designated final approaches for Kalman filter	26-Apr	11-May	Completed
Completed verification of Kalman filter and comparisons with other low pass filter and literary works.	27-Apr	12-May	Completed
Installed ROS, MATLAB 2021b on desktop and installed robot localization package.	10-May	23-May	Completed
Report Creation	1-Sep	25-May	Completed
Progress Report -2	30-Jan	20-Apr	Completed
Project Progress Report - 3	31-Jan	23-May	Completed
Final Project Report	5-Sep	25-May	Completed

Figure 11: *Gantt chart for phase B.*

Chapter 2 - Geographic Coordinates, Reference Frames and Quaternions For GPS and MPU Sensor Data

2.1 Geographic Coordinates

2.1.1 Background

Traditional grid mapping done by explorers and mathematicians in early times have carried into modern day conventional forms of geographical coordinate determination and navigation. These methods laid the foundation for global non/inertial reference frames used in, either 2D or 3D, engineering applications that required Euler transformations with respect to the orientation and mobility of an object. This chapter will focus on the correlation between different sensors' parameters, and the methods used to transform and mathematically equate them in the same context.

2.1.2 Different Types of Geographic Coordinates

There are different representations, derived from flattening the Earth to a 2D grid (map projections), to determine any particular geographic point on the globe. There is a spatial order, thus an understanding and awareness of them is critical to determine the correct context for the reference frame of an application. There are two main categories of geographical coordinates: cartesian (x-y-z, usually metric coordinates) and curvilinear angular (spherical/ellipsoidal polar coordinates). The earth is not a perfect sphere, but rather an ellipsoidal (oblate sphere) due to the flattened features at the poles. Thus, the geodetic datum and its ellipsoidal values, rather than the geocentric datum and its spherical values, are used.

Table 2. Typical geographic coordinates and reference frames.

Geographic Coordinate or Reference Frame	Details
East-North-Up (ENU) or North-East-Down (NED) local coordinate system	● Inertial x-y-z frame positioned locally within each UTM zone. ● Difference is within where x, y and z are pointing. ● Down doesn't necessarily mean the z axis is pointing towards the center of the Earth. But rather normal to the xy plane. It depends on where this tangential local surface plane is

	located with respect to the many zones that make up the ellipsoidal globe. ● Interchangeably defined as : ‘topocentric plane’ and ‘tangential local reference frame’.
Geodetic (longitude, latitude and altitude)	● Longitude and latitude expressed in degrees while altitude is in meters. The altitude is ‘h’ and is actually a summation of the height ‘N’ between the ellipsoidal datum’s surface and the geodic layer, as well as the height ‘H’ between the geodic layer and the actual surface of the Earth. ● Origin is the prime meridian for longitude and the equator for latitude.
Universe Transverse Mercator (UTM)	● Established by the Karney Kreuger equations [27] or the Redfearn’s formula. ● It’s origin is located at the intersection of the prime meridian and the equator. ● The X-axis runs horizontally along the equator and the Y-axis runs vertically along the prime meridian. ● Units are metric
Earth-centered-Earth-fixed (ECEF) coordinates	● Origin of this inertial frame located at the center of the Earth. ● It’s positioned the same for either geodetic (ellipsoidal) or geocentric (perfect sphere/spherical) coordinates. ● Sometimes used interchangeably with ECI coordinates, but ECI is primarily used for spacecraft applications for Euler based orbital transformations. ● The X-axis points towards the prime meridian, the Y-axis points towards 90 deg. East, and the Z-axis points upward towards the Zenith (North pole).

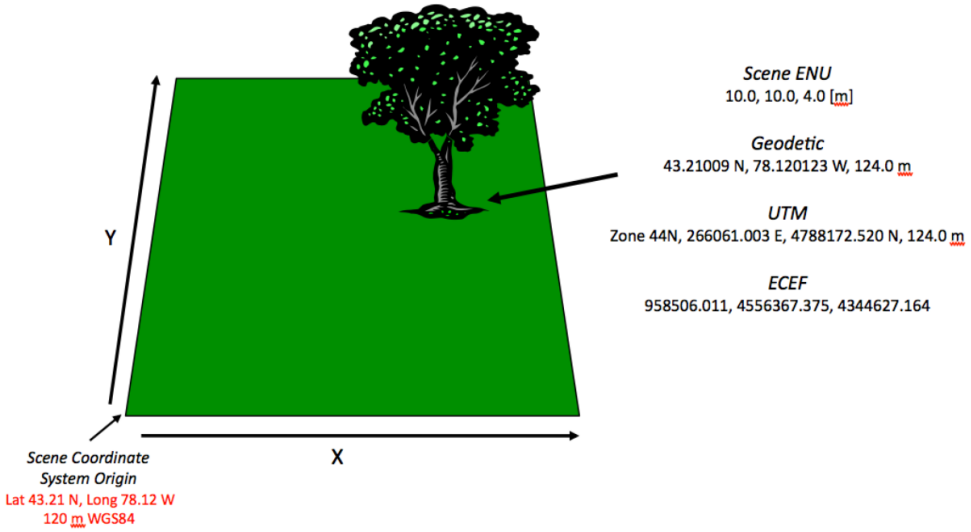


Figure 12: *An example of each coordinate and reference frame type. The main units are metric or degrees. Ellipsoidal datum selected is the WSG84 projection. ECEF is in meters. Each zone starts reading from the left bottom corner. [28]*

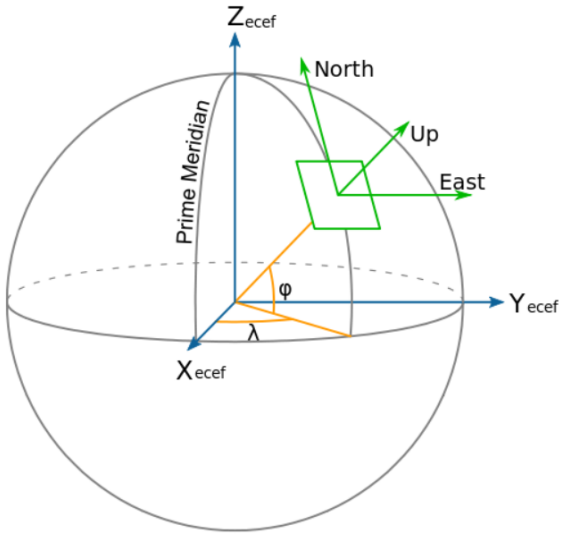


Figure 13: *Relationship between the different geographic coordinate systems and reference frames [28]. The ENU tangential local reference frame along the globe (can also be NED). In ENU, the y-axis is along the North, the x-axis is along the East and the z-axis is along Up. Note the direction of Up can be projected on points other than the center of the Earth depending on which zone it is at. Regardless if the datum is geodetic or geocentric, the origin of the X-Y-Z ECEF frame will be located at the center of the Earth. The longitudinal and latitudinal angles are* λ *and* ϕ *.*

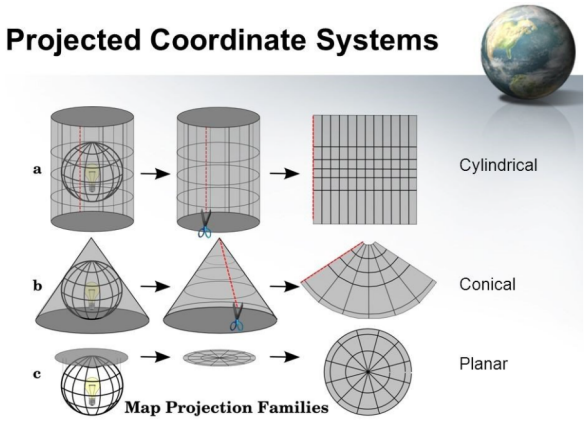


Figure 14: *The different types of map projection groups [29]. Each has its pros and cons. This is how the different UTM projections are established (i.e WSG84).*

The globe's continents are projected directly onto the surface of a fixed shape, either cylindrical, cone (secant or tangent) or a top plane (sliced through a latitudinal layer), and then cut open and flattened. The Universe Transverse Mercator grid's (UTM projection) shape, that is used by commercial use GPS/GNSS receivers, is cylindrical positioned at 90 deg (hence, the 'transverse' term). The pros of using this projection is that it preserves the shape of the continents and distances drawn and calculated between geographic points. However, this is only true for land masses below the latitude bands that do not encompass the North and South poles, where the land masses become highly exaggerated (as shown in the figure below).

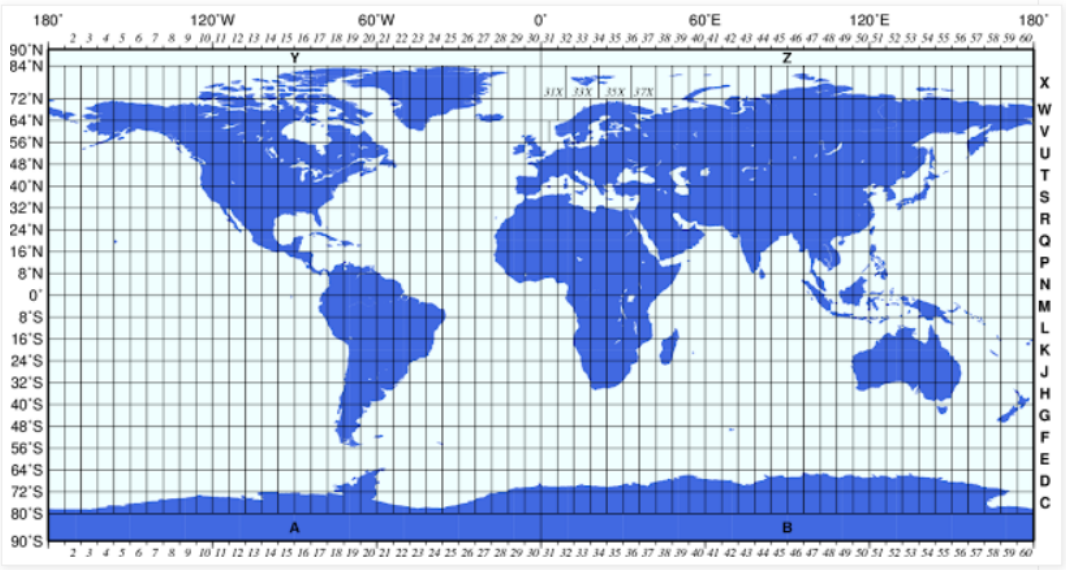
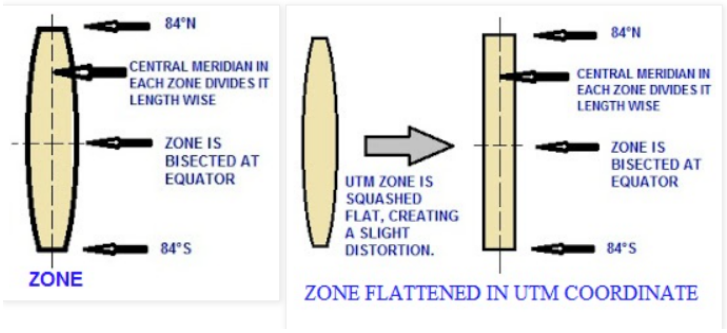


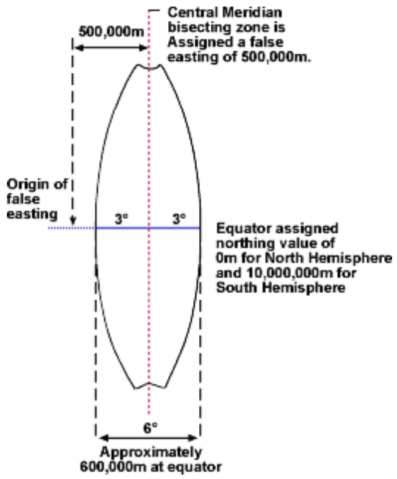
Figure 15: *UTM grid [30]. The degrees on the left and top sides of the grid are longitude and latitude, respectively. Each latitude corresponds to a letter, found on the right side of the grid, and are defined as 'quadrants' or 'bands'. The letters found on the top and bottom refer specifically to the poles. The '0*

deg.' is where the prime meridian line is located. Each grid is spaced at 6 degrees, from East to West (longitude, between 0 degrees and 180 degrees; positive is East, while negative is West), or along the X -axis. The Y-axis vertically cuts the whole grid in the middle - at 0 deg., while the X-axis cuts at the equator, also 0 degrees. The Y-axis covers the different latitudes, above (North) and below (South), of the equator (0 deg., x-axis.). Note how Greenland and Antartica's sizes are exaggerated.

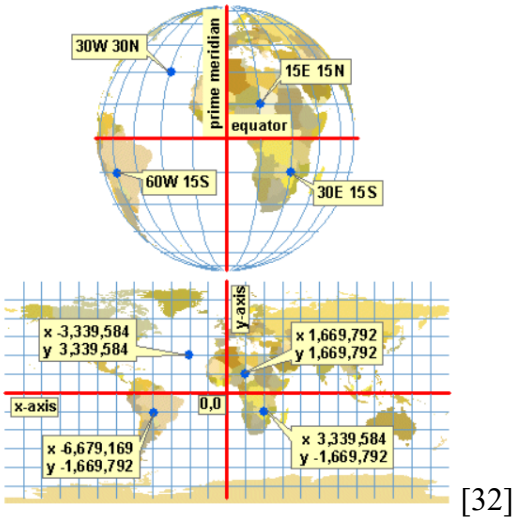
Each grid space or 'zone' has a central meridian and is spaced at 6 deg. along the x-axis.



[a] [30]



[b] [31]



[32]

Figure 16: *UTM zones are spaced at 6 deg. and use metric units. [30] [31][32]*

There are different datums for the Earth in which the cartesian and curvilinear angular geographical coordinates are dependent upon, and are generically categorized as spherical or elliptical. Note that since there are various eccentricities of the ellipse, there will be a variety of datums existing within the elliptical category. The type that GPS/GNSS receivers use is the WGS84 Ellipsoid datum. It is the most commonly used datum for most navigational systems. Thus, the sensor data received from the GPS is with respect to the WGS84 Ellipsoid datum, and

can then be converted to UTM coordinates via geographical based software, such as QGIS and ArcGIS.

2.1.3 Mathematical Conversions between the Geographic Coordinates

The datum is generally defined first and its selection is dependent on the type of sensors used. The datum is either a spheroid or an ellipsoid, and is projected from the center of the earth. The earth is generally divided into three generic layers : datum's surface, geoid (magnetic layer), and Earth's topology and atmosphere (depends if the mobile agent is terrestrial or aerial). There are different datums, especially for the ellipsoid, because just like projection mapping (discussed in section 2.1.2), it can fit for certain areas/features of the globe, and depending which features need to be accurate, that datum is selected. Fortunately, the type of datum is already selected due to the WGS84 ellipsoid being used in (and referenced within) the GPS receiver used for this study.

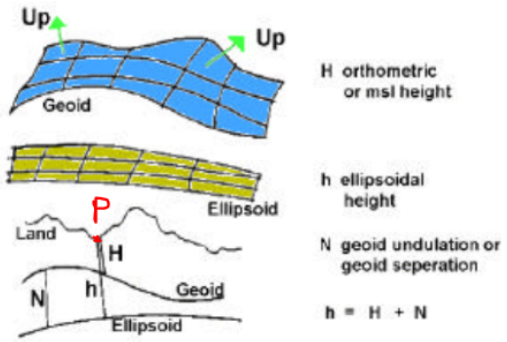


Figure 17: *Visual aide and calculation for altitude, 'h' in geodetic relative coordinates. [33] The altitude is one of the output parameters given by a GPS receiver and is in meters. It measures the distance between the datum's perimeter (in this case, the ellipsoid) and point 'P'. Point 'P' designates the location of either the local reference frame or the local body frame of the mobile agent as it travels upon the surface of the Earth.*

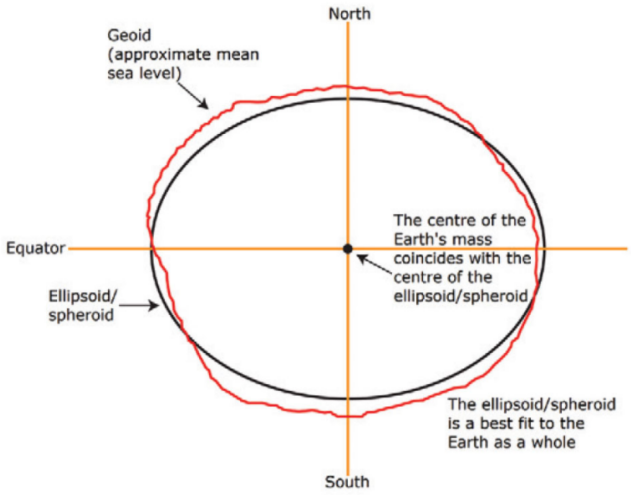


Figure 18: *An example of fitting a spheroid or ellipsoid datum over the Earth. [34] Note that the geoid layer is atop of the perimeter of the datum. Other spheroid/ellipsoid datums are: GDA94, NAD83 and ITRF2014.*

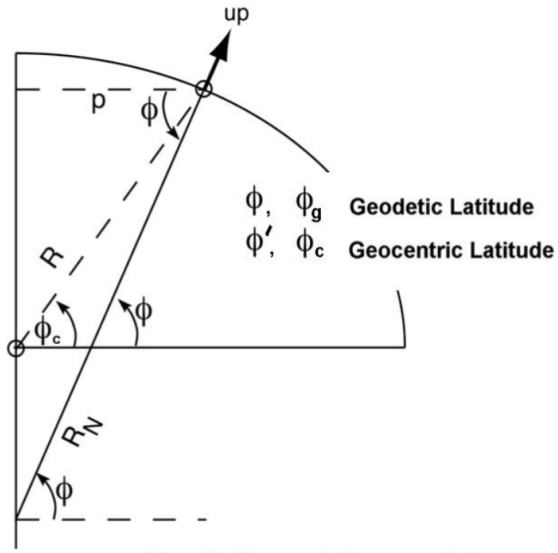


Figure 19: *Difference between geodetic (ellipsoidal) and geocentric (spherical - perfectly round) latitude measurements [35]. Note that despite the different 'R' values and locations of the origin, both datums are centered along the z-axis that cuts through the Earth from the North to the South pole. Thus, each datum has its own definition of where the center of the earth is located.*

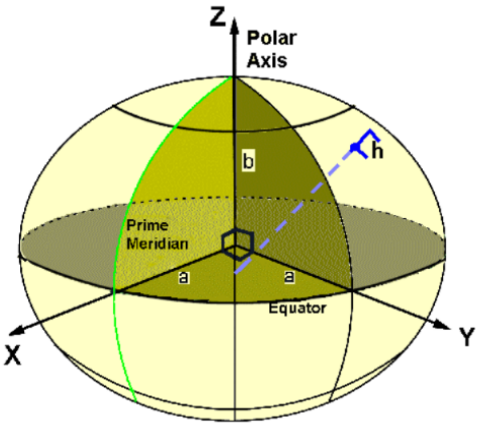


Figure 20: *Elliptical parameters 'a' and 'b' along the ECEF (cartesian) coordinates, as well as 'h' (height) from the geodetic ellipsoidal (curvilinear of angular) coordinates. X, Y is along the equatorial plane and Z is along the polar axis. [35]*

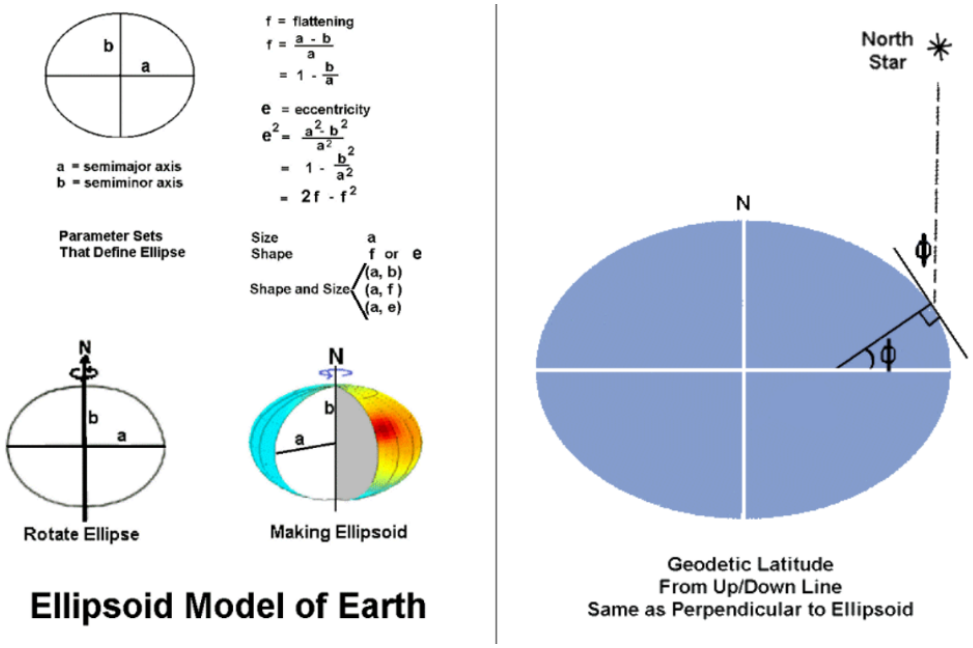


Figure 21: *Ellipsoid model of Earth and geodetic latitude [33].*

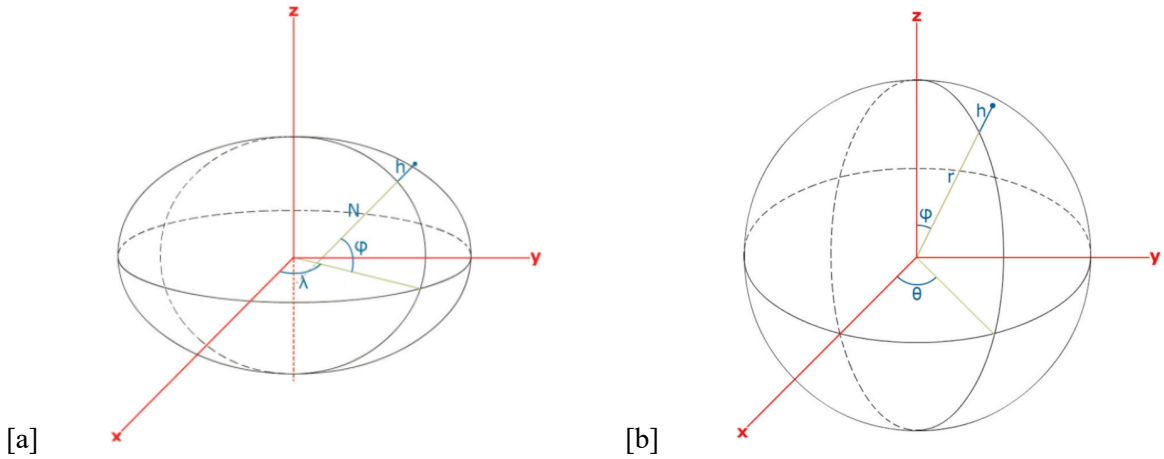


Figure 22: *Visual depiction of geodetic [a] vs geocentric[b]. [34]*

The table below lists the conversions between geographical coordinates. Please, refer to the figures above as reference to the parameters used in the calculations.

Table 3. Mathematical Conversions between geographical coordinates [34][35].

Geographic Coordinate Conversions	Mathematics
Geodetic and Geocentric	$\tan\varphi_c = \left[1 - e^2 \frac{R_N}{R_N + h}\right] \tan\varphi$ $R_N = \frac{a}{\sqrt{1 - e^2 \sin^2\varphi}}$ $\tan\varphi = \left[1 - e^2 \frac{R_N}{R_N + h}\right]^{-1} \tan\varphi_c$
Geodetic (longitude, latitude, height) to ECEF (X-Y-Z)	$x = (R_N + h)\cos\varphi\cos\lambda$ $y = (R_N + h)\cos\varphi\sin\lambda$ $z = ([1 - e^2]R_N + h)\sin\varphi$ [Geodetic] $N = R_N$

	Where: $N = \frac{a}{\sqrt{1 - e^2 \sin^2 \varphi}}$ And: $e^2 = \frac{a^2 - b^2}{a^2} = 2f - f^2$ N = radius of curvature of the prime vertical a = semi-major axis b = semi-minor axis f = flattening factor Geocentric: $r = R$ $x = (r + h) \cos\theta \sin\varphi$ $y = (r + h) \sin\theta \sin\varphi$ $z = (r + h) \cos\varphi$
ECEF (XYZ) to Geodetic (latitude, longitude and height)	$\lambda = \text{a tan}(y/x)$ $= \text{a tan 2}(y, x)$ $r = \sqrt{x^2 + y^2 + z^2}$ $p = \sqrt{x^2 + y^2}$ Using initial reference latitude value: $\varphi_c = \text{a tan}(p/z)$ $= \text{a tan 2}(p, z)$ $\varphi_{now} = \varphi_c$ For an interactive loop: $h = \frac{p}{\cos\varphi_{now}} - R_N(\varphi_{now})$ $\varphi_{next} = \text{a tan}\left[\frac{z}{p}\left(1 - e^2 \frac{R_N}{R_N + h}\right)^{-1}\right]$ Once latitude, ϕ, is found, then altitude is: $h = \frac{p}{\cos\varphi} - R_N$

2.2 Reference Frames For Sensor Data

2.2.1 Background

It is important to distinguish, visually, the different types of reference frames with respect to the field of robotics and navigation. Just like the coordinates, there is a particular order to transform from one reference frame to another. For simplification, it is advisable to use a reference frame that better aligns with the sensor's reference frame. The purpose of this section is to explore, mathematically, the relationship between the different types of reference frames used in mobile navigation.

2.2.2 Reference Frames

The reference frames used in pose estimation with respect to a mobile agent (regardless if terrestrial or aerial) are listed below [36]:

Table 4. Different reference frames for a mobile agent to determine position and orientation.

Reference Frame	Details
Geodetic coordinate system	● Longitude , λ (East or West) ● Latitude, ϕ (North or South) ● Height , h [N+H = h [m]] ● Origin at the prime meridian for longitude and at the equator for latitude. ● Longitude and latitude can be in units of degree decimal (dd) or degree-minute-seconds (dms).
ECEF coordinate system	● X, Y, Z ● Origin is at the center of Earth ● X is pointing towards the prime meridian, Y is pointing 90 deg. East and Z is pointing towards the North pole.
Local NED (North-East-Down) coordinate system	● x, y, z ● x is pointing north ● y is pointing east ● z is pointing down (normal)

	• Origin is at the central axis of the UTM zone. • Other versions are ENU (x-y-z) and END.
Vehicle carried NED coordinate system	• Local NED projection onto the mobile agent. • Origin of the local NED projection is at the center of mass of the mobile agent.
Body coordinate system	• Coordinates of the sensor, since it is attached to the mobile agent as it moves. • Make sure that the local reference coordinates used (either NED, ENU or ENH) have their axes aligned with that of the sensor's/body frame.

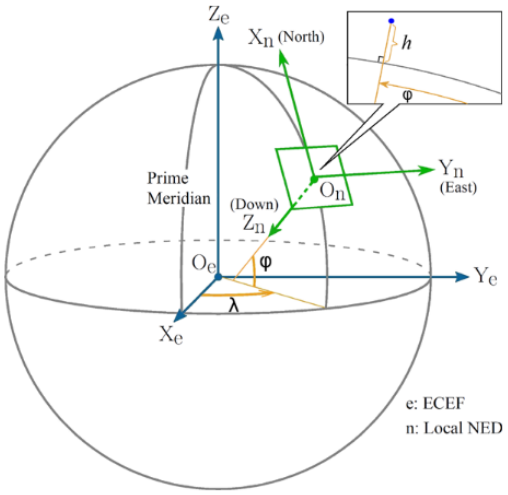


Figure 23: *Relationship between ECEF and NED coordinates [36].*

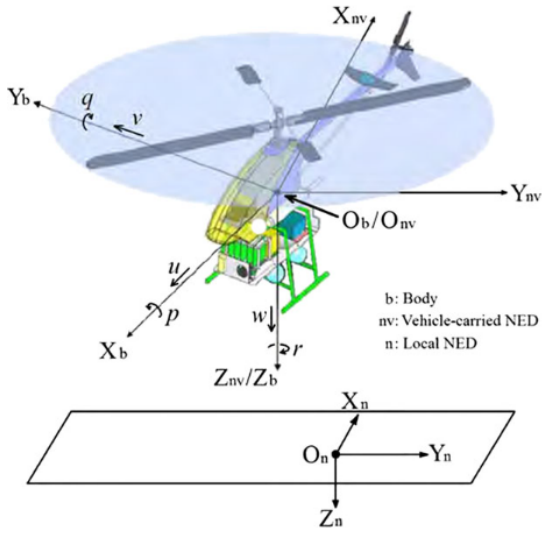


Figure 24: *Visual relationship between local NED, vehicle and body frame NED. [36]*

Certain assumptions that are to be made in association to the default geodetic datum: WSG84 Ellipsoid:

$$
\begin{aligned}
R_{Ea} &= 6{,}378{,}137.0 \text{ m}, \\
\mathbf{f} &= 1/298.257223563, \\
R_{Eb} &= R_{Ea}(1-\mathbf{f}) = 6{,}356{,}752.0 \text{ m}, \\
\mathbf{e} &= \frac{\sqrt{R_{Ea}^2 - R_{Eb}^2}}{R_{Ea}} = 0.08181919, \\
M_E &= \frac{R_{Ea}(1-\mathbf{e}^2)}{(1-\mathbf{e}^2\sin^2\varphi)^{3/2}}, \\
N_E &= \frac{R_{Ea}}{\sqrt{1-\mathbf{e}^2\sin^2\varphi}}.
\end{aligned}
$$

[36]

Where subscript, g, denotes geodetic and:

1. the semi-major axis R_{Ea},
2. the flattening factor $\mathbf{f}$,
3. the semi-minor axis R_{Eb},
4. the first eccentricity $\mathbf{e}$,
5. the meridian radius of curvature M_E, and
6. the prime vertical radius of curvature N_E.

[36]

The position vector for the geodetic coordinate frame is defined as:

$$P_g = \begin{pmatrix} \lambda \\ \varphi \\ h \end{pmatrix}$$ [36]

The position vector for the ECEF coordinate frame is defined as:

$$P_e = \begin{pmatrix} x_e \\ y_e \\ z_e \end{pmatrix}$$ [36]

Converting from geodetic position to ECEF position:

$$\mathbf{P}_e = \begin{pmatrix} x_e \\ y_e \\ z_e \end{pmatrix} = \begin{pmatrix} (N_E + h)\cos\varphi\cos\lambda \\ (N_E + h)\cos\varphi\sin\lambda \\ [N_E(1 - \mathbf{e}^2) + h]\sin\varphi \end{pmatrix},$$ [36]

The position, velocity and acceleration vectors for the local NED frame are:

$$\mathbf{P}_n = \begin{pmatrix} x_n \\ y_n \\ z_n \end{pmatrix}, \qquad \mathbf{V}_n = \begin{pmatrix} u_n \\ v_n \\ w_n \end{pmatrix}, \qquad \mathbf{a}_n = \begin{pmatrix} a_{x,n} \\ a_{y,n} \\ a_{z,n} \end{pmatrix}$$ [36]

The velocity and acceleration vectors for the vehicle's relative coordinate system (basically a projection of the inertial local NED onto the center of mass of the mobile agent):

$$\mathbf{V}_{nv} = \begin{pmatrix} u_{nv} \\ v_{nv} \\ w_{nv} \end{pmatrix}, \qquad \mathbf{a}_{nv} = \begin{pmatrix} a_{x,nv} \\ a_{y,nv} \\ a_{z,nv} \end{pmatrix}$$ [36]

The body/sensor relative/local coordinates for velocity and acceleration are :

$$\mathbf{V}_b = \begin{pmatrix} u \\ v \\ w \end{pmatrix}, \qquad \mathbf{a}_b = \begin{pmatrix} a_x \\ a_y \\ a_z \end{pmatrix}$$ [36]

Mathematical conversion, via Euler rotation matrix, from ECEF to geodetic local NED reference frame for a mobile agent to determine position in the local NED frame:

$$\mathbf{P}_n = \mathbf{R}_{n/e}(\mathbf{P}_e - \mathbf{P}_{e,ref}), \quad [36]$$

$$\mathbf{R}_{n/e} = \begin{bmatrix} -\sin\varphi_{ref}\cos\lambda_{ref} & -\sin\varphi_{ref}\sin\lambda_{ref} & \cos\varphi_{ref} \\ -\sin\lambda_{ref} & \cos\lambda_{ref} & 0 \\ -\cos\varphi_{ref}\cos\lambda_{ref} & -\cos\varphi_{ref}\sin\lambda_{ref} & -\sin\varphi_{ref} \end{bmatrix}, \quad [36]$$

Where:

- Local NED origin is at the starting (or take off for UAV) point of the mobile agent.
- $R_{n/e}$ is the rotation matrix to convert ECEF position to local NED position
- φ_{ref} and λ_{ref} are the geodetic latitude and longitude angles, respectively, at the local NED (basically the starting point of GPS values at time 0 seconds.).
- P_n position vector of the local inertial frame NED.
- P_e position vector of ECEF coordinates.

2.3 Quaternion Rotations for Orientation

2.3.1 Background

Determining orientation of a mobile agent can be found in multiple ways. However, due to gimbal locking, where certain rotational axes of the sensor lock in place, it is ideal to calculate orientation using quaternions instead of euler transformations. Quaternions remove the ambiguity that Euler transformations have when interpolating and normalizing from one rotation to the next.

Table 5. Methods of determining orientation of a mobile agent with respect to sensor selection.

Selected sensor for Orientation Determination	Pros and Cons
Gyroscope	• Pro: Orientation is its output. • Con: Has a large drifting error as time advances.
Accelerometer	• Pro: Can use arctan(y/x) to get angles. • Con: Need to use a complementary filter with gyro.

Magnetometer	• Pro: Can convert magnetometer to find the azimuth angle, either using in combination with accelerometer or using arctan(y/x) and subtracting declination to get it in respect to true North instead of magnetic North (or heading). • Con : Accurate reading is via quaternion conversion.

Quaternions are in 4D space and are used to determine continuous orientation (pitch (or elevation), roll and yaw (or azimuth when in relation to true North)) with respect to the sensor/body frame. Quaternions are derived from the way electrons and photons behave dynamically, also known as quantum mechanics [37]. It is visualized as a sphere (or hyper-sphere) with a radius of one (like a unit circle) and is placed at the center of the 3-axial imaginary plane (ijk), where the real part of the point vector is positioned in the 4D space -either inside (if positive) or outside (if negative) -of this plane. As its name suggests, it is a vector of four, instead of three, elements. It is made up of one real part and three imaginary parts. It is defined as 'q_n' where 'n' is the count number of rotations, so if $n = 0$, then this is the original starting point of the sphere without any rotation being done, but if $n = 1$, then that means the sphere has completed one rotation about a particular angle, and so on and so forth. To do multiple rotations, one after another, that is possible with quaternions where the 'q's' are compounded (or multiplied to each other) about the point consecutively. This process of completing one rotation after another is called: conjugating rotations. It is also important to note that quaternion multiplication is non-commutative. (Please, keep in mind that 'n' can start at any arbitrary number, just as long there is a progression involved to keep track of the number of completed rotations.)

The formula for completing one rotation of a point about a sphere is:

$$p' = f(p) = q_n p q_n^{-1}$$

Where:

- $n =$ the rotation count, so the next rotation with its specified angle will be defined within q_{n+1}. n=1,2,3….etc.
- $p =$ the point in 4D space where the vector consists of one real part and 3 imaginary parts. (ex: [$w\ a\hat{i}\ b\hat{j}\ c\hat{k}$]). If the real part, 'w'= 0, then the point is along the surface of the hypersphere (because it is a unit sphere). If the 'w' > 0, then the point is inside of that hyper-sphere. If 'w'< 0, then it is positioned outside the hypersphere. The real values of

point p usually defined in a 3D xyz space will be converted as coefficients of the imaginary plane. Thus, for example if p= [1 2 3] in 3D xyz cartesian, then in quaternion space, p = [0 $1\hat{i}$ $2\hat{j}$ $3\hat{k}$].

- $f(p)\ or\ p' =$ the new point after the rotation.
- $q =$ Normalized 4D vector about the hypersphere where the real part is in the fourth dimension and the hyper-sphere is defined by the three-axial imaginary plane. Every angular input will be defined as a half angle. Thus, the 'untouched' quaternion vector is of the following format:

$$q = cos(\tfrac{\theta}{2}) + (a\hat{i} + b\hat{j} + c\hat{k})sin(\tfrac{\theta}{2})$$

For every rotation (pitch, roll and yaw) a sensor's frame is expected to undergo, the point vector is multiplied by 'q' and its inverse: 'q^{-1}'; where

$$q^{-1} = cos(-\tfrac{\theta}{2}) + (a\hat{i} + b\hat{j} + c\hat{k})sin(-\tfrac{\theta}{2}).$$

The coefficients ('a', 'b' and 'c') in the normalized quaternion vector, q, correspond to the rotating axis in the imaginary plane ($\hat{i}, \hat{j}, \hat{k}$).

Thus, a compounded expression as the following:

$$f(p) = q_3(q_2(q_1 p {q_1}^{-1}){q_2}^{-1}){q_3}^{-1}$$

Where three rotations have been done.

Furthermore, the reason why quaternion multiplication is noncommutative is due to the way complex multiplication behaves. For instance, pick any two axis, 'i' and 'k', and conduct left multiplication where $i * k$ and the direction of multiplication is from 'i' to 'k'. This means that 'i' is the line of action while 'k' is the point on the hypersphere that moves at a distance of 'i'. Thus, the final destination of point 'k' would be '-j'. This point has rotated about the unit circle on the j-k plane. The line of action 'i' is actually a unit circle, with 'i' parameters, that has been stensographed into a line along the 'i' axis. Thus, as 'i' is pulled, the point 'p' is rotated to either 'k' or 'j' on the unit circle that is at the j-k plane.

If the order of multiplication was reversed (($k * i$)to the right) then 'k' will behave as the line of action (or axis of rotation), and the point will be rotating on the unit circle that is on the i-j plane. It behaves like a 'springy pendulum' (where the ball is the hypersphere, and the hanging spring that connects the ball to a fixed joint is the line of action).

Another important note is that there are two different scenarios of how quaternions are used [38]:

1. The $f(i),\ f(j),\ and\ f(k)$, which are arbitrary marker points along the 'i', 'j' and 'k' axes, only focus on rotation and all move in sync. These points are in sync and move with respect to one degree at a time for a single rotation along the hyper-sphere.
2. The $f(i),\ f(j),\ and\ f(k)$, which are arbitrary marker points along the 'i', 'j' and 'k' axes, are not linked with respect to movement and so the equation changes to the following instead:

$$f(p) = q_{n-1}pq_n$$

Where $q_{n-1} = cos(\frac{\theta_{n-1}}{2}) + (a\hat{i} + b\hat{j} + c\hat{k})sin(\frac{\theta_{n-1}}{2})$ which will be the 1st rotation that the hyper-sphere undergoes at a particular angle.

Followed by $q_n = cos(\frac{\theta_n}{2}) + (a\hat{i} + b\hat{j} + c\hat{k})sin(\frac{\theta_n}{2})$, which is the 2nd rotation for the hyper-sphere at a different angle.

The overall size of the hyper-sphere also changes in this mode.

2.3.2 Implementation of the quaternion towards the magnetometer and accelerometer

The scope of this study will primarily focus on the yaw (or azimuth) angle, since the basic analysis will focus upon a 2D cartesian plane that considers translation and orientation of a mobile agent. It is critical to note which local tangential/topocentric plane - either NED or ENU - will be used. For in AHRS, the quaternion derivations for their INS systems are dependent on which axis the different orientations: elevation (pitch), roll and yaw (azimuth) occurs.

Thus, assuming the local NED frame, in order to determine the azimuth angle, using quaternions with respect to sensor data, the magnetometer and accelerometer values are necessary inputs. Furthermore, the azimuth is not computed without the computation of the pitch and roll. (For simplicity's sake, whenever pitch, roll and yaw are referred to within a mathematical context, the three angles will be distinguished by the following characters: θ, ϕ, ψ, respectively.) The order of the rotation angles can be arbitrary (i.e yaw can be computed before roll and pitch…etc.)

The following steps show how to find the elevation quaternion in the NED frame:

1. Define quaternion vector with respect to rotation axis for pitch (which is the y-axis in the NED frame):

$$\mathbf{q} = \begin{pmatrix} \cos\frac{\beta}{2} \\ u_x \sin\frac{\beta}{2} \\ u_y \sin\frac{\beta}{2} \\ u_z \sin\frac{\beta}{2} \end{pmatrix}$$ [39] and $\mathbf{u} = [u_x\ u_y\ u_z]^T$ [39]

Where β = rotation angle, q = generic quaternion, and u = the rotation axis.

2. Normalize the accelerometer outputs:

   In an accelerometer, the z-axis is the only axis that senses gravitational acceleration : g = -9.8m/s^2 when there is no orientation (no elevation and roll done).
   However, once there is an elevation in the NED frame:

   $a_x = g sin\theta$[39] $a_z = -g cos\theta$[34] $a = [a_x\ a_y\ a_z]^T$ [39]

   Where 'a' is the accelerometer vector of values.

   $\mathbf{a} = \frac{a}{||a||}$[39]

   Then get the pitch angle from : $sin\theta = a_x$ [39]

3. Calculate the half angles:

   $sin\frac{\theta}{2} = sgn(sin\theta)\sqrt{\frac{1-cos\theta}{2}}$, $cos\frac{\theta}{2} = \sqrt{\frac{1+cos\theta}{2}}$ [39]

   Where sgn(x), where 'x' is a real number :

$$\operatorname{sgn} x := \begin{cases} -1 & \text{if } x < 0, \\ 0 & \text{if } x = 0, \\ 1 & \text{if } x > 0. \end{cases}$$

4. Place them in the elevation quaternion:

$$\mathbf{q}_e = \begin{pmatrix} \cos\frac{\theta}{2} \\ 0 \\ \sin\frac{\theta}{2} \\ 0 \end{pmatrix}$$ [39]

The following steps show how to find the roll quaternion in the NED frame:

1. Introducing a roll angle changes the equations to:

$a_y = -gcos(\theta)sin(\phi)$, $a_z = -gcos(\theta)sin(\phi)$ [39] , but since magnitude = g and when normalized is equal to '1', thus it is often omitted. Thus, final expression is:

$$a_y = -cos\theta sin\phi \ , \quad a_z = -cos\theta cos\phi \quad [39]$$

2. If $cos(\theta) \neq 0$, then $sin(\phi)\ and\ cos(\phi)$ are:

$$sin\phi = -\frac{a_y}{cos\theta} \ , \ cos\phi = -\frac{a_z}{cos\theta} \quad [39]$$

3. However, if $cos(\theta) = 0$ and the roll angle, ϕ, is too small (undefined) or is zero, then it would be a similar set up to that of the 1st elevation reading's quaternion:

$$\mathbf{q}_r = \begin{pmatrix} \cos\frac{\phi}{2} \\ \sin\frac{\phi}{2} \\ 0 \\ 0 \end{pmatrix} \quad [39]$$

The following steps show how to find the azimuth quaternion in the NED frame:

1. Normalize the magnetic readings (because a quaternion's coefficients, when summed up equate to 1) and rotate the normalized magnetic readings through the elevation and roll quaternions :

$$\mathbf{m} = \frac{m}{||m||} [39] \qquad {}^e\mathbf{m} = \mathbf{q}_e\mathbf{q}_r\,{}^b\mathbf{m}\mathbf{q}_r^{-1}\mathbf{q}_e^{-1} \quad [39]$$

Where the magnetometer's values, in the body frame, are written in quaternion space:

$${}^b\mathbf{m} = \begin{pmatrix} 0 & {}^bm_x & {}^bm_y & {}^bm_z \end{pmatrix} \quad [39]$$

2. Use the normalized known local geomagnetic field to normalize the magnetometer's values, except for the azimuth, and then find azimuth.:

$\mathbf{n} = [n_x\ n_y\ n_z]$[39] , where 'n' is the normalized known local geomagnetic field vector.

$$\begin{bmatrix} n_x \\ n_y \end{bmatrix} = \begin{bmatrix} \cos\psi & -\sin\psi \\ \sin\psi & \cos\psi \end{bmatrix} \begin{bmatrix} {}^em_x \\ {}^em_y \end{bmatrix} \quad [39] \text{ , non-normalized form.}$$

Next, using the following normalization equations :

$$\begin{bmatrix} N_x \\ N_y \end{bmatrix} = \frac{1}{\sqrt{n_x^2+n_y^2}} \begin{bmatrix} n_x \\ n_y \end{bmatrix}$$
$$\begin{bmatrix} M_x \\ M_y \end{bmatrix} = \frac{1}{\sqrt{{}^e m_x^2+{}^e m_y^2}} \begin{bmatrix} {}^e m_x \\ {}^e m_y \end{bmatrix}$$ [39]

$$\begin{bmatrix} N_x \\ N_y \end{bmatrix} = \begin{bmatrix} \cos\psi & -\sin\psi \\ \sin\psi & \cos\psi \end{bmatrix} \begin{bmatrix} M_x \\ M_y \end{bmatrix}$$ [39] , normalized form, on both sides.

$$\begin{bmatrix} \cos\psi \\ \sin\psi \end{bmatrix} = \begin{bmatrix} M_x & M_y \\ -My & M_x \end{bmatrix} \begin{bmatrix} N_x \\ N_y \end{bmatrix}$$ [39], use this matrix to solve for the azimuth angle.

3. Estimate the half angle values, as in the earlier rotations, and define the quaternion for azimuth:

$$\mathbf{q}_a = \begin{pmatrix} \cos\frac{\psi}{2} \\ 0 \\ 0 \\ \sin\frac{\psi}{2} \end{pmatrix}$$ [39]

Table 6. The Orientation Quaternions [39].

Orientation (degrees)	Quaternion
Elevation (pitch) (rotation about the y-axis)	$\mathbf{q}_e = \begin{pmatrix} \cos\frac{\theta}{2} \\ 0 \\ \sin\frac{\theta}{2} \\ 0 \end{pmatrix}$
Roll (rotation about the x-axis)	$\mathbf{q}_r = \begin{pmatrix} \cos\frac{\phi}{2} \\ \sin\frac{\phi}{2} \\ 0 \\ 0 \end{pmatrix}$

Azimuth (yaw) (rotation about the z-axis)	$\mathbf{q}_a = \begin{pmatrix} \cos\frac{\psi}{2} \\ 0 \\ 0 \\ \sin\frac{\psi}{2} \end{pmatrix}$

The final quaternion multiplies each orientation's quaternion to determine the final orientation of the rigid body:

$$\mathbf{q} = q_a q_e q_r \quad [39]$$

Chapter 3 - Kalman Filtering in Modern Control Systems Design Theory

3.1 Deriving the Parameters for the Kalman Filter

3.1.1 Background

Control systems design theory is divided between two main categories: classical and modern. The classical mode interprets the open or closed system via the transfer functions (output vs. input) for both controller and plant (or physical system), while the modern mode uses the state space representation matrices. The latter can be converted to and from the transfer functions calculated. The kalman filter is the estimator that converts an open loop system to a closed one and helps modulate the response of the system's desired parameter through error reduction. It is critical to note that the kalman filter is not a closed loop controller, but rather an estimator/observer model. The purpose of this chapter is to present the overall setup of the kalman filter using modern control systems design theory.

3.1.2 FBD and State Space Representation

For theoretical simplification, the mobile agent's relative orientation and position will be in the context of a 2D cartesian plane. The movement of the mobile agent (its relative body/sensor frame) will be with respect to the local inertial NED/ENU vehicular frame (depending on the sensor's axial orientation). Since the GPS sensor data outputs geodetic coordinates, (mathematically) it must be converted to ECEF and then to the local NED/ENU frame. Once the equations of motion have been discretized, the position values of the local NED/ENU frame will make up the state variables for the state space representation matrix, while the accelerometer's values along the x and y axes make up the input state variables.
Hand calculations:

Setting up the FBD with respect to the mobile agent's position and orientation.

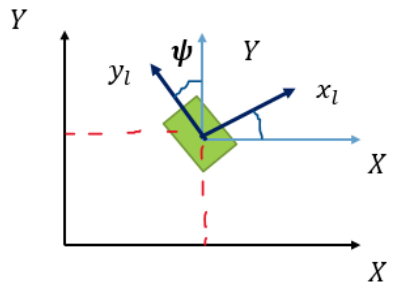


Figure 25: *FBD of ENU body frame. Following the standard right handed rule where the positive angles move in counter clockwise rotations. X , Y make up the vehicular fixed inertial frame (projected from the global ENU frame), while x_l, y_l are axes of the mobile or body frame.*

Linear acceleration and its noise:

$$a_x = \ddot{x}_l + n_{accx} \qquad a_y = \ddot{y}_l + n_{accy} \quad (1)$$

Yaw angle and its noise, which can be collected via matlab's libraries using the magnetometer and accelerometer data (or mathematically via the quaternion's transformation with respect to the magnetometer):

$$\boldsymbol{\psi} = \boldsymbol{\psi} + n_{\psi} \quad (2)$$

Second derivation involves the position equations, with respect to FBD, to convert to acceleration.

$$X_{GPS} = X + n_{GPS} \longrightarrow \ddot{X}_{GPS} = \ddot{X} + n_{GPS}$$

$$Y_{GPS} = Y + n_{GPS} \longrightarrow \ddot{Y}_{GPS} = \ddot{Y} + n_{GPS} \quad (3)$$

X_{GPS}, Y_{GPS} are data points from GPS data. GPS data is in longitude, latitude and altitude (lla) format, thus needing to convert to either ENU or NED local/global frame's xyz cartesian coordinates. The reference frame of choice is dependent on the overall body frame's x-y-z orientation of the device. This inertial reference frame was also denoted as the vehicular frame (it remains fixed at the center of mass of the mobile device).

Due to the standard-right handed rule, where the counter-clockwise (CCW) angular rotation, $\boldsymbol{\psi}$, the new GPS/NED/ENU frame will be calculated using Euler Rotation Matrices (or Transformations). However, since this study is interested in only the yaw angle, only one of the directional cosine matrices (DCMs) will be formulated. Typically there are a total of three DCMs, which are the rotation matrices about either the x, y or z axes (roll, pitch and yaw).

The NASA euler angle sequence is: yaw, pitch and roll; where depending on the mobile device's reference frame can either be: Z-X-Y (ENU) or Z-Y-X (NED). This is also known as the Tait-Bryan (intrinsic) angles. There are two different projection approaches when deriving the rotation matrix for each euler angle based rotation. Can either project on to the original frame's axes (X , Y) (where x_l, y_l are treated as the hypotenuse) or the new rotated frame's axes (x_l, y_l) (where X , Y are treated as the hypotenuse). Either methodology will yield the same final

equations. The DCM about the z axis (CCW rotation matrix for yaw angle, φ, with respect to the ENU reference frame) to transform from the original to the new axis is:

	X	Y	Z
x_l	cos(psi)	sin(psi)	0
y_l	-sin(psi)	cos(psi)	0
z_l	0	0	1

(3.1)

Note the older frame on top and the newer frame on the left side. The z axis remains the same because the new reference (or body) frame is rotating about the z axis of the vehicle (projected inertial/fixed ENU) frame. The equations are formulated across for CCW rotations, while vertical for CW rotations. Note that the CW rotations would have a different looking table, but should end up with the same equations. Keep in mind that the equations are positions and that the signs and trigonometric coefficients within either axes are dependent on what reference frame is being used (ENU or NED) or rather where the axial projections are at.

The complete rotation with respect to all axes is the multiplication of all the DCMs, thus: $R = R_X R_Y R_Z$. However, the only rotation of concern is about the z axis only; thus $R = R_z$.

Substitute in acceleration within the new frame's (body's) position based equations:

$$X = x_l \cos(\varphi) + y_l \sin(\varphi) \quad (3.2)$$

$$\ddot{X} = \ddot{x}_l \cos(\varphi) + \ddot{y}_l \sin(\varphi)$$

$$\ddot{X} = (a_x - n_{accx}) \cos(\varphi) + (a_y - n_{accy}) \sin(\varphi)$$

Distribute it out:

$$\ddot{X} = a_x \cos(\varphi) - n_{accx} \cos(\varphi) + a_y \sin(\varphi) - n_{accy} \sin(\varphi) \quad (4)$$

$$Y = -x_l \sin(\varphi) + y_l \cos(\varphi) \quad (3.3)$$

$$\ddot{Y} = -\ddot{x}_l \sin(\varphi) + \ddot{y}_l \cos(\varphi)$$

$$\ddot{Y} = -(a_x - n_{accx}) \sin(\varphi) + (a_y - n_{accy}) \cos(\varphi)$$

Distribute it out:

$$\ddot{Y} = -a_x \sin(\varphi) + n_{accx} \sin(\varphi) + a_y \cos(\varphi) - n_{accy} \cos(\varphi) \quad (5)$$

Next, is to discretize/linearize using 2nd order discretization.
Where:

$$\ddot{x}(t) = \frac{x(k+1) - 2x(k) + x(k-1)}{\Delta t^2} \quad (6)$$ and $t=k\Delta t$; thus $k=t/\Delta t$.

Substitute all 2nd order terms, in equations (4) and (5), with the discretized equation (6). (Highest order to one side of the equation)

$$X(k+1) = 2X(k) - X(k-1) + \Delta t^2 (a_x \cos(\varphi) - n_{accx} \cos(\varphi) + a_y \sin(\varphi) - n_{accy} \sin(\varphi)) \quad (7)$$

$$Y(k+1) = 2Y(k) - Y(k-1) + \Delta t^2 (-a_x \sin(\varphi) + n_{accx} \sin(\varphi) + a_y \cos(\varphi) - n_{accy} \cos(\varphi)) \quad (8)$$

Next, insert the equations, (7) and (8), into the A,B, C and G matrices that will be used by the estimator (Kalman filter with Observer 'L').

Discontinuous version:

$$\boldsymbol{q(k+1) = Aq(k) + Bu(k) + Gw(k)} \quad (9)$$

$$\boldsymbol{z(k) = Cq(k) + Du(k) + v(k)} \quad (10)$$

Where:

A	q	B	u	G	w
State transition matrix	State variable	Control coefficient matrix	input	Noise coefficient matrix	Noise of input

z	C	q	D	v
Observable/measurable state variable outputs	Measurement matrix for states related to the output	State variable	'D' is usually a zero matrix	Noise of output

Thus, now converting equations (7) and (8) into the state space representation equations, (9) and (10) , of the discontinuous Kalman filter.

$$X(k+1) = 2X(k) - X(k-1) + \Delta t^2(a_x \cos(\varphi) - n_{accx} \cos(\varphi) + a_y \sin(\varphi) - n_{accy} \sin(\varphi)) \quad (7)$$

For X:

$$\begin{bmatrix} X(k) \\ X(k+1) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -1 & 2 \end{bmatrix} \begin{bmatrix} X(k-1) \\ X(k) \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ \Delta t^2 \cos(\varphi) & \Delta t^2 \sin(\varphi) \end{bmatrix} \begin{bmatrix} a_x(k) \\ a_y(k) \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ -\Delta t^2 \cos(\varphi) & -\Delta t^2 \sin(\varphi) \end{bmatrix} \begin{bmatrix} n_{accx}(k) \\ n_{accy}(k) \end{bmatrix} \quad (11)$$

Measured output:

$$z_X(k) = X_{GPS}(k) = \begin{bmatrix} 0 & 1 \end{bmatrix} \begin{bmatrix} X(k-1) \\ X(k) \end{bmatrix} + n_{GPS}(k) \quad (12)$$

For Y:

$$Y(k+1) = 2Y(k) - Y(k-1) + \Delta t^2(-a_x \sin(\varphi) + n_{accx} \sin(\varphi) + a_y \cos(\varphi) - n_{accy} \cos(\varphi)) \quad (8)$$

$$\begin{bmatrix} Y(k) \\ Y(k+1) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -1 & 2 \end{bmatrix} \begin{bmatrix} Y(k-1) \\ Y(k) \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ -\Delta t^2 \sin(\varphi) & \Delta t^2 \cos(\varphi) \end{bmatrix} \begin{bmatrix} a_x(k) \\ a_y(k) \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ \Delta t^2 \sin(\varphi) & -\Delta t^2 \cos(\varphi) \end{bmatrix} \begin{bmatrix} n_{accx}(k) \\ n_{accy}(k) \end{bmatrix} \quad (13)$$

$$z_Y(k) = Y_{GPS}(k) = \begin{bmatrix} 0 & 1 \end{bmatrix} \begin{bmatrix} Y(k-1) \\ Y(k) \end{bmatrix} + n_{GPS}(k) \quad (14)$$

Now, since in the results section 3.2, it indicates that the static noise plot distribution is that of a Gaussian distribution, also the variances of the accelerometer data (along the x and y axes) are equal. Thus, statistically, terms that have equal variances and are added with one sine and the other is a cosine coefficient will equal to a scalar value of equal variance - regardless of angle. Thus, the nonlinear terms can be removed (for both, X and Y representations).

$$\begin{bmatrix} X(k) \\ X(k+1) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -1 & 2 \end{bmatrix} \begin{bmatrix} X(k-1) \\ X(k) \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ \Delta t^2 \cos(\varphi) & \Delta t^2 \sin(\varphi) \end{bmatrix} \begin{bmatrix} a_x(k) \\ a_y(k) \end{bmatrix} + \begin{bmatrix} 0 \\ \Delta t^2 \end{bmatrix} n_{acc}(k) \quad (15)$$

$$\begin{bmatrix} Y(k) \\ Y(k+1) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -1 & 2 \end{bmatrix} \begin{bmatrix} Y(k-1) \\ Y(k) \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ -\Delta t^2 sin(\varphi) & \Delta t^2 cos(\varphi) \end{bmatrix} \begin{bmatrix} a_x(k) \\ a_y(k) \end{bmatrix} + \begin{bmatrix} 0 \\ \Delta t^2 \end{bmatrix} n_{acc}(k) \quad (16)$$

Thus, the full Kalman filter with the observer 'L', along the X and Y axes, will be of the following form:

$$\hat{q}(k+1) = A\hat{q}(k) + Bu(k) + L(z(k) - C\hat{q}(k)) \quad (17)$$

$$\begin{bmatrix} \hat{X}(k) \\ \hat{X}(k+1) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -1 & 2 \end{bmatrix} \begin{bmatrix} \hat{X}(k-1) \\ \hat{X}(k) \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ \Delta t^2 \cos(\varphi) & -\Delta t^2 \sin(\varphi) \end{bmatrix} \begin{bmatrix} a_x(k) \\ a_y(k) \end{bmatrix} + L\left(X_{GPS}(k) - \begin{bmatrix} 0 & 1 \end{bmatrix} \begin{bmatrix} \hat{X}(k-1) \\ \hat{X}(k) \end{bmatrix} \right) \quad (18)$$

$$\begin{bmatrix} \hat{Y}(k) \\ \hat{Y}(k+1) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -1 & 2 \end{bmatrix} \begin{bmatrix} \hat{Y}(k-1) \\ \hat{Y}(k) \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ -\Delta t^2 sin(\varphi) & \Delta t^2 cos(\varphi) \end{bmatrix} \begin{bmatrix} a_x(k) \\ a_y(k) \end{bmatrix} + L\left(Y_{GPS}(k) - \begin{bmatrix} 0 & 1 \end{bmatrix} \begin{bmatrix} \hat{Y}(k-1) \\ \hat{Y}(k) \end{bmatrix} \right) \quad (19)$$

Note: In MATLAB: L = dlqe(A,G,C,Q (variance noise with respect to the input states), R (variance of the noise with respect to GPS's measured output). Q and R are square matrices (where the squared variance values of the collected device's after it's in motion are positioned diagonally), and are tunable. Related code and the final matrices can be viewed in Appendix A.

3.2 Initial Implementation results in Matlab and QGIS

Initial data collected from a smartphone for the first test run to be implemented into the Kalman observer model. (All code can be found in the github repository under Appendix A)

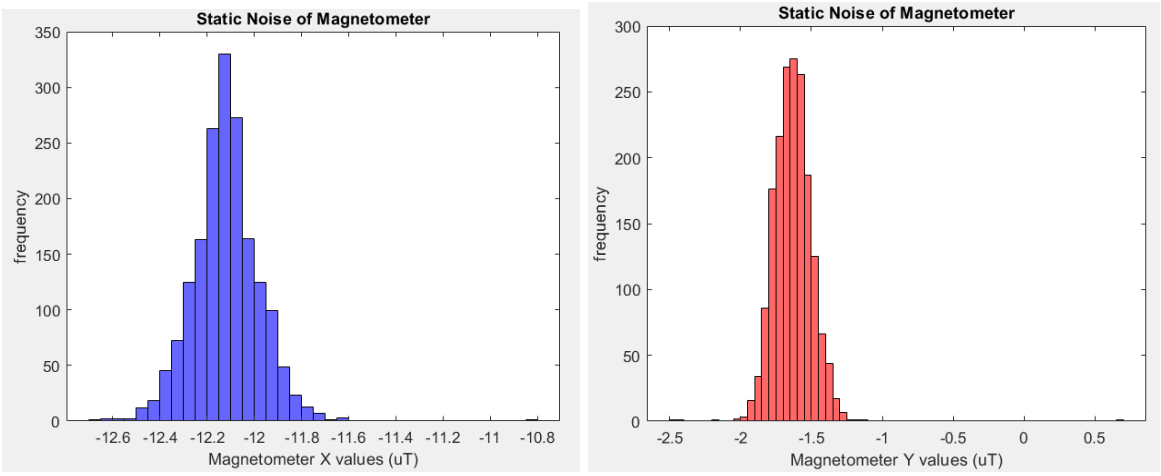


Figure 26: *Gaussian distribution of one of the sensor data's static noise. Smartphone is just sitting on a table. All the other sensors (accelerometer, gyroscope and GPS) underwent the same test.*

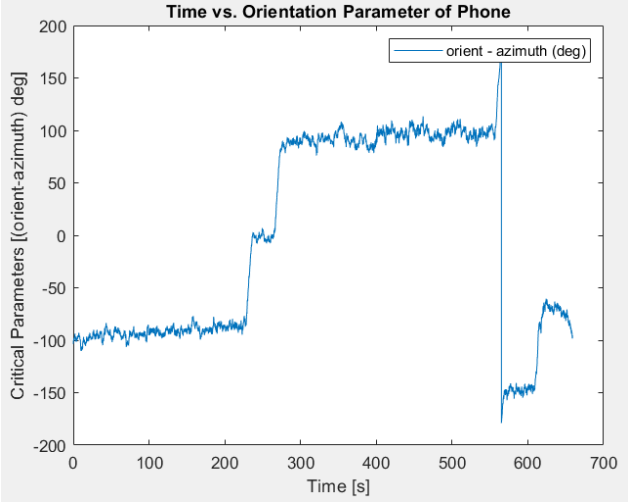


Figure 27: *Time vs Orientation (azimuth angle) from smartphone when it is in motion. Keep in mind that orientation is a fusion of gyro+accelerometer+magnetometer and it only relates to the orientation of the device or sensor itself. The yaw or azimuth angle given is inaccurate for it is not with respect to the actual magnetic or true north of one's geographic location.*

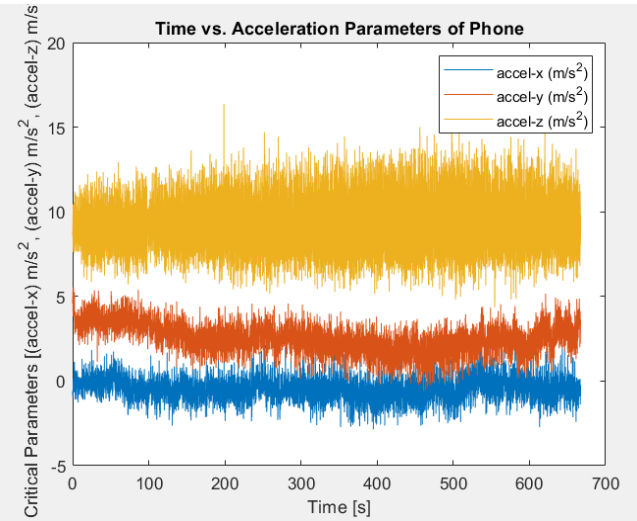


Figure 28: *Acceleration vs. time of the smartphone as it is in motion.*

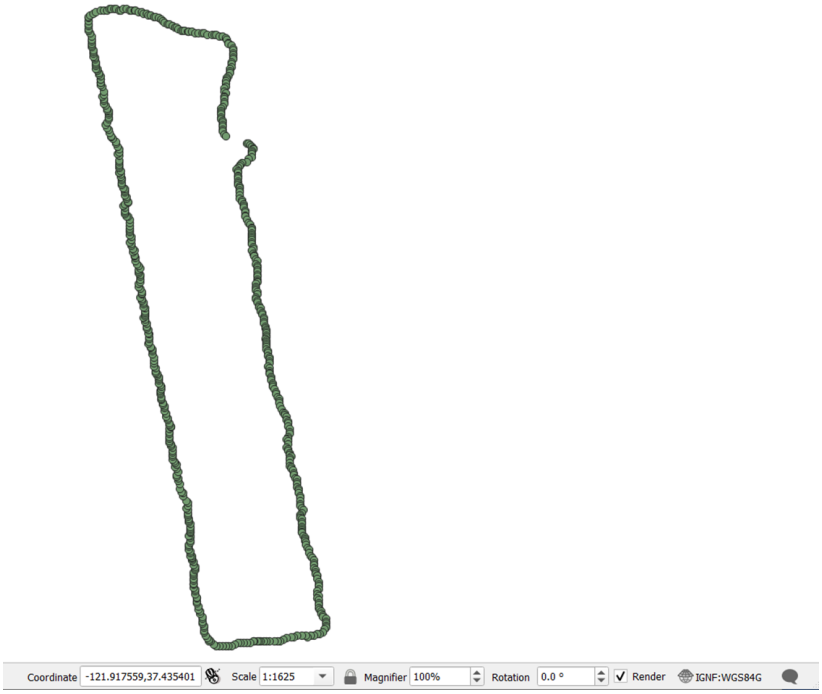


Figure 29: *Plot geodetic coordinates into QGIS. Layer established after converting lla to UTM. This is the reference path for the Kalman filter.*

Chapter 4 - Theory Application and Approach

4.1 Reference Frames and the Kalman Filter's Inputs and Outputs

4.1.1 Background

The Kalman filter's state space representation model designed in section 3.1.2 has the accelerometer and azimuth angles as inputs and the GPS's current and previous steps as outputs. It is critical to note that the Kalman filter is not a controller but an observer, for its purpose is to find/observe the hidden measurable states. This attribute is useful for multi-input multi-output (MIMO) systems. The Kalman filter consists of two processes:

1. predicts the hidden states of the system
2. Uses noisy measurements to refine the estimated states.

In this project, the hidden states are X_{GPS} *and* Y_{GPS} at the smaller intervals that a GPS's receiver cannot relay due to the GPS satellite's low frequency (resolution). While the noisy instruments are the accelerometer and magnetometer data (due to their higher frequencies). Thus, this chapter will delve into, and continue from Chapter 2, the sensors and the mathematical processes used to compute the inputs in order to yield the expected outputs in the final frame of reference.

4.1.2 Sensors and Vehicular vs. Body Frames

The units of the GPS's raw data is in *lla* geodetic format and outputs: longitude [dd], latitude [dd] and altitude[m] - within that order (and sometimes speed). The geodetic format is one geographical frame of reference, and can, thus, be converted to either UTM, ECEF or NED/ENU local reference frames. The body's local reference frame is dependent on the axes of the sensor's position. The figure below shows an example of a sensor's frame being of the NED orientation. Fortunately, this is not a concern with respect to GPS, thus can select any local body frame.

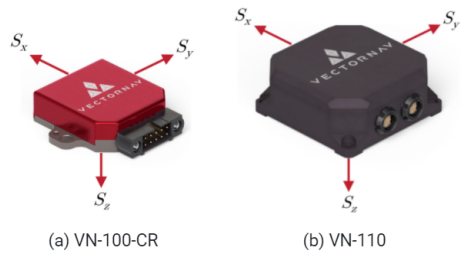


Figure 30: *Sensor's reference frame [40].*

Since, the phone was used in the initial gathering of the data and will be used initially to test the Kalman filter's model, it is important to note that the phone's sensors' orientation is ENU

(East-North-Up) with respect to the x-y-z axes. Thus, the natural orientation of the phone is when it is in an upright position. However, due to the nature of a mobile land rover, where the navigation system will be placed horizontally, the phone was placed horizontally (as it was in motion) during data collection.

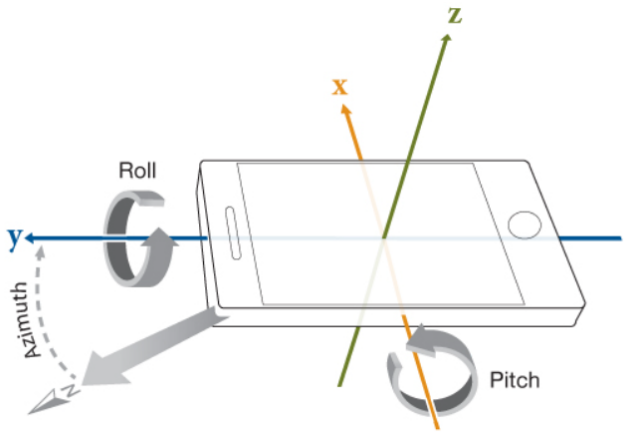


Figure 31: *Phone orientation for data collection[41]. Azimuth angle is determined by both the magnetometer and gyroscope. Roll and pitch are other angular orientations determined by the integration of the accelerometer and magnetometer. Phone's local reference frame is in ENU.*

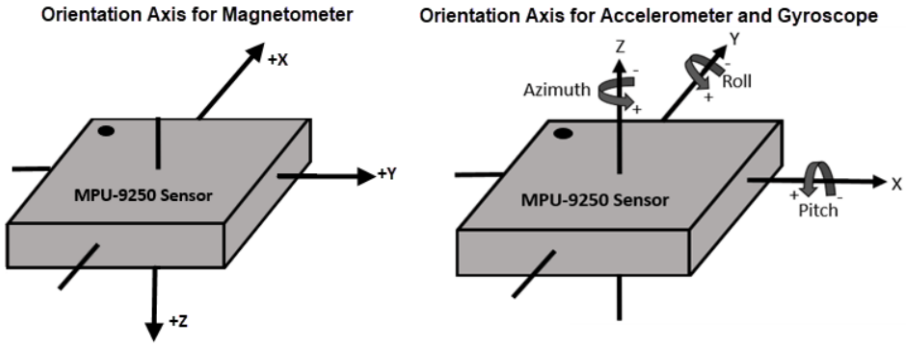


Figure 32: *Orientation axes for sensors in the MPU-9250 [42]. Note the NED orientation of the magnetometer and the shared axes of the accelerometer and gyroscope. The accelerometer helps in determining which is 'right side up' (it measures at which axis the gravitational force is pushing) , the gyroscope determines the rate and rotation degree of an object (think of it as a spinning disk that moves in correlation with ones motion) and the magnetometer behaves as a compass and helps determine where the device is at with respect to 'Magnetic North' in comparison to 'True North'.*

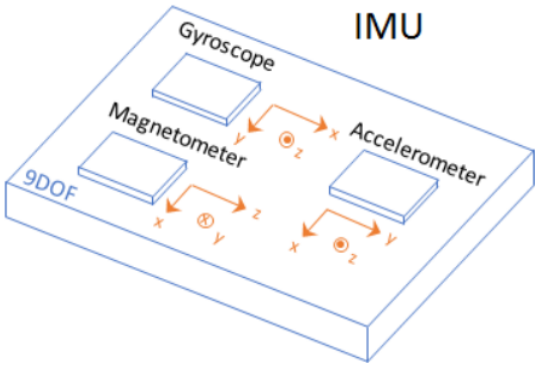


Figure 33: *IMU/MPU 9-DOF physical construct [43].*

The advantage of using a local body reference frame is that the starting point of the vehicle can be defined as the origin. Thus, local tracking of the initial and end points of a rover, or any mobile vehicle, are feasible.

Hence, the following equations depict the relationship between the locally projected NED reference frame and the NED body frame (of the moving vehicle).

First the mobile vehicle's vectors defined:

$$\mathbf{V}_{\mathrm{b}} = \begin{pmatrix} u \\ v \\ w \end{pmatrix} \quad [36]$$

$$\mathbf{a}_{\mathrm{b}} = \begin{pmatrix} a_{\mathrm{x}} \\ a_{\mathrm{y}} \\ a_{\mathrm{z}} \end{pmatrix} \quad [36]$$

Where V_b *is the velocity vector of the body* and
a_b *is the acceleration vector of the moving body.*

Next, an Euler rotation matrix with respect to the mobile vehicle's movement about the z-axis (azimuth angle) is visually represented. (This is similar to the FBD shown in section 3.1.2 to derive the Kalman filter's state space representation of the model.)

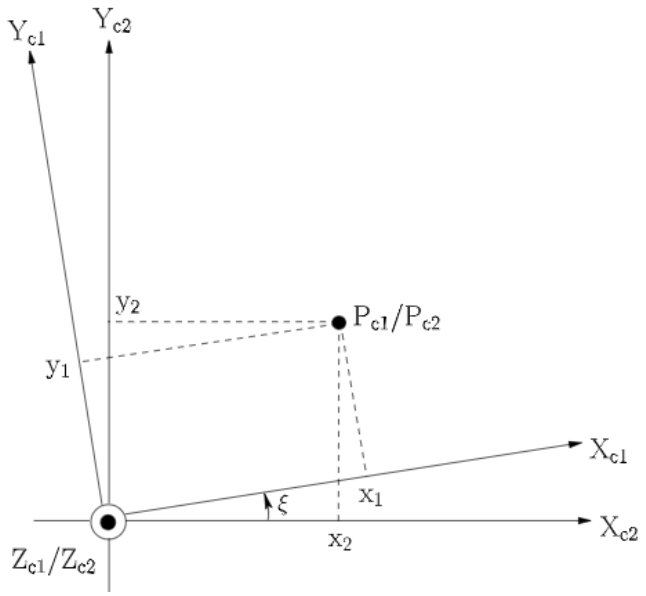


Figure 34: *Euler rotation representation [36]. C1 is the new reference frame after C2 rotates ξ rotated degrees (azimuth angle) about the Z axis. Note that the position, P, of the mobile vehicle is the same, in either reference frame, despite the rotation taking place.*

The position of the rover after each rotation will be depicted mathematically as:

$$\mathbf{P}_{C1} = \mathbf{R}_{C1/C2}\mathbf{P}_{C2}, \qquad [36]$$

Where the rotation matrix is defined as:

$$\mathbf{R}_{C1/C2} = \begin{bmatrix} \cos\xi & \sin\xi & 0 \\ -\sin\xi & \cos\xi & 0 \\ 0 & 0 & 1 \end{bmatrix}. \qquad [36]$$

$$\mathbf{R}_{C2/C1} = \mathbf{R}^{-1}_{C1/C2} = \mathbf{R}^{T}_{C1/C2}. \qquad [36]$$

There are other rotation matrices and intermediate matrices (rotations due to successive angular changes in the context of pitch and roll for instance) that the vehicle can go through. However, this study will just focus on the changing of the yaw angle.

The angular velocity vector is defined as follows:

$$\boldsymbol{\omega}^{*}_{C1/C2} = \begin{pmatrix} \omega_x \\ \omega_y \\ \omega_z \end{pmatrix}, \qquad [36] \qquad \boldsymbol{\omega}^{*}_{C2/C1} = -\boldsymbol{\omega}^{*}_{C1/C2}. \qquad [36]$$

The kinematic rate of change on mobile trajectories (with respect to the vehicle's NED carried inertial reference frame [nv]) derivations with respect to the geodetic parameters are listed below:

$$\dot{\lambda} = \frac{v_{\mathrm{nv}}}{(N_{\mathrm{E}} + h)\cos\varphi},$$
$$\dot{\varphi} = \frac{u_{\mathrm{nv}}}{M_{\mathrm{E}} + h},$$
$$\dot{h} = -w_{\mathrm{nv}}. \qquad [36]$$

Where the longitudinal and latitudinal equations are derived from spherical triangles.

The change in velocities with respect to the vehicle carried NED frame (nv) are as follows:

$$\dot{u}_{\mathrm{nv}} = -\frac{v_{\mathrm{nv}}^2 \sin\varphi}{(N_{\mathrm{E}} + h)\cos\varphi} + \frac{u_{\mathrm{nv}} w_{\mathrm{nv}}}{M_{\mathrm{E}} + h} + a_{\mathrm{mx,nv}},$$
$$\dot{v}_{\mathrm{nv}} = \frac{u_{\mathrm{nv}} v_{\mathrm{nv}} \sin\varphi}{(N_{\mathrm{E}} + h)\cos\varphi} + \frac{v_{\mathrm{nv}} w_{\mathrm{nv}}}{N_{\mathrm{E}} + h} + a_{\mathrm{my,nv}},$$
$$\dot{w}_{\mathrm{nv}} = -\frac{v_{\mathrm{nv}}^2}{N_{\mathrm{E}} + h} - \frac{u_{\mathrm{nv}}^2}{M_{\mathrm{E}} + h} + g + a_{\mathrm{mz,nv}}, \qquad [36]$$

Where 'g' is gravitational acceleration and the proper measured acceleration of the vehicle carried NED frame (nv) is defined as follows:

$$\mathbf{a}_{\mathrm{mea,nv}} = \begin{pmatrix} a_{\mathrm{mx,nv}} \\ a_{\mathrm{my,nv}} \\ a_{\mathrm{mz,nv}} \end{pmatrix} \qquad [36]$$

It is important to understand the difference between the local NED body frame and the vehicular inertial NED carried frame. The latter incorporates rate of change of the geodetic parameters (λ', ψ' *and* h') and translational kinematics (i.e. velocity and acceleration).

Equations below derive the relationships between the local body NED frame and the vehicular carried NED frame.

$$\mathbf{V}_{\mathrm{b}} = \mathbf{R}_{\mathrm{b/nv}}\mathbf{V}_{\mathrm{nv}},$$
$$\mathbf{a}_{\mathrm{b}} = \mathbf{R}_{\mathrm{b/nv}}\mathbf{a}_{\mathrm{nv}},$$

$$\mathbf{a}_{\mathrm{mea,b}} = \mathbf{R}_{\mathrm{b/nv}}\mathbf{a}_{\mathrm{mea,nv}}, \quad [36]$$

Where 'b' is body, 'nv' is normal vehicular and 'mea' is measured.

The rotational matrix from the vehicle carried NED frame to the body reference frame:

$$\mathbf{R}_{\mathrm{b/nv}} = \begin{bmatrix} \mathbf{c}_\theta\mathbf{c}_\psi & \mathbf{c}_\theta\mathbf{s}_\psi & -\mathbf{s}_\theta \\ \mathbf{s}_\phi\mathbf{s}_\theta\mathbf{c}_\psi - \mathbf{c}_\phi\mathbf{s}_\psi & \mathbf{s}_\phi\mathbf{s}_\theta\mathbf{s}_\psi + \mathbf{c}_\phi\mathbf{c}_\psi & \mathbf{s}_\phi\mathbf{c}_\theta \\ \mathbf{c}_\phi\mathbf{s}_\theta\mathbf{c}_\psi + \mathbf{s}_\phi\mathbf{s}_\psi & \mathbf{c}_\phi\mathbf{s}_\theta\mathbf{s}_\psi - \mathbf{s}_\phi\mathbf{c}_\psi & \mathbf{c}_\phi\mathbf{c}_\theta \end{bmatrix}, \quad [36]$$

Conversions of the angular velocity vector from the vehicle carried inertial NED reference frame to the local body's reference frame is defined as follows (also known as the Euler kinematical equations):

$$\begin{aligned} \boldsymbol{\omega}^{\mathrm{b}}_{\mathrm{b/nv}} &:= \begin{pmatrix} p \\ q \\ r \end{pmatrix} = \begin{pmatrix} \dot{\phi} \\ 0 \\ 0 \end{pmatrix} + \mathbf{R}_{\mathrm{b/int2}}\left[\begin{pmatrix} 0 \\ \dot{\theta} \\ 0 \end{pmatrix} + \mathbf{R}_{\mathrm{int2/int1}}\begin{pmatrix} 0 \\ 0 \\ \dot{\psi} \end{pmatrix}\right] \\ &= \mathbf{S}\begin{pmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{pmatrix}, \end{aligned} \quad [36]$$

Where R_{int} is the 'intermediate' body frame with respect to the changes in yaw, pitch and roll angle switching sequences, respectively:

$$\mathbf{R}_{\mathrm{int1/nv}} = \begin{bmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix},$$

$$\mathbf{R}_{\mathrm{int2/int1}} = \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix},$$

$$\mathbf{R}_{\mathrm{b/int2}} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & -\sin\phi & \cos\phi \end{bmatrix}. \quad [36]$$

For simplicity, the lumped sum rotation matrix, $R_{b/nv}$, is given below:

$$\mathbf{S} = \begin{bmatrix} 1 & 0 & -\sin\theta \\ 0 & \cos\phi & \sin\phi\cos\theta \\ 0 & -\sin\phi & \cos\phi\cos\theta \end{bmatrix}. \quad [36]$$

$$\mathbf{S}^{-1} = \begin{bmatrix} 1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi/\cos\theta & \cos\phi/\cos\theta \end{bmatrix}. \quad [36]$$

It is critical to note that $sin(\theta) = \pm 90^{\circ}$ will cause singularities and can only be avoided via the usage of quaternions.

Since, the local NED body frame equivocates to the vehicle carried NED frame's parameters throughout this study, it is reasonable to conclude that:

$$\mathbf{V}_n = \mathbf{V}_{nv}, \qquad \boldsymbol{\omega}^b_{b/n} = \boldsymbol{\omega}^b_{b/nv}, \qquad \mathbf{a}_n = \mathbf{a}_{nv}, \qquad \mathbf{a}_{mea,n} = \mathbf{a}_{mea,nv}, \quad [36]$$

Note that the rotation matrices would slightly differ for an ENU frame, since the x and y axes are switched, and that the standard rotation matrices given are usually assuming the standard-right-hand-rule where a counterclockwise (ccw) angular rotations are positive. Clockwise (cw) rotations are negative and are following the left-hand-rule. It is important to remain consistent and aware of which mathematical rules are being applied.

Also, a simpler way of visualizing the different transitions between reference frames is to understand that the order goes from macro (lla - geodetic coordinates) to micro (local body frame) or vice versa.

4.1.3 Pre-processing Sensor Data in Matlab

The next steps are to setup Matlab scripts in converting GPS's raw data (in *lla* format) to xyz-ENU frame, deriving the yaw angle from the magnetometer and accelerometer data, as well as interpolating the time frames, will be discussed briefly in this section. The full code is viewable in Appendix A.

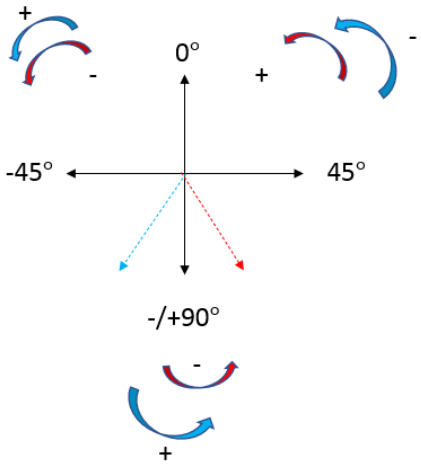
Matlab's libraries were used to grab the necessary variables from the raw data (which was collected via Matlab online app in smartphone):

- X_{GPS}, Y_{GPS} (Matlab libraries: lla2enu and xyzENU)
- φ (Matlab libraries: ecompass, eulerd, units: degrees, but final units need to be in radians, since Matlab prefers working with radians as output or input of a system.)
- a_x *and* a_y (interpolated raw acceleration data, units: m/s^2)

GPS's raw data can be converted from lla to body frame xyz -ENU, and the yaw angle is derived from each sampled data's quaternion via the accelerometer and magnetometer data. Interpolation between the GPS, accelerometer and magnetometer data is necessary, despite log streaming via Matlab Connector and Matlab sensor phone app at the same rate (10 Hz or 0.1 sec). There is a total mismatch of the overall data points collected by each sensor. GPS has the lowest number of data points due to its inherent low resolution (frequency), but the accelerometer ends up having the highest number of data points. Thus, interpolating GPS, magnetometer and accelerometer data with respect to the accelerometer's length of data is necessary before using Matlab's library to find the quaternion for each sampled data.

After, finding the rotation information for each frame, in quaternions, then can find the respective euler angles (yaw, pitch, roll). There were other methods that were explored to capture the yaw angle with respect to magnetic north, such as:

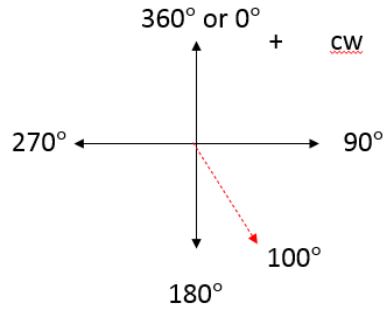
- multiplying the azimuth quaternion to each sampled data's full quaternion,
    - Cons:
        - Imaginary outputs in the final angle values.
        - May not take into account additional directional cosine terms (found within the coefficients for each of the i-j-k components) from within the full quaternion output (for each data sample).
        - The psi angle of the real term is the overall angle of the vector within the hypersphere after the rotation is complete. It's diagram is depicted as follows (where the vector can rotate continuously between the negative and positive quadrants, and the sign will change after each full rotation.):



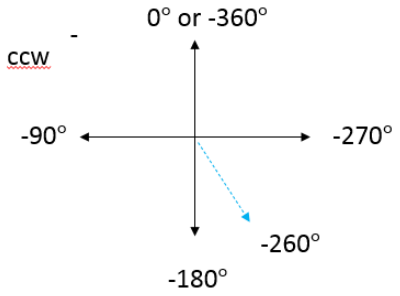
Red is 1st rotation. Blue is 2nd rotation. No matter which side the rotation starts, the sign will consecutively change after 1 full rotation. Thus, for ex: +84° can be either side, as denoted by blue/red vectors – if the rotation starts from the left side. However, where the rotation initially starts determines which sign it starts with. If rotation starts from the left side, it's negative, and + if starts from the right side. This example has the rotation starting from the left side.

-
- Due to the nature of how angles are read from the real term of the quaternion, when mathematically deriving the psi angle by multiplying the azimuth quaternion to the full quaternion, the result is two yaw angles. This is because depending on whether the angle was originally from a clockwise (cw) or counter clockwise (ccw) direction. Thus, let's say if the psi angle of the real part of the

quaternion of a particular data sample was 84°, then the result will yield either of the following angular values:



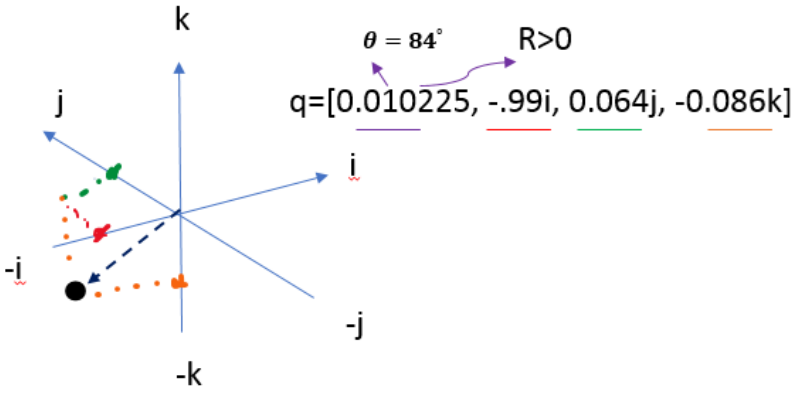
- or



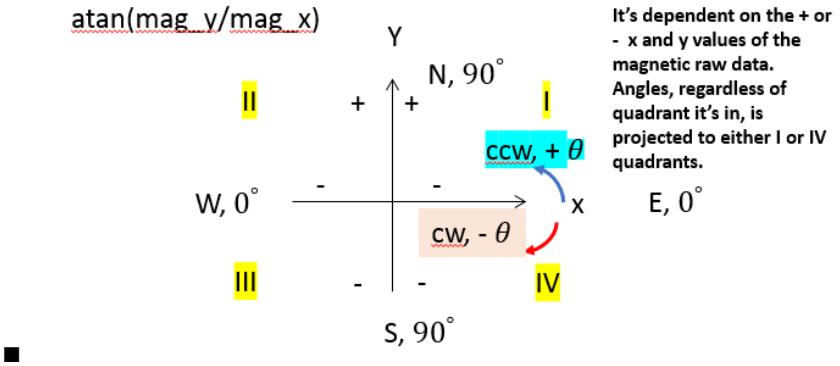
But can be remedied by creating a limits in a conditional if statement to only consider values between 0° $and \pm 180^{\circ}$.

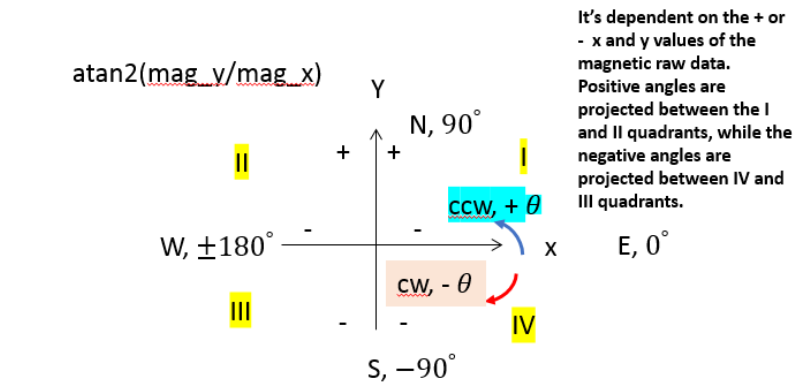
- The coefficients of i-j-k, basically the imaginary parts that make up the quaternion, are values that go between -1 and 1. The vector only travels half a circle for each component. For example, if the quaternion of a sample data looked like the following: [0.010225, -0.98898i, 0.064297j, -0.085567k] (for more mathematical information and derivations on quaternions please refer to sections 2.3.1 and 2.3.2)
  - The first term: is the real part of the quaternion and is also called the 'R' value, or the real value of the quaternion vector within the hypersphere. If R>0 it is inside the hypersphere, but if R<0 then it is outside of the hypersphere. It is also mathematically defined as cos(angle/2). This 'angle' can either be yaw, pitch or roll, if looking at each of the euler based quaternions separately. However, the final quaternion will have the final angle within the hypersphere as indicated in the earlier bullets. (Diagram below)
  - The second term: is the location of the i component, and is usually between the values of -1 and 1.
  - The third term: is the location of the j component, and is usually between the values of -1 and 1.
  - The fourth term: is the location of the k component, and is usually between the values of -1 and 1.

- Visual diagrams depicting this example can be viewed below:



- or the azimuth angle given by the orientation sensor in the phone,
    - Cons: the azimuth angle is with respect to where the y-axis is pointing when the operator first starts to stream log the data. It does not care where the north or south poles are, it will treat either North, South, East and West as its starting point (0°).
- The 'azimuth' Matlab library:
    - Cons: It depends on the operator to state where point A is at with respect to a point B (similar to how a mapmaker traditionally determines one's geographic location by drawing the center of the compass at point A then drawing a vector, from its center, to a point B). Also, the angles go full 360°.
- or the atan and atan2:
    - Cons: It gives positive or negative angles with respect to quadrant location and not with respect to 0° N and $\pm$ 180° S. Although atan2 can help determine directionality, it still determines it via the +/- values of the raw y and x magnetometer's data. Thus, even that method is quadrant based. Furthermore atan only gives values within either the I or IV quadrants; thus, if a magnetometer's raw y and x data are found to be in the III quadrant, the final angle value using atan yields it in the I quadrant (it projects it to it). Likewise, if the magnetometer's raw y and x data values are found to be in the II quadrant, its final angular value, using atan, is found to be in the IV quadrant.
    - Diagram of atan and atan2 :



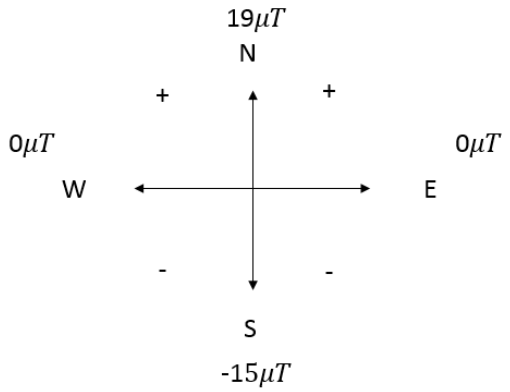


- or even using Matlab's PhoneProcessorData function:
    - Cons: It depends on the 'Orientation' sensor to determine the euler angles, and need to conduct a q_fix*q_true for correction. However, it may not incorporate the correct declination angle with respect to one's geographic location.

However, each proved to be insufficient in grabbing the correct yaw angle. Their respective explanations of such can be found in Appendix A's Matlab code (titled 'ME295B_1' under the sections in whose titles have an additional '-Test' at its end).

It is critical to note that the yaw angle with respect to the true north is called azimuth and to derive it, one must subtract (if positive/East) or add (if negative/West) the corresponding declination angle of their geographic location. Thus, the yaw angle that is the final input for the Kalman filter is the azimuth angle and is in radians.

The diagram for how the magnetometer sensor behaves is depicted below:



The non-zero values given are arbitrary and are based on the sensor's aptitude in detecting the magnetic strength as the Y or X axis is facing the N or S poles. These values are based on when Y is the forward axis (ENU frame) and is pointing along either of the poles. Note that when Y is facing E or W, the X axis is facing S or N respectively.

-

Matlab code for each step is done in the order as discussed within its respective sections (viewable in Appendix A).

Chapter 5 - MATLAB and ROS1

5.1 MATLAB Results

5.1.1 Background

Two approaches were explored with respect to the Kalman filter's linear acceleration inputs. The approaches are with respect to how linear acceleration is derived. The first is with respect to assuming a constant gravitational acceleration of $9.81 m/s^2$ for each planar (xz and yz) tilt angle; while the second approach uses the rotation matrix of each data sample to extract the actual sensor's gravitational acceleration for each tilt angle. However, since both approaches yielded the same performance response curves, only Approach 1 will be displayed in this chapter. Approach 2 can be viewed in Appendix A's Matlab code titled: 'ME295B_F_A2'.

Section 5.1.2 will display the error and performance response curves at different critical hyperparameters using Matlab as the selected IDE. Section 5.1.3 will present the performance response curves with respect to ROS's selected packages. Thus, a comparison between a linearized Kalman filter, low pass filter and a non-linearized Extended Kalman Filter is done throughout this chapter. Appendix A contains all the Matlab code used for this project. Appendix B will contain all the code for the ROS1 setup used for this project.

Q and R are variance matrices for the accelerometer and GPS, respectively. Q was determined by taking the average of the variances of the accelerometer along x and the accelerometer along y. While R was determined via the variance of the GPS measurements and the 2m resolution of the Y_GPS vs X_GPS plots. Since, tolerances changed per GPS receiver, this study aimed to tune the Kalman filter via a change in R.

5.1.2 MATLAB Results

5.1.2.1 Kalman Filter

All plots below are based off of Approach 1, where Q = 0.15 m/s^2, R = 4 m and L = [0.0585, 0.0603], unless otherwise stated.

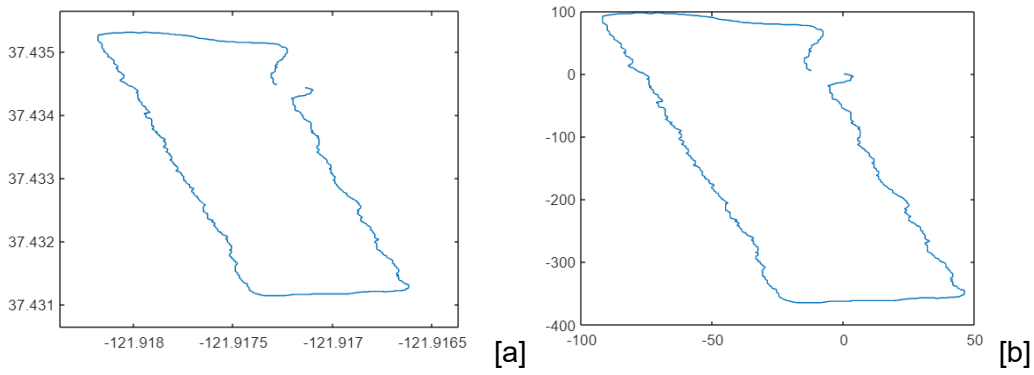


Figure 35: *Converting lla geodetic [a] format to local xyz ENU coordinates [b]. Note that the origin is at the starting point and that the + y axis attributes to the path that's traveling North, while the - y axis attributes to traveling South. Latitude corresponds to the Y axis (Northerly/Southerly) while Longitude corresponds to the X axis (Easterly/Westerly).*

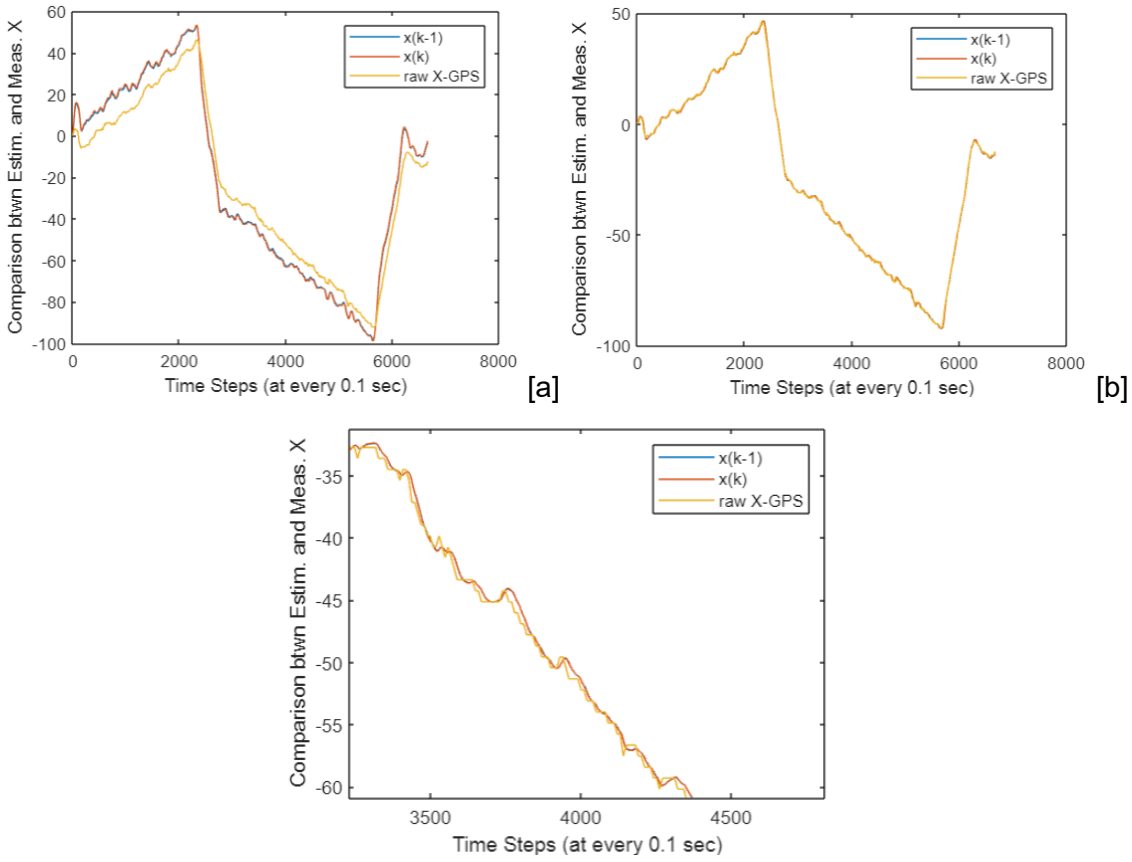
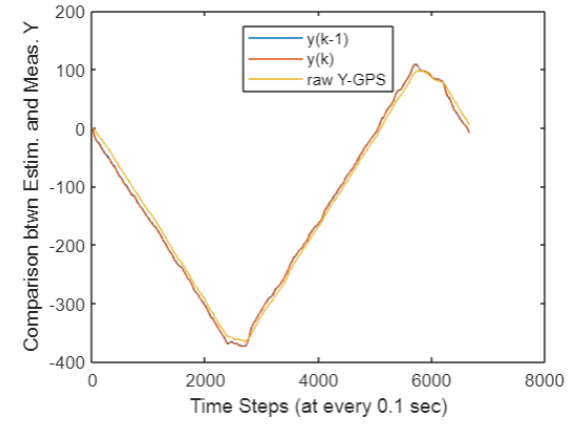
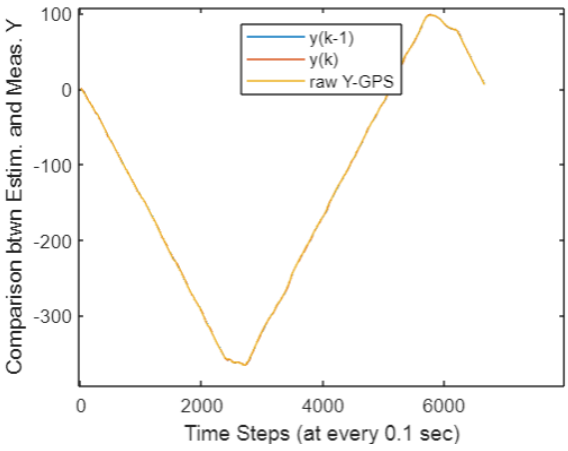


Figure 36: *Response curves between the estimated and measured states along the X axis. Comparison between the accelerometer data as is [a] (gravitational + linear components) vs. using only the linear acceleration [b]. Additional close up for the latter figure [b], as well.*



[a]



[b]

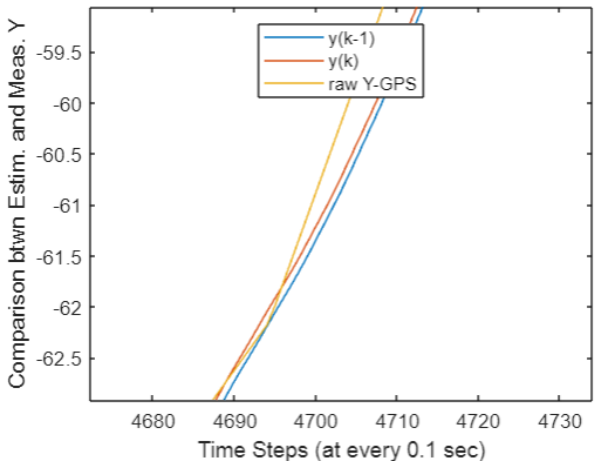


Figure 37: *Response curves between the estimated and measured states along the Y axis. Comparison between the accelerometer data as is [a] vs. using only the linear acceleration [b]. Additional close up for the latter figure [b], as well.*

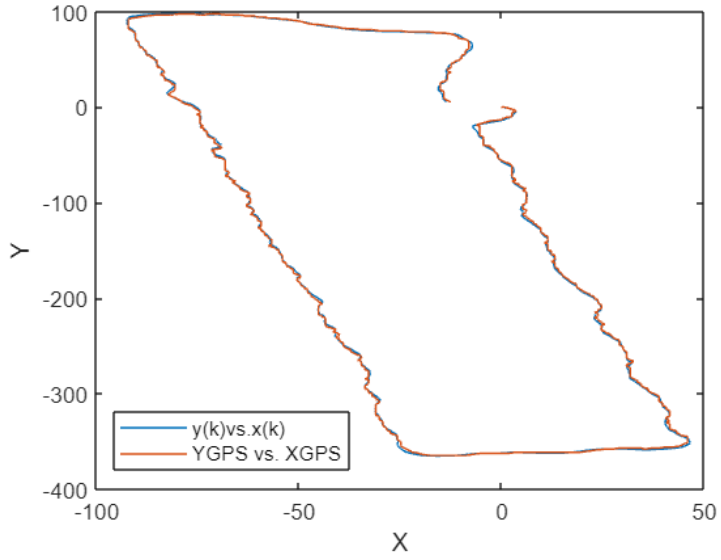


Figure 38: *Plot comparisons between estimated and measured states (for along both x and y axes).*

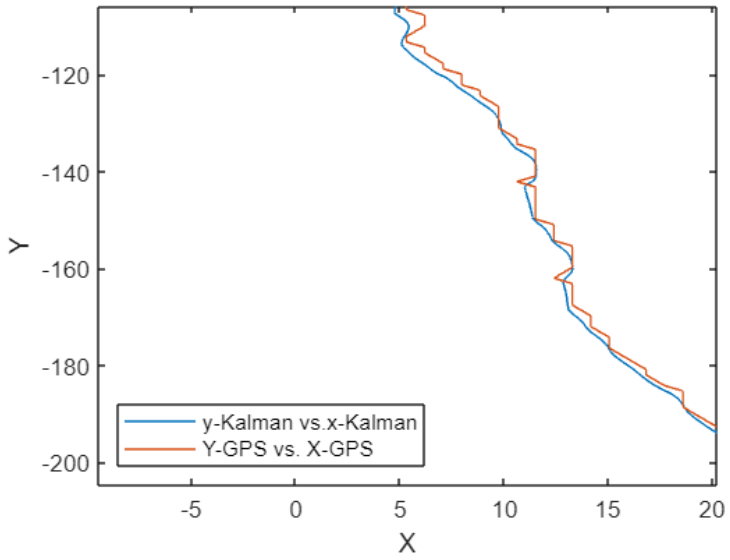
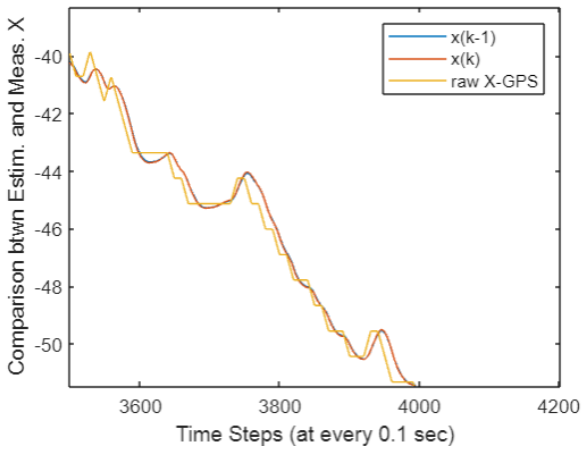


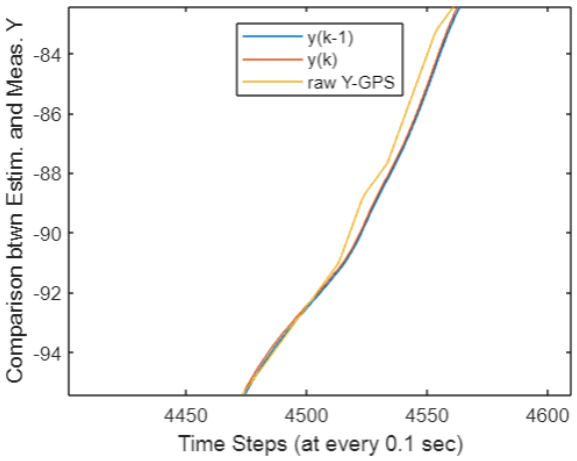
Figure 39: *Close up on Figure 38's plot. This is with R = 4 and Q = 0.15. The mobile device is traveling south. Note the difference in resolution between measured (red) and estimated states (blue).*

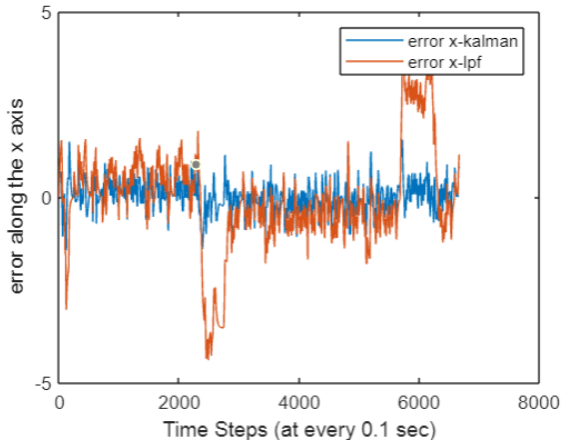
Table 7. R Value Comparisons in Kalman Filter (Alpha = 0.95).

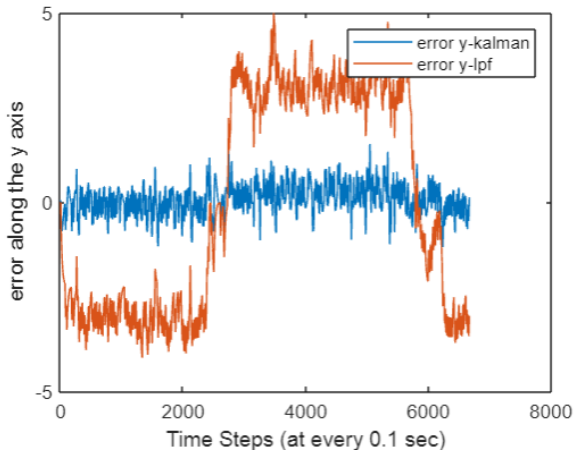
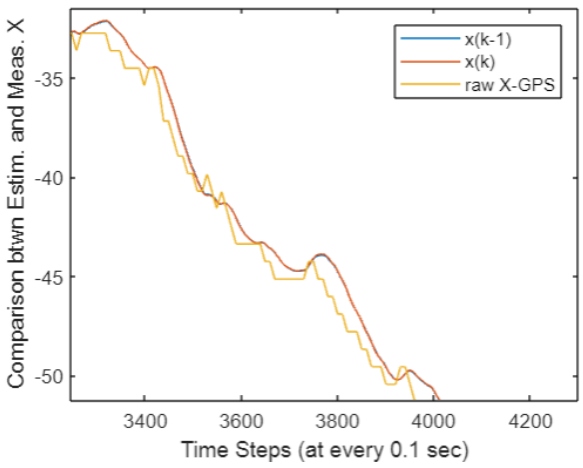
R values in Kalman Filter	**Final Response Curves & Error Plots**

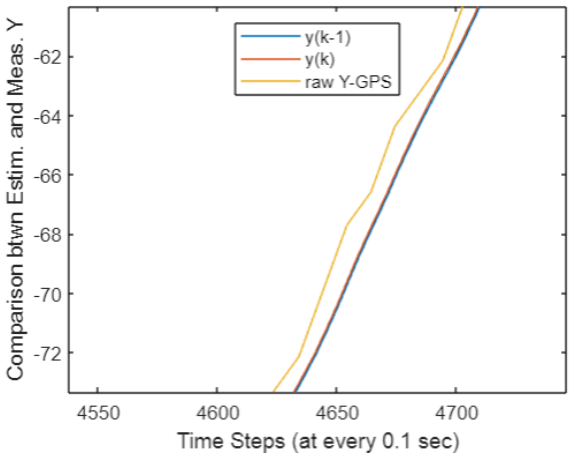
R = 1 (+/- stnd_dev = 1 m)

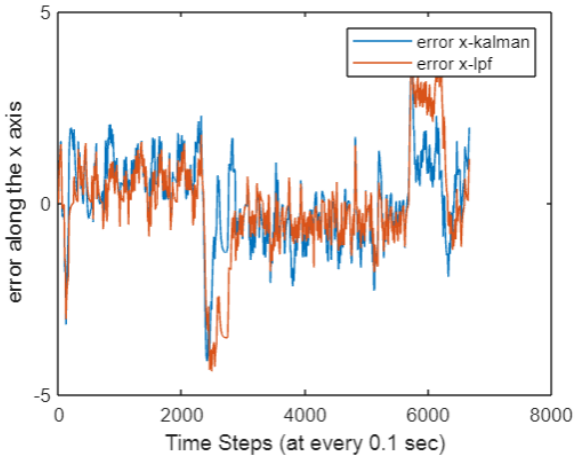


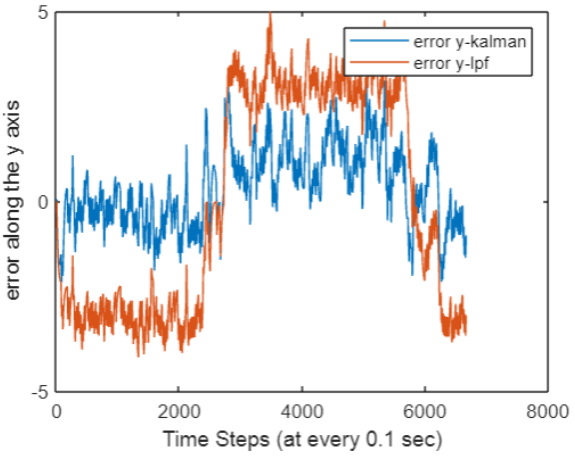




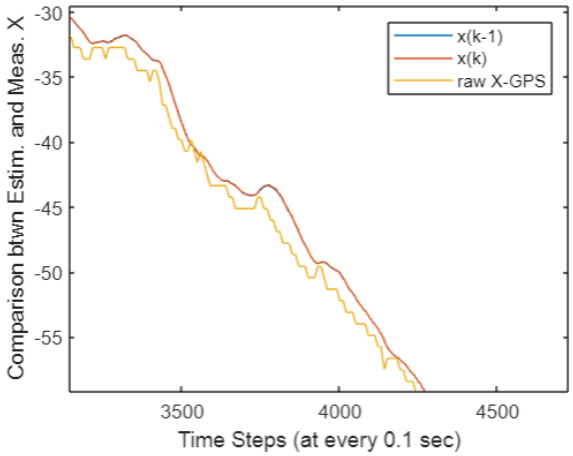
	
R = 4 (+/- stnd_dev = 2 m)	Please check earlier figures (for X and Y comparisons) and Figure 42 (for the error plots. However, alpha = 0.85 for those figures.)
R = 25 (+/- stnd_dev = 5 m)	

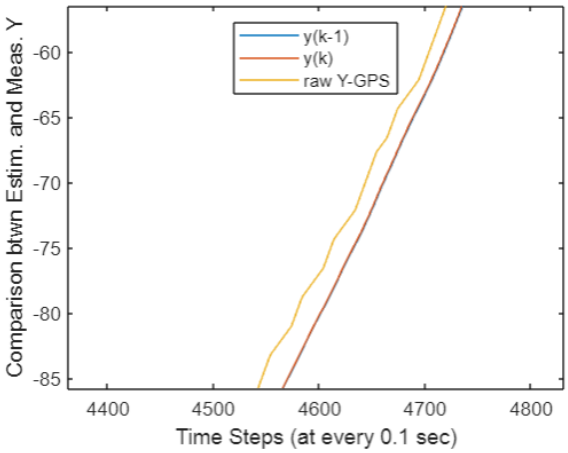

y(k-1)
y(k)
raw Y-GPS
Comparison btwn Estim. and Meas. Y
-62
-64
-66
-68
-70
-72
4550
4600
4650
4700
Time Steps (at every 0.1 sec)

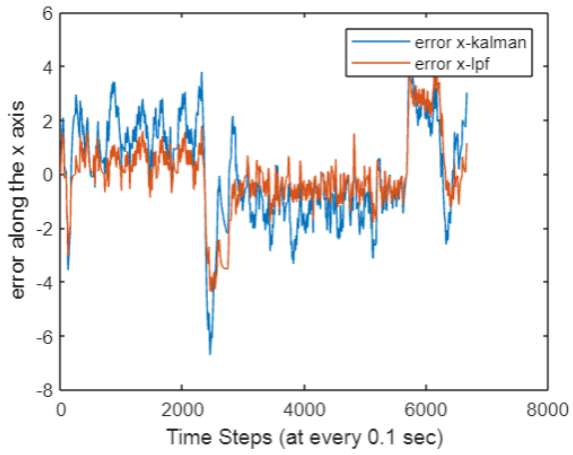

error x-kalman
error x-lpf
error along the x axis
5
0
-5
0
2000
4000
6000
8000
Time Steps (at every 0.1 sec)

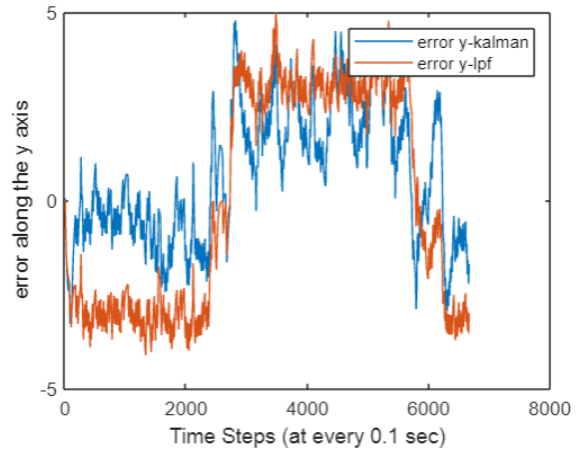

error y-kalman
error y-lpf
error along the y axis
5
0
-5
0
2000
4000
6000
8000
Time Steps (at every 0.1 sec)

R = 100 (+/- stnd_dev = 10 m)









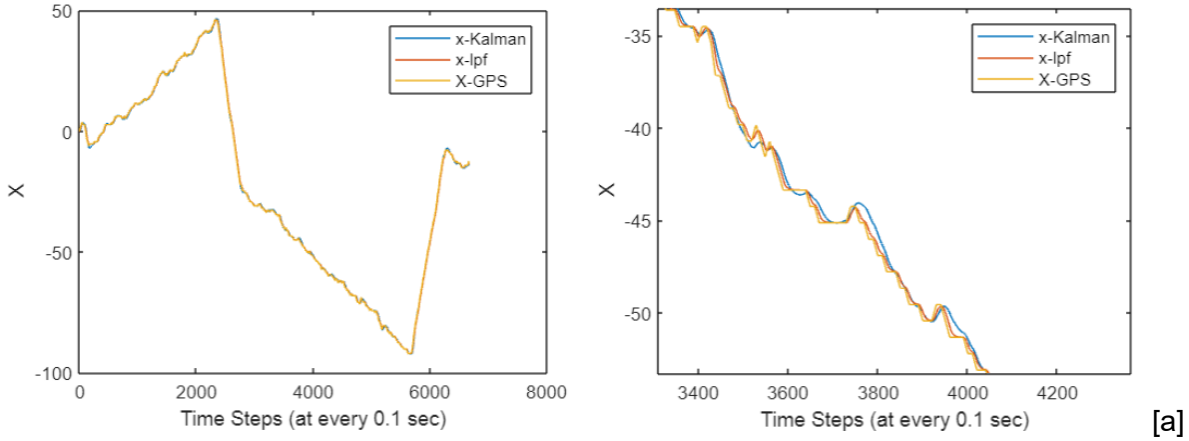
5.1.2.2 Low Pass-Filter

Low pass filter constructed as a comparison to the Kalman filter's outputs. Adjustment of alpha value is explored (while R= 4, Q=0.15). Low pass filter's formula can be viewed in Appendix A's MATLAB code under file titled 'ME295B_Validation.ii'.

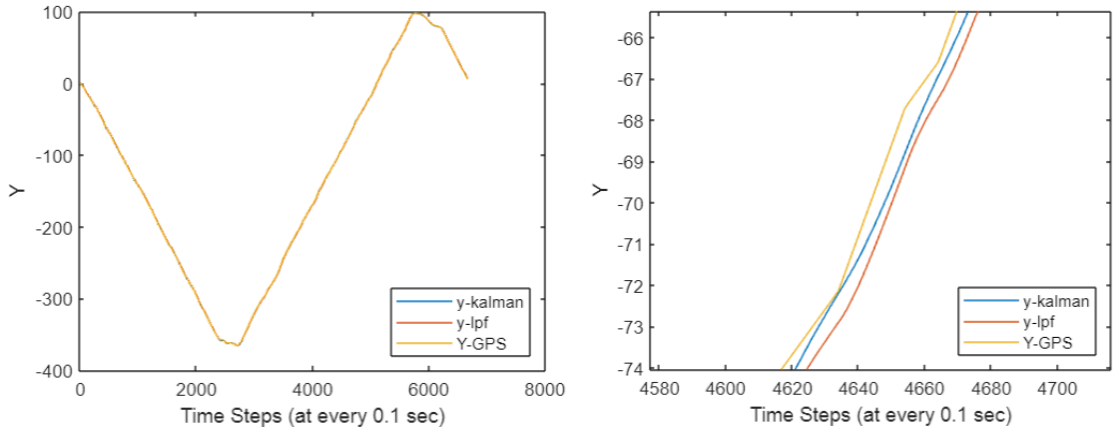
Low pass formula for X and Y are as follows:

$$X_{filt}(k+1) = \alpha(X_{hat}(k)) + (1-\alpha)X_{GPS} \quad (20.1)$$

$$Y_{filt}(k+1) = \alpha(Y_{hat}(k)) + (1-\alpha)Y_{GPS} \quad (20.2)$$

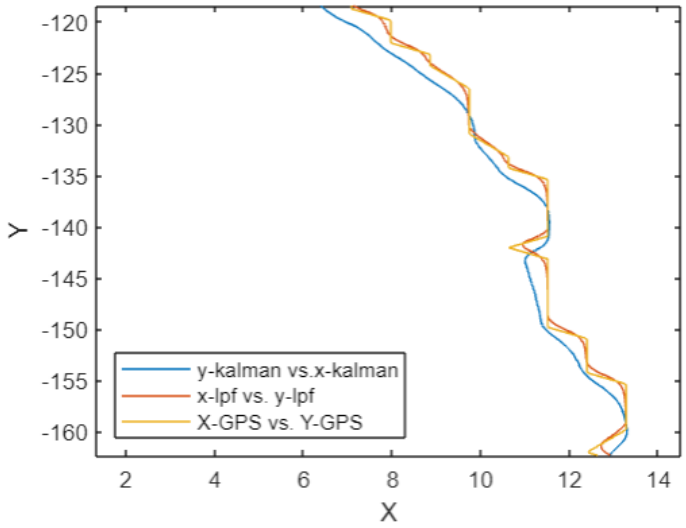


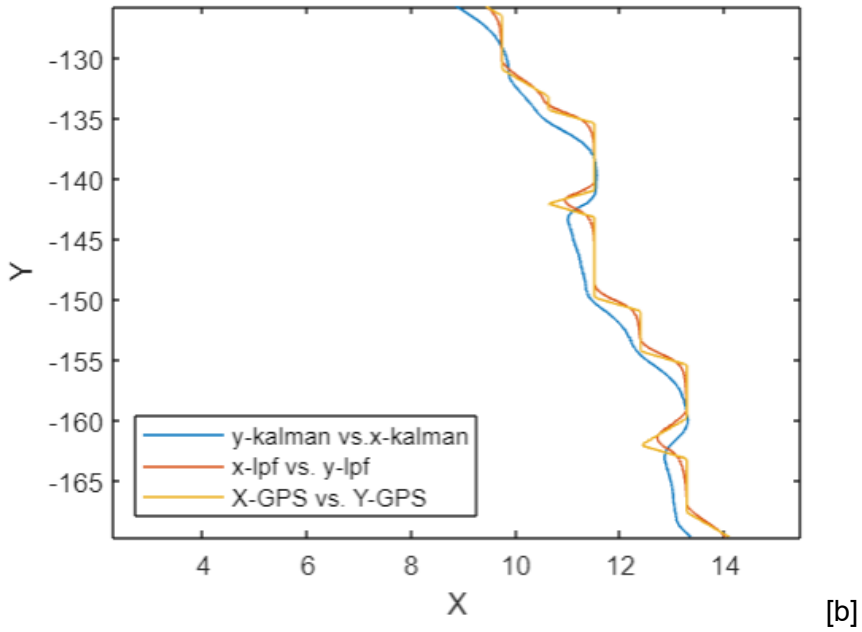
[a]



[b]

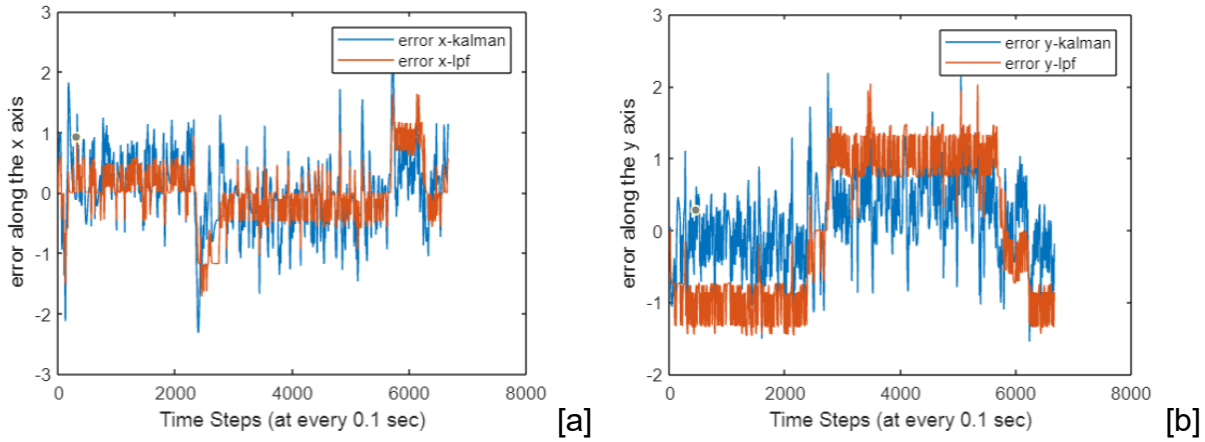
Figure 40: *Plot comparisons between the low pass filter in the x and y directions with that of the estimated states produced by the Kalman filter and the measured state (raw GPS data in either axes). Alpha = 0.85. A close up of each plot's respective axes is also provided.*





[b]

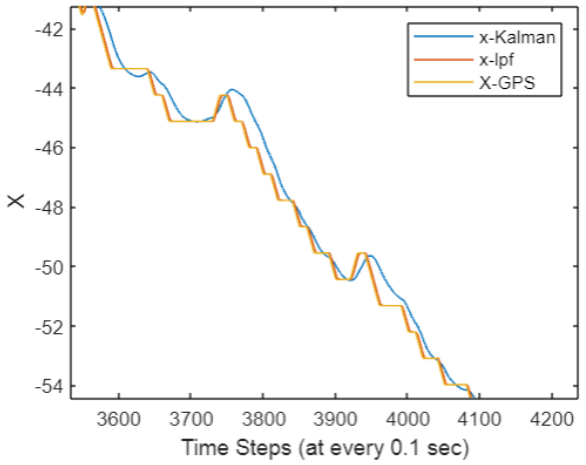
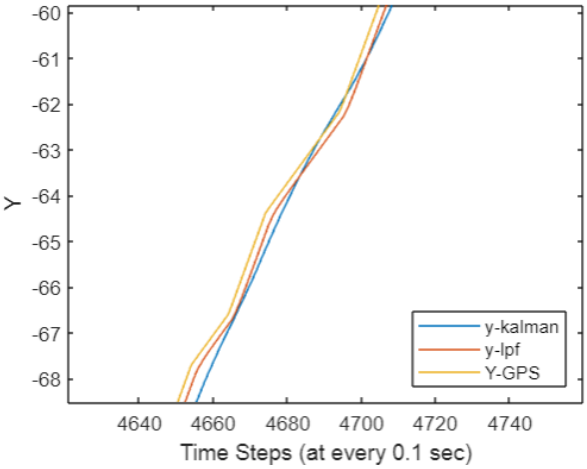
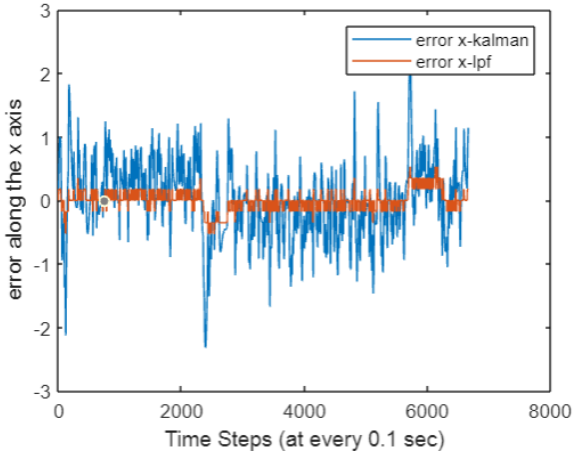
Figure 41: *Plot comparison between the final estimated, measured and low- filter states. A close up is provided. Alpha = 0.85*

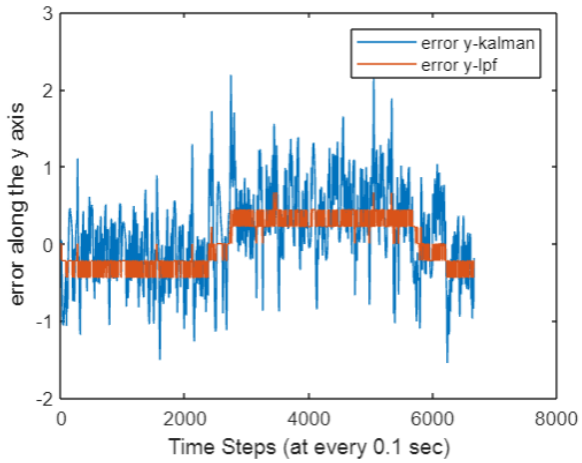
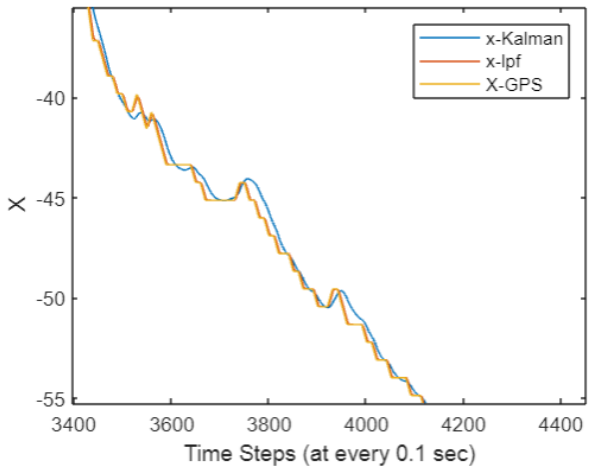
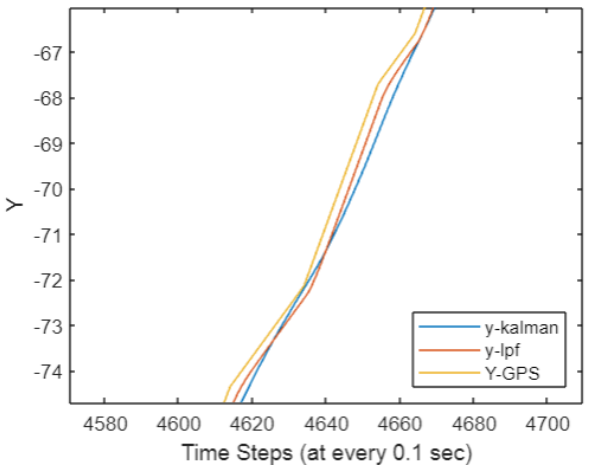


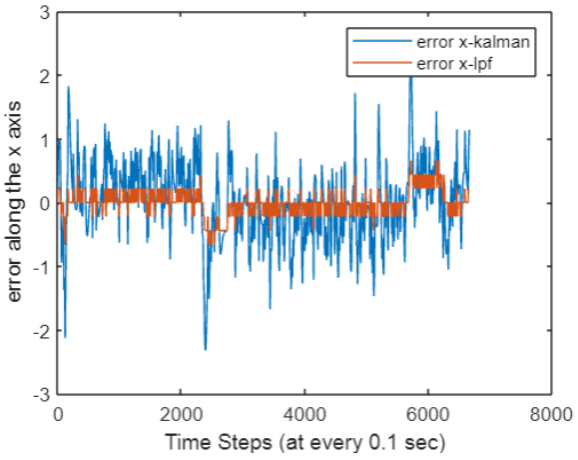
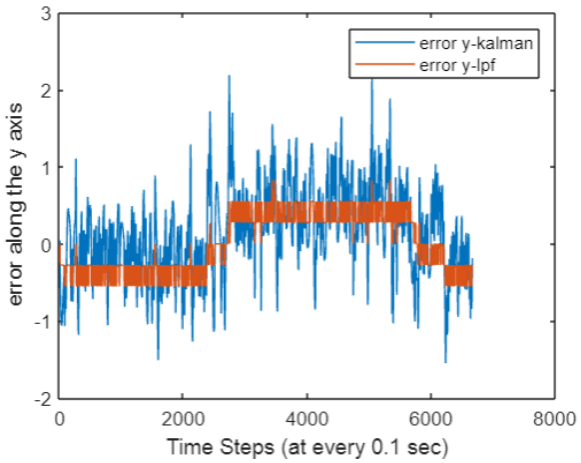
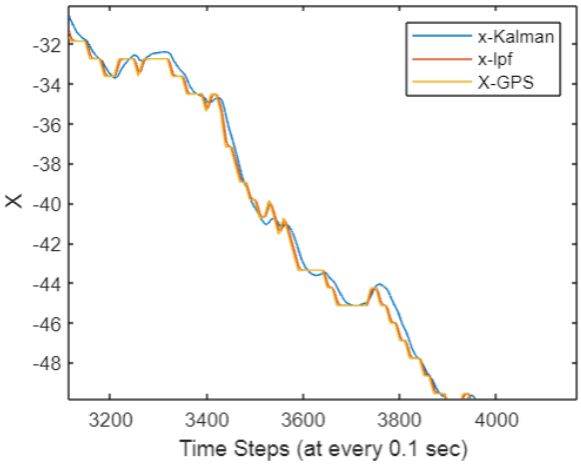
[a] [b]

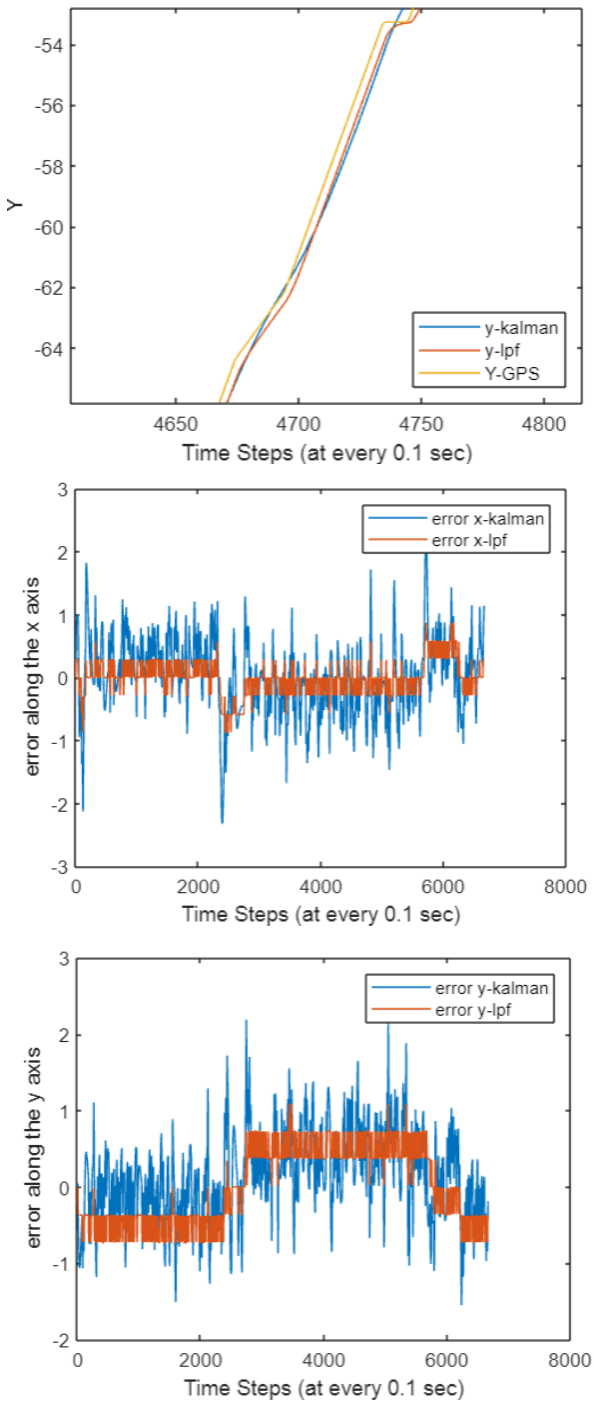
Figure 42: *Error plot comparisons between the Kalman filter and low pass filter's estimated states. Note that the low pass filter is close to the measured state (reduced error than that of the Kalman filter's error with respect to the measured states (X-GPS, Y-GPS)). Alpha = 0.85 and R=4.*

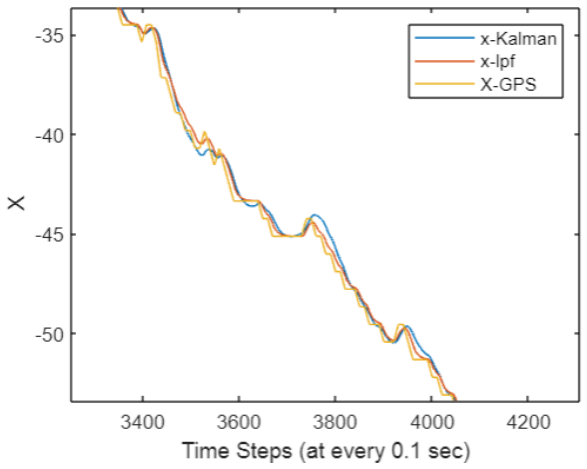
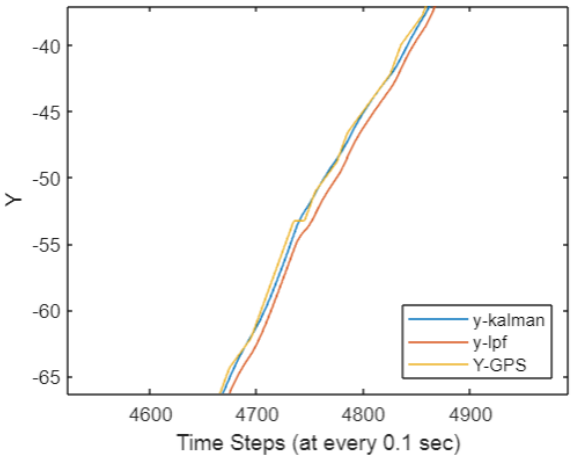
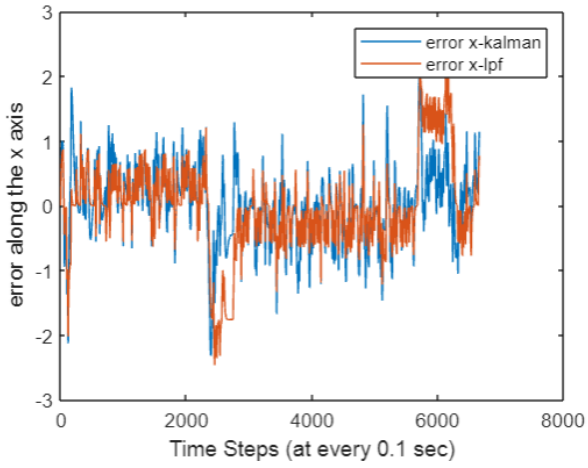
Table 8. Alpha Value Comparisons in Low Pass Filter at R=4.

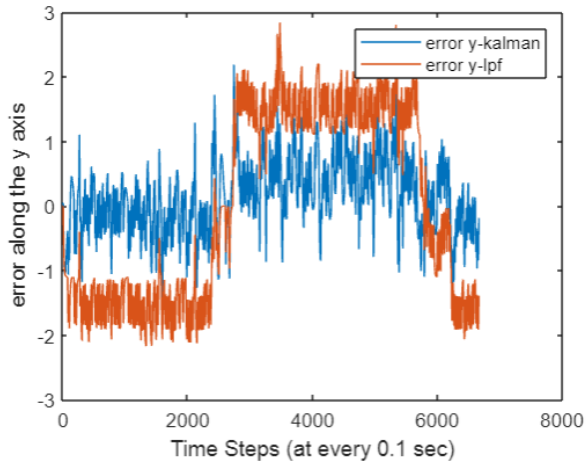
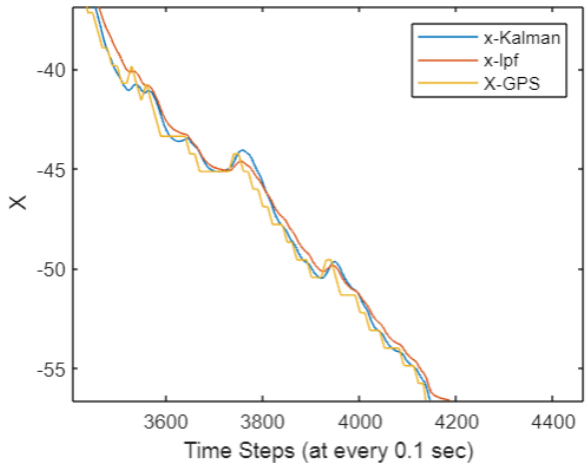
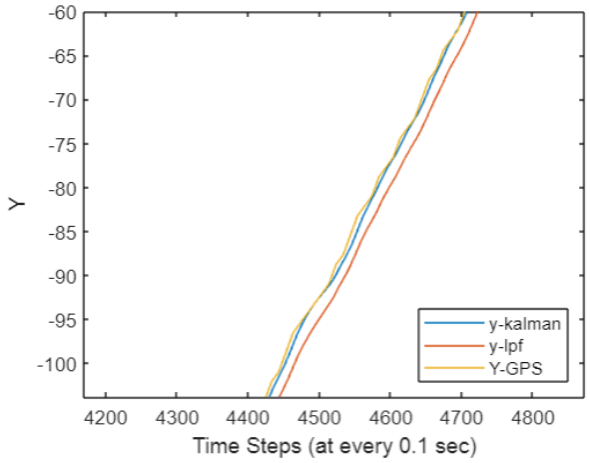
Alpha values wrt Low Pass Filter	Final Response Curves & Error Plots
Alpha = 0.5	  

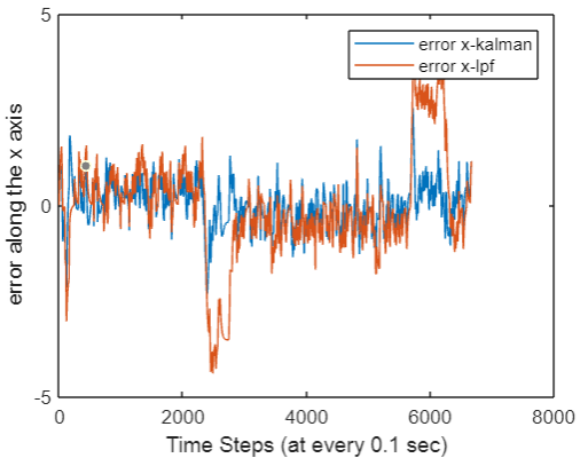
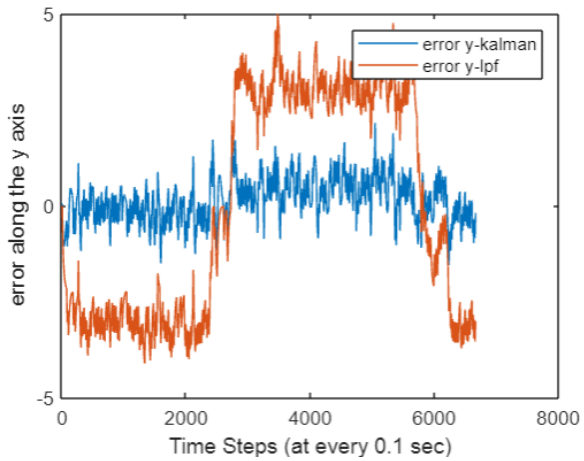
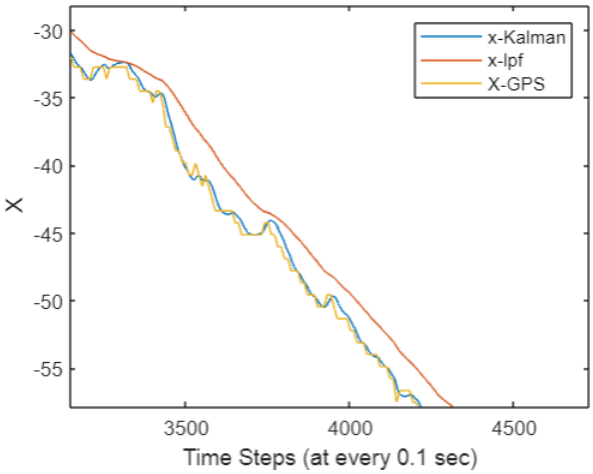
	 error y-kalman error y-lpf error along the y axis Time Steps (at every 0.1 sec)
Alpha = 0.6	 x-Kalman x-lpf X-GPS X Time Steps (at every 0.1 sec)  y-kalman y-lpf Y-GPS Y Time Steps (at every 0.1 sec)

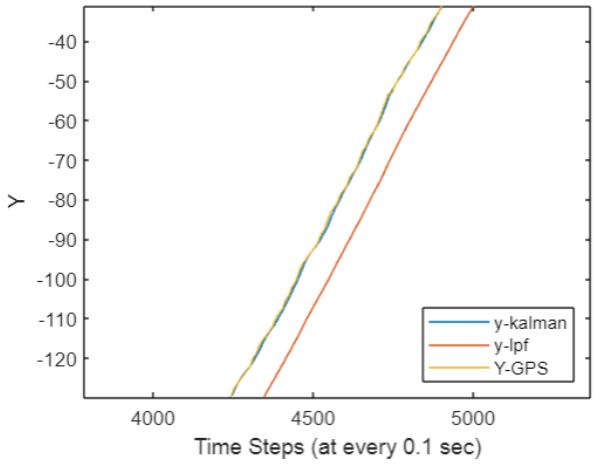
	 error x-kalman error x-lpf error along the x axis Time Steps (at every 0.1 sec)  error y-kalman error y-lpf error along the y axis Time Steps (at every 0.1 sec)
Alpha = 0.7	 x-Kalman x-lpf X-GPS X Time Steps (at every 0.1 sec)

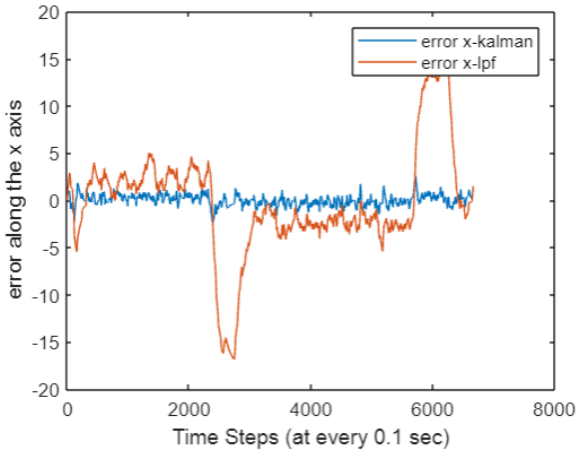
	
Alpha = 0.85	Please see previous figures

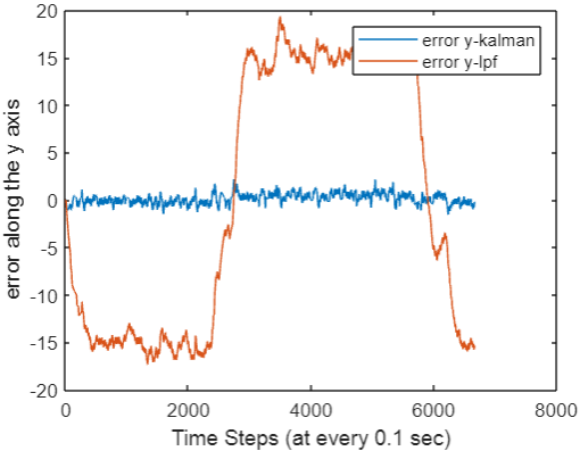
Alpha = 0.9	  

	 error y-kalman error y-lpf error along the y axis Time Steps (at every 0.1 sec)
Alpha = 0.95	 x-Kalman x-lpf X-GPS X Time Steps (at every 0.1 sec)  y-kalman y-lpf Y-GPS Y Time Steps (at every 0.1 sec)

	 error x-kalman error x-lpf error along the x axis Time Steps (at every 0.1 sec)  error y-kalman error y-lpf error along the y axis Time Steps (at every 0.1 sec)
Alpha = 0.99	 x-Kalman x-lpf X-GPS X Time Steps (at every 0.1 sec)





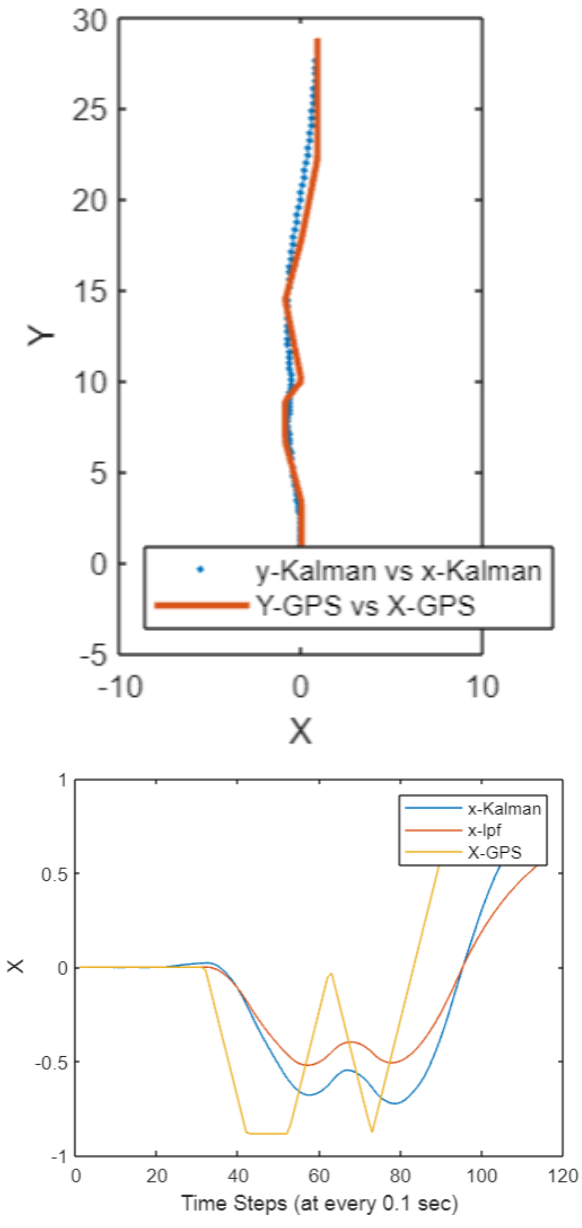


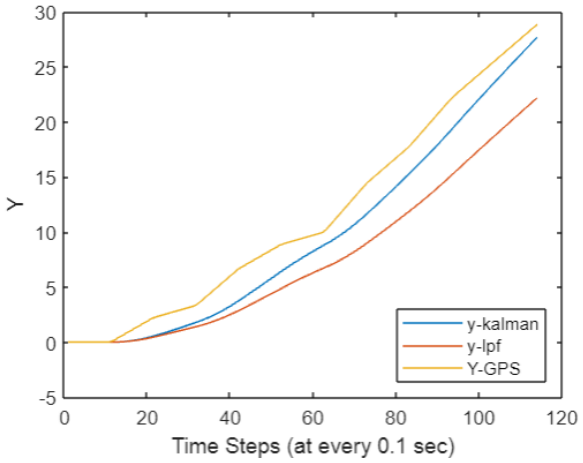
Although the low pass filter's results are close to the Kalman filter's estimated states, at an alpha = 0.95, the latter is more robust. The Kalman filter's observer gain values are the corrected weight values with respect to the covariances of the measured and estimated states. The error between the measured and estimated states are closer to zero for the Kalman filter in the forward y direction than that of the low pass filter.

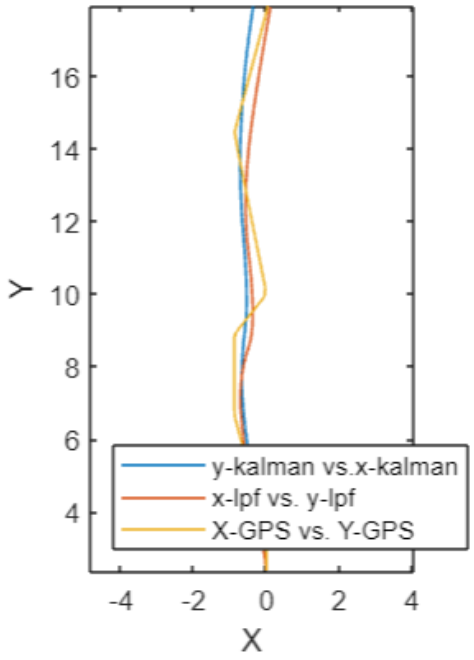
5.1.2.3 Other Validation Runs

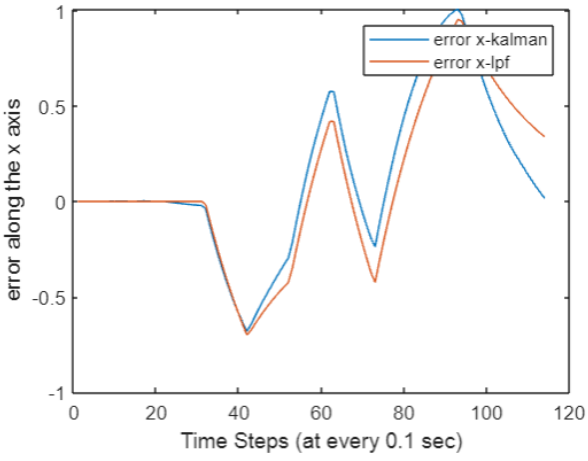
This section explores the ideal R=4 and alpha=0.95 selected from the Kalman and low pass filters, respectively, in different contexts of movement.

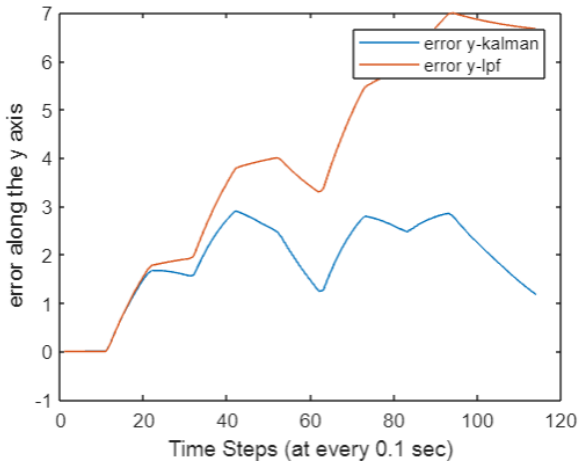
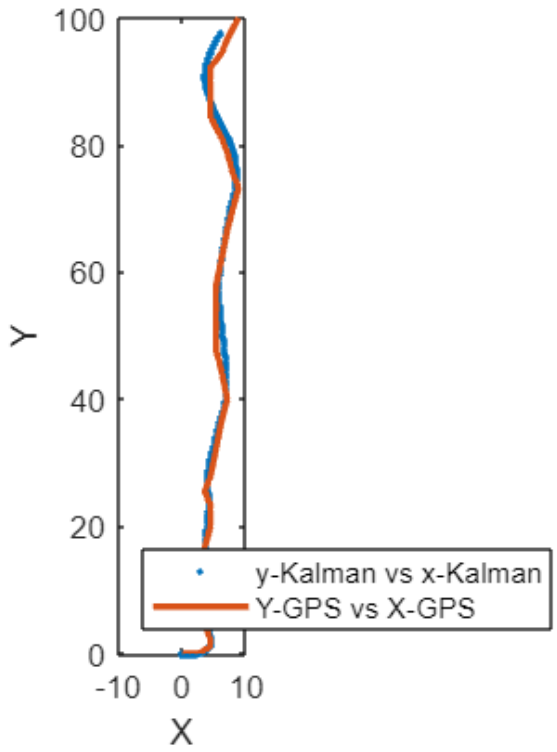
Table 9. Validation Runs wrt R=4 and alpha=0.95.

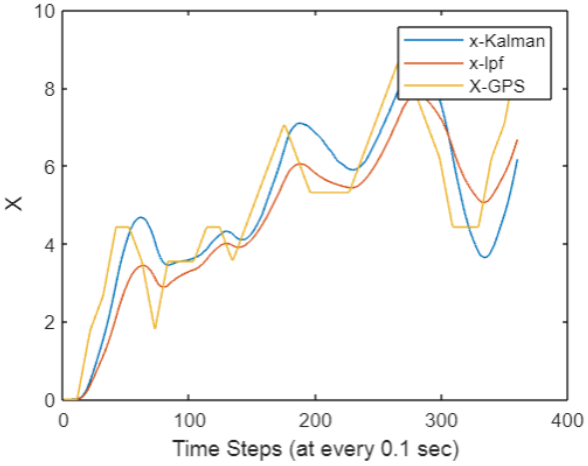
Context of the Movement	Final Response Curves & Error Plots
Accelerating in the magnetic north direction. (`'accelfn.mat'`)	

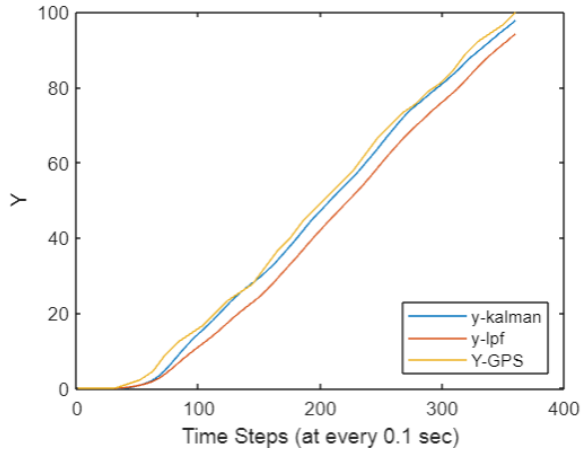

y-kalman
y-lpf
Y-GPS
Y
Time Steps (at every 0.1 sec)

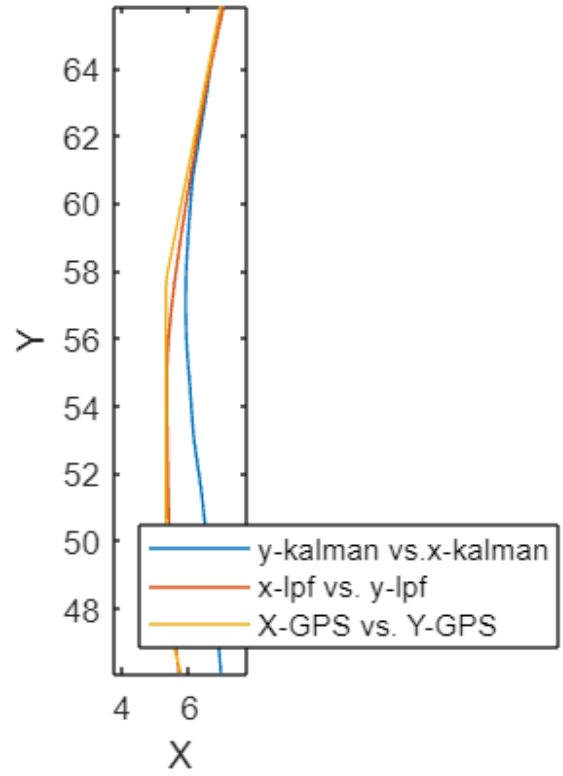

y-kalman vs.x-kalman
x-lpf vs. y-lpf
X-GPS vs. Y-GPS
Y
X

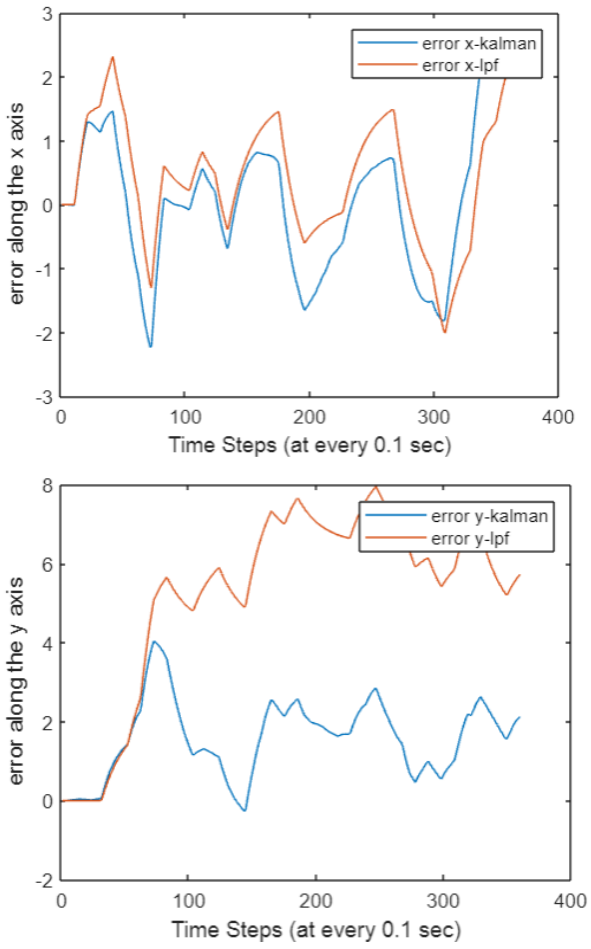
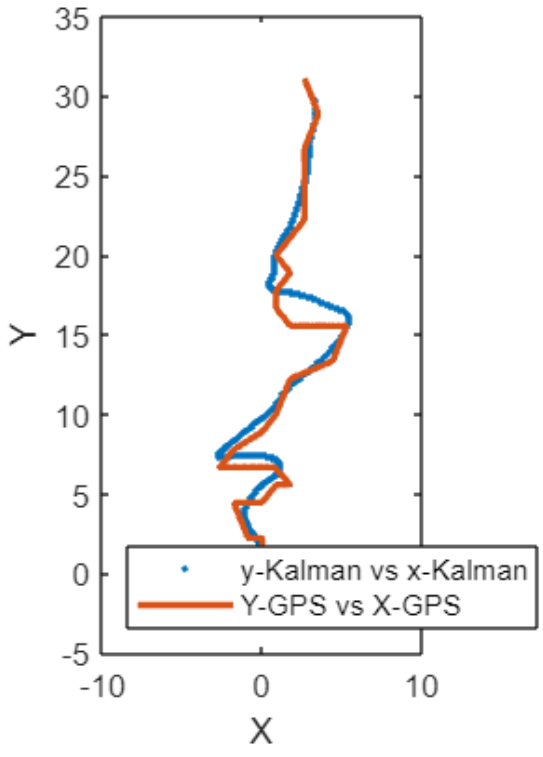

error x-kalman
error x-lpf
error along the x axis
Time Steps (at every 0.1 sec)

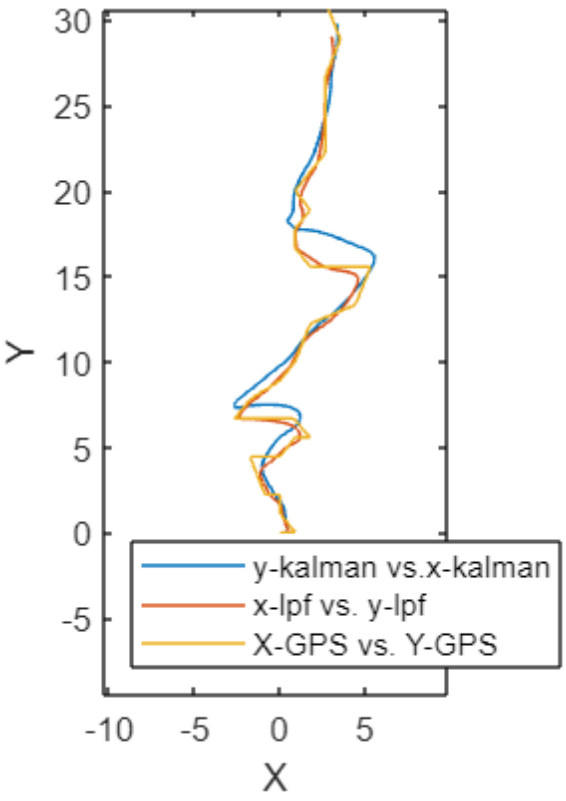
	
Accelerating and wiggling (traveling 1 m off course and then back - quickly 3 times) while traveling in the magnetic north direction. (`'accelwigglenewnorth1.mat'`)	

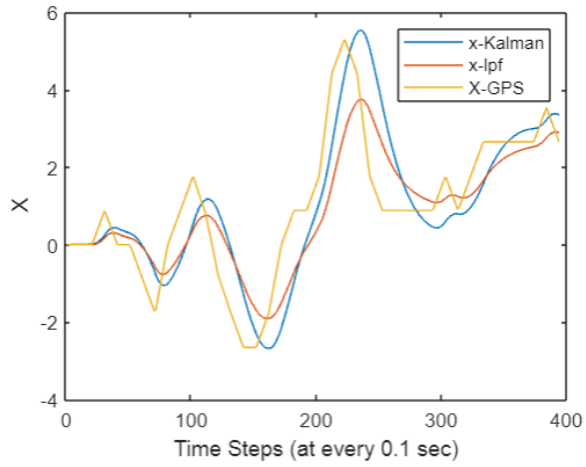

x-Kalman
x-lpf
X-GPS
X
Time Steps (at every 0.1 sec)

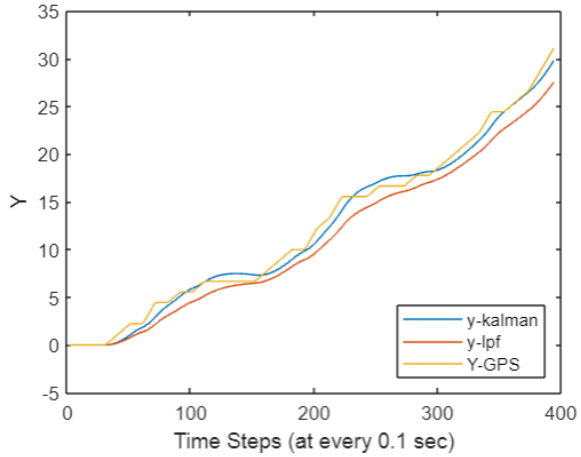

y-kalman
y-lpf
Y-GPS
Y
Time Steps (at every 0.1 sec)

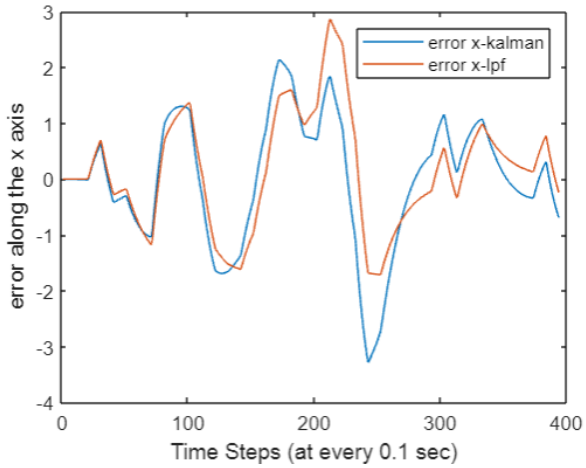
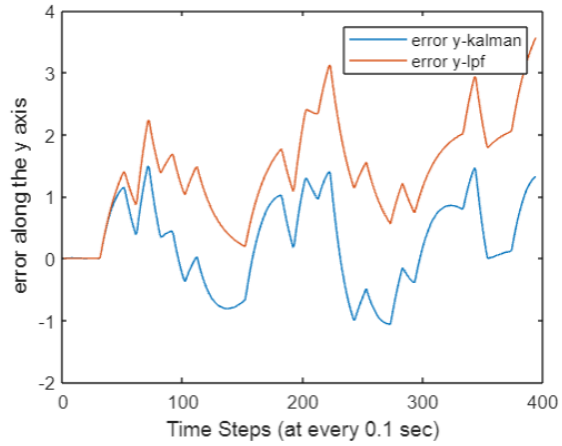
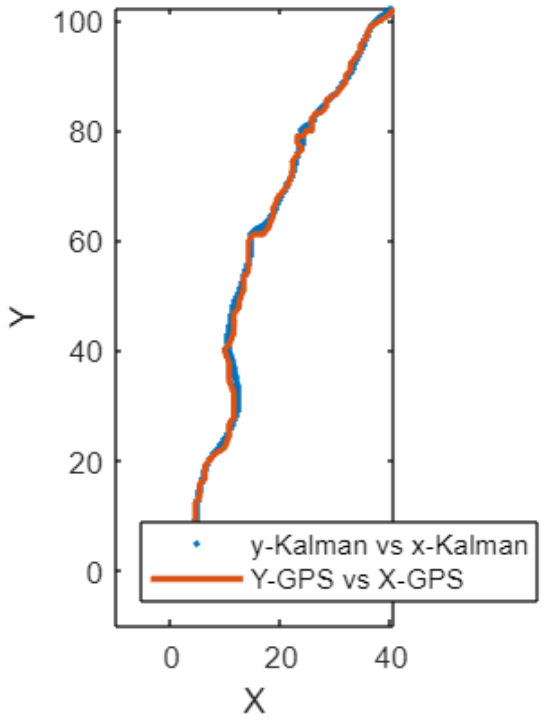

y-kalman vs.x-kalman
x-lpf vs. y-lpf
X-GPS vs. Y-GPS
Y
X

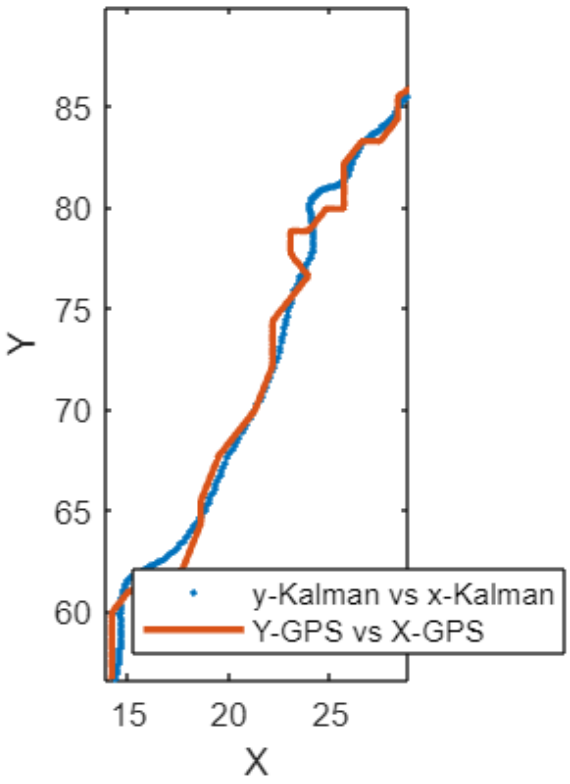
	
Slow wiggle while traveling due magnetic north. (`wiggle1.mat`)	

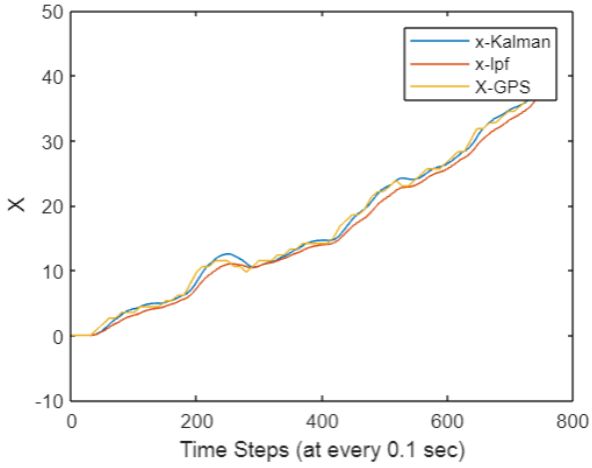

30
25
20
15
10
5
0
-5
Y
-10
-5
0
5
X
y-kalman vs.x-kalman
x-lpf vs. y-lpf
X-GPS vs. Y-GPS

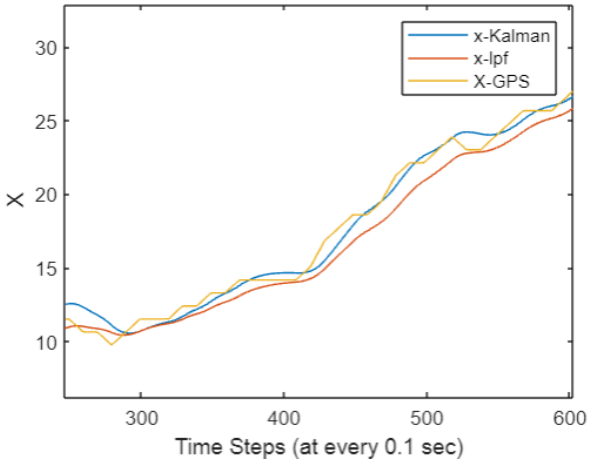

6
4
2
0
-2
-4
X
0
100
200
300
400
Time Steps (at every 0.1 sec)
x-Kalman
x-lpf
X-GPS

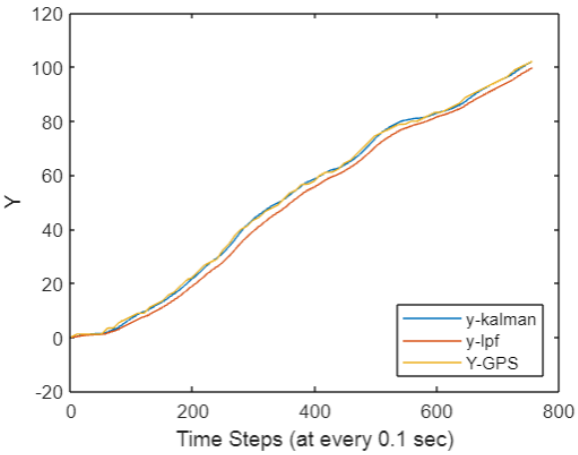

35
30
25
20
15
10
5
0
-5
Y
0
100
200
300
400
Time Steps (at every 0.1 sec)
y-kalman
y-lpf
Y-GPS

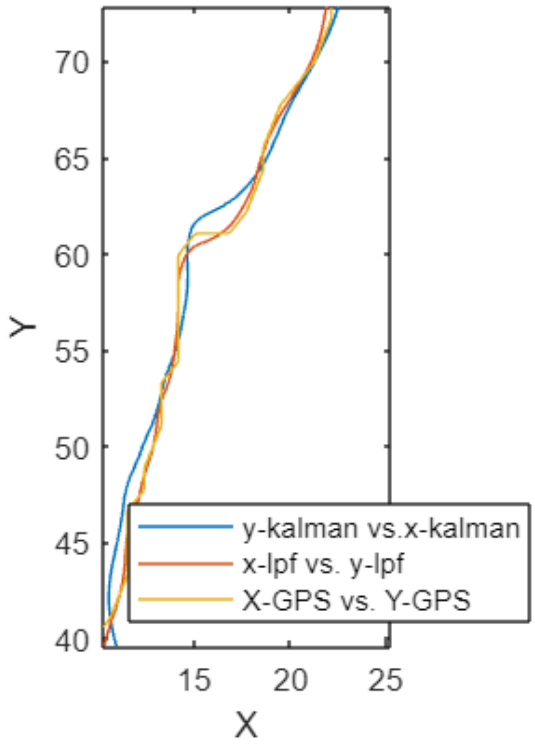
	 
Quick wiggling due magnetic north (`'wigglenewnorth3.mat'`)	

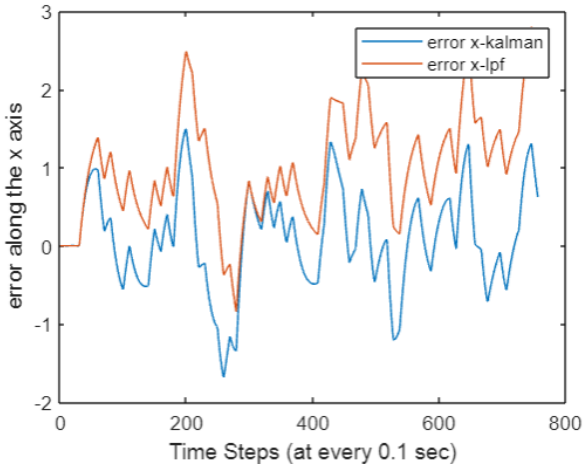

85
80
75
70
65
60
Y
15
20
25
X
y-Kalman vs x-Kalman
Y-GPS vs X-GPS

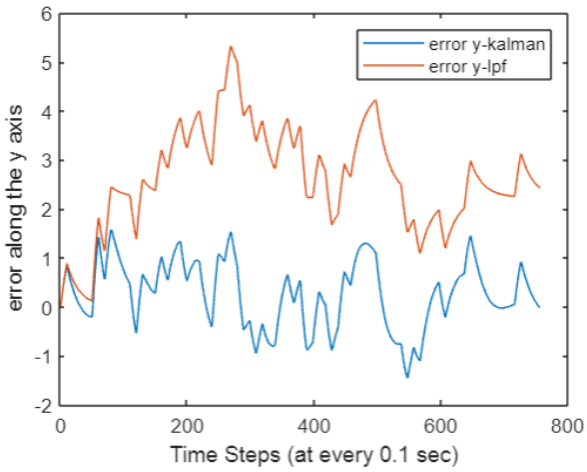

50
40
30
20
10
0
-10
X
0
200
400
600
800
Time Steps (at every 0.1 sec)
x-Kalman
x-lpf
X-GPS


30
25
20
15
10
X
300
400
500
600
Time Steps (at every 0.1 sec)
x-Kalman
x-lpf
X-GPS


120
100
80
60
40
20
0
-20
Y
0
200
400
600
800
Time Steps (at every 0.1 sec)
y-kalman
y-lpf
Y-GPS


70
65
60
55
50
45
40
Y
15
20
25
X
y-kalman vs.x-kalman
x-lpf vs. y-lpf
X-GPS vs. Y-GPS


3
2
1
0
-1
-2
error along the x axis
0
200
400
600
800
Time Steps (at every 0.1 sec)
error x-kalman
error x-lpf



5.1.3 ROS1 Results

Plots below are accomplished via ROS1, Melodic and the Matlab-ROS connector so as to use Matlab's workspace data. All corresponding code and final plots can be found in Appendix B.

EKF, with respect to ROS, can be found in the following packages (which are distinguished by dimension, number and type of sensors, and flexibility with respect to user interface.):

- Robot_localization: 3-D tracking and can stack multiple sensors. Allows an operator to use binaries in selecting which variables to consider as states, as well as using real values for the initial process and measurement noise covariance matrices.
- Robot_pose_ekf: 3-D tracking, limited to two odometry and one IMU data, and to ignore certain variables must set its respective covariance high (binary setup).
- Amcl: 2-D localization and considers odometry and laser.

Due to its flexibility with respect to its application, the robot_localization package was selected. The results can be seen in Appendix B.

5.1.2.3 ROS Hardware and Approach

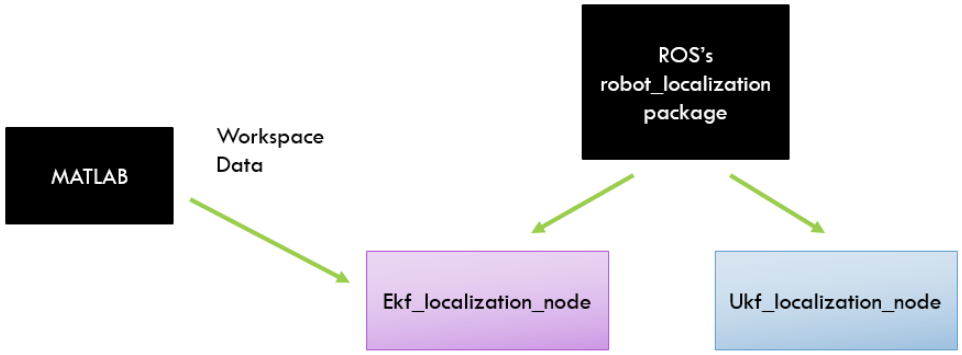


Figure 43: *Used Ubuntu 18.04, Matlab 2021b and ROS 1 Melodic, all in one PC (iMAC).*

Completed the following steps:

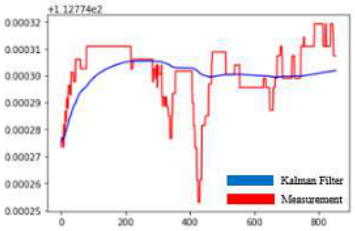
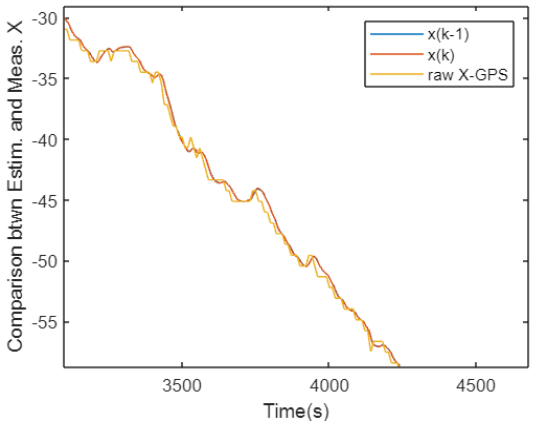
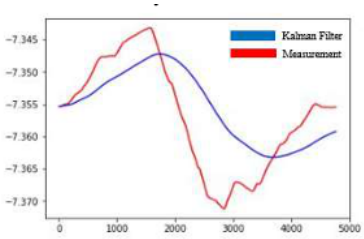
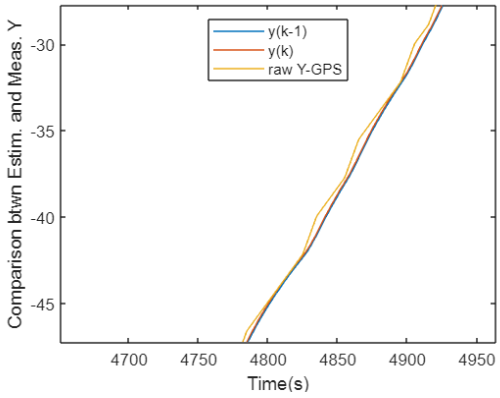
- Melodic/ROS1 full desktop set up
- Installed Matlab R2021b
- Installed the robot_localization package and rosran the ekf_localization_node
- ROS initiated setup in Matlab – thus creating a Matlab global master node
- Verified that ROS master and Matlab's global node were connected.
- Uploaded workspace data from Matlab Online to the desktop Matlab
- Established publishing node to topic of interest.

Stay tuned for updates within Appendix B for more ROS updates (which includes comparison plots).

5.1.4 Literature Comparisons

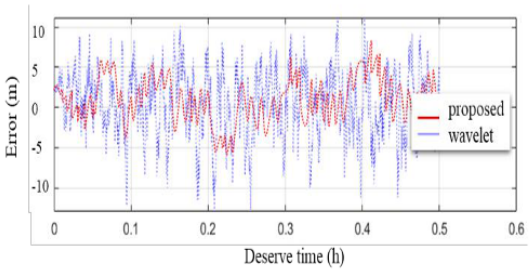
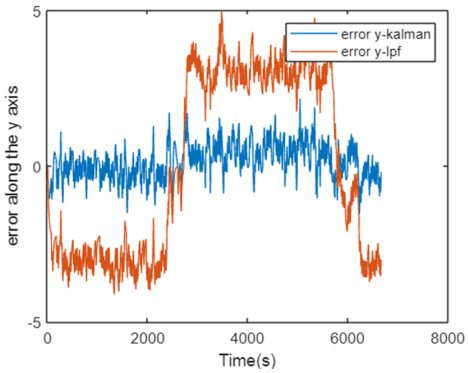
Kumalasari et al. latitudinal (Y) and longitudinal (X) response plots where they found that the ideal values for Q and R = 10^{-6}. Their R and Q values are much lower since their units are with respect to the units of the geodetic plots (decimal degrees).

Table 10. Comparisons of Literary Works and Kalman Filter.

Literary Work Results	Kalman filter Results
 Fig. 11. Graph of Longitude measurement with Longitude prediction with $R=10^{-6}$ and $Q=10^{-6}$ (1 loc)	 R = 4, Q=0.15
 Fig. 15. Graph of Latitude measurement with Latitude prediction with $R=10^{-6}$ and $Q=10^{-6}$ (>1 loc)	 R = 4, Q=0.15

Song et al's NS for their UAV consisted of a sensor fused (GPS + camera) Kalman filter. Their system improved the positioning accuracy by 0.8m - with camera data being less noisy than GPS when switching between primary and auxiliary modes. The Kalman filter of this study had an accuracy of 99.9% in the y direction and a 94.9% accuracy in the x direction.

Table 11. Comparisons of Literary Works and Kalman Filter.

Literary Work Results	Kalman filter Results
 Figure 5. Positioning accuracy analysis of gps as the main navigation mode	 R = 4, Q=0.15

Chapter 6 - Conclusion and Future Works

6.1 Conclusion and Future Works

6.1.1 Background

The Kalman filter was constructed and comparisons were made with respect to the sensor data collected via smartphone only (due to time constraints) instead of the individual sensors (MPU9250 and GPS). The plots in section 5.1.2 showed that the sensor fused linear Kalman filter, as well as, the low pass filter smoothed the tracking resolution of the GPS. It is critical to note that the Kalman filter is a combination of the state (A,B,noise) and observer (L, C, z) parts. The Kalman filter predicts the next state via control inputs while the low pass filter refines those predictions via the measured states. This study selectively fused GPS, accelerometer and magnetometer to develop a simple but robust sensor fused Kalman filter. The performance of the observer model in the x direction is 95%, while the y direction is at 99.9%. The performance of the low pass filter in the x direction is 86% while in the y direction is at 99.7%.

6.1.2 Conclusion and Observations

The Kalman filter and the low pass filter did as expected, which was to reduce the error (or enhance the resolution) of the lower frequency measurement (GPS); and in the Kalman filter's case, through the inference of new measurements despite the noisy accelerometer and magnetometer's inputs. The ideal Kalman filter's gain (L) values were different for the two estimated states ('k' and 'k+1') ; with the observer's gain value higher for the future predicted step at R=4. The eigenvalues for the open state system (eig(A)) and closed observer model (E) are within the bounded unit circle (values between -1 and 1); thus, the system is stable. It is critical to note that the study used a discontinuous system, for a continuous system follows different expectations with respect to what values the hyperparameters (such as poles) need to be in order to be considered stable. Please, refer to Appendix A to find the final Matlab scripts for further information and explanations.

The greater the alpha value becomes, the closer the error curves between the Kalman and low pass filter becomes. However, there is a threshold; as shown, with an alpha of 0.99 (meaning that the GPS values are barely considered in the low pass filter equation) a greater offset from the measured GPS and estimated states of the Kalman filter is observed. The ideal alpha value = 0.95, where the low pass filter is closest to the Kalman filter, however a higher level of accuracy is compensated with a delay. Furthermore, when fine tuning the Kalman filter, the higher the R value, error also increases in the estimated states. R states the level of confidence the measured

data's values have with respect to the observer model. Thus, the higher level of R is a higher level of variance or standard deviation, thus, the less confident the measured data values are considered in the observer model.

Sensor sensitivity to bad weather and trees can affect the output data, as well as, sudden jerk movements and sudden change in acceleration.

6.1.3 List of Future Works

- Completing the EKF comparison to the Kalman filter's response curve. (Stay tuned to Appendix B's github repo. It will be updated by Sept. 2022 with results.)
- Using physical individual sensors (GPS receiver + MPU9250) to MATLAB's Simulink and observing the behavior of the Kalman filter in real time. (Real time tracking)
- Using the physical individual sensors (GPS receiver + MPU9250) and connecting it to ROS's robot_pose_ekf or robot_localization packages and comparing the plot's behavior to that of the linear observer model created in this study.
- Explore whether non linear filters can track less than or equal to a meter distance in the exact change of movement - rather than just a 'smoothing' of the GPS resolution in between points.
- Tune Kalman filter via Q and leave R constant.
- Include camera and lidar as part of the INS.

REFERENCES

[1] O'Kane, J.M., "A Gentle Introduction to ROS," 2.1.6(ab984b3), Department of Computer Science and Engineering, University of South Carolina. Columbia, SC, 2014..

[2] Thalel, S.P; Prabhu, M.M; Thakur, P.V; Kadam, P., "ROS based SLAM implementation for Autonomous Navigation using Turtlebot," *ITM Web of Conferences 32, 01011*, *ICACC,* Electronics Engineering Dept., Ramrao Adik Institute of Technology, Navi Mumbai, India., 2020. Doi: https://doi.org/10.1051/itmconf/20203201011

[3] Menna, B.V; Villar, S.A; Rozenfeld, A.; Acosta, G.G, "GPS aided strapdown inertial navigation system for autonomous robotics applications," *IEEE.,* 2017.

[4] JPL, "Project Tango – Visual SLAM Development," *JPL, Robotics, Completed Research Tasks.,* 2017. Doi: https://www-robotics.jpl.nasa.gov/tasks/showTask.cfm?FuseAction=showTask&TaskID=256&tdaID=700060

[5] Erickson, L.K, "Space Flight: History, Technology, and Operations," The Scarecrow Press, Inc., Lanham, Maryland; Plymouth, UK, 2010, Chapter 5.

[6] University of Mississippi, "GPS and Motion Sensors," *How Stuff Works*, 2020. Doi: http://faculty.bus.olemiss.edu/breithel/b620s02/pepper/gps.htm

[7] Dr. Enge; Dr. Diggelen, V., "GPS: An Introduction to Satellite Navigation," Stanford University, 13 October 2014. html: https://www.youtube.com/playlist?list=PLGvhNIiu1ubyEOJga50LJMzVXtbUq6CPo [retrieved 22 August 2020]

[8] Kim, Min Ji; Bae, Seol B; Joo, Moon G; Park, James J; Stojmenovic, Ivan; Jeong, Hwa Young; Yi, Gangman, "A Simplified Hybrid Navigation System Design of a Mobile Robot Using Kalman Filter," 2015. Doi: 10.1007/978-3-662-45402-2_3

[9] Motlagh, H.D.K; Lotfi, F.; Taghirad, H.D; Germi, S.B, "Position Estimation for Drones based on Visual SLAM and IMU in GPS-denied Environment," *IEEE, 2019 7th International Conference on Robotics and Mechatronics (ICRoM),* 2020. Doi: 10.1109/ICRoM48714.2019.9071826

[10] Kellalib, B.; Achour, N.; Demim, F., "Sensors Faults Detection and Isolation using EKF-SLAM for a Mobile Robot," *IEEE Xplore,* 02 March 2020. Doi: 10.1109/ICAEE47123.2019.9014663

[11] Słowak, P.; Kaniewski, P., "LIDAR-based SLAM implementation using Kalman filter," *Radioelectronic*

Systems Conference, Vol. 11442, Jachranka, Poland, 2019.

[12] Parka, G. ; Choia, S.B ; Hyun, D. ; Lee, J. , "Integrated observer approach using in-vehicle sensors and GPS for vehicle state estimation," *ScienceDirect,* Vol. 50, April 2018, pp. 134-147. Doi: https://doi.org/10.1016/j.mechatronics.2018.02.004

[13] Jin, W. ; Qi, J. ; Wu, N., "State estimation of unmanned vehicles in GPS restricted environment," *IEEEXplore*, 13 February 2020. Doi: 10.1109/CAC48633.2019.8996424

[14] Oh, S.H; Hwang, D., "Low-cost and high performance ultra-tightly coupled GPS/INS integrated navigation method," *Science Direct*, Vol. 60, Issue 12, 15 December 2017, pp.2691-2706. Doi:https://doi.org/10.1016/j.asr.2017.06.007

[15] Cheng, L.L; Liu, H.B, "Examples of quadrocopter control on ROS", *IEEEXplore,* 15 February 2016. Doi: 10.1109/ICASID.2015.7405668

[16] Kiss-Illés, D,; Barrado,C; Salamí, E., "GPS-SLAM: An Augmentation of the ORB-SLAM Algorithm," *NCBI (PMC Labs)*, 2019 Nov; 19(22): 4973. doi: 10.3390/s19224973

[17] Cho, Y.S; Jang, S.H; Cho, J.S; Kim, M.J; Lee, H.D; Lee, S.Y; Moon, S.B, "Evaluation of Validity and Reliability of Inertial Measurement Unit-Based Gait Analysis Systems," *NCBI (PMC Labs)*, 2018 Dec; 42(6), pp. 872–883. doi: 10.5535/arm.2018.42.6.872

[18] Yousif, K.; Bab-Hadiashar, K.; Hoseinnezhad, R., "An Overview to Visual Odometry and Visual SLAM: Applications to Mobile Robotics," *SpringerLink*, 2015 ,pp. 289–311. Html: https://link.springer.com/article/10.1007/s40903-015-0032-7

[19] Kumalasari, I.N; Zainudin, A.; Pratiarso, A., "An Implementation of Accuracy Improvement for Low-Cost GPS Tracking Using Kalman Filter with Raspberry Pi,"*IES, IEEE,*2020. Html: https://ieeexplore-ieee-org.libaccess.sjlibrary.org/document/9231778

[20] Song, W., "An integrated GPS/vision UAV navigation system based on Kalman filter," *IEEE, ICAIIS, 2020.* html:https://ieeexplore-ieee-org.libaccess.sjlibrary.org/document/9194819

[21] Khatib, E.IA.; Jaradat, M.A; Abdel-Hafez, M.; and Roigari, M., "Multiple sensor fusion for mobile robot localization and navigation using the Extended Kalman Filter," *IEEE,* 2015.

[22] Menna, B.V; Villar, S.A; Rozenfeld, A.; and Acosta, G.G, "GPS Aided Strapdown Inertial Navigation System for Autonomous Robotics Applications,"*IEEE,* 2021.

[23] Maimone, M.; Cheng, Y.; Matthies, L., "Two Years of Visual Odometry on the Mars Exploration Rovers," *JetPropulsionLaboratory*, California Institute of Technology , Pasadena, CA. Html: https://www-robotics.jpl.nasa.gov/publications/Mark_Maimone/rob-06-0081.R4.pdf [retrieved 22 August 2020]

[24] Aqel, M.O.A; Marhaban, M.H; Saripan, I.M;. Ismail, N.B, "Review of visual odometry: types, approaches, challenges, and applications," *NCBI (PMC Labs)*, 2016 Oct 28, doi: 10.1186/s40064-016-3573-7

[25] Galoogah, H.K; Fagg, A.; Huang, C.; Ramanan, D.; Lucey, S., "Need for Speed: A Benchmark for Higher Frame Rate Object Tracking,"Robotics Institute, Carnegie Mellon University, SAIVT Lab, Queensland University of Technology. https://www.ci2cv.net/media/papers/nfs_camera_ready.pdf [retrieved 22 August 2020]

[26] Wong, N., "ros-curriculum,"*github,* SJSU Robotics Club. html: https://github.com/SJSU-Robotic [cited 22 August 2020]

[27] Deakin, R.E, "TRANSVERSE MERCATOR PROJECTION Karney-Krueger equations",html: chrome-extension://oemmndcbldboiebfnladdacbdfmadadm/http://www.mygeodesy.id.au/documents/Karney-Krueger%20equations.pdf

[28] DISRIG, "Coordinate Systems," html: http://www.dirsig.org/docs/new/coordinates.html

[29] ASPEXIT-Precise Agriculture, "Coordinate Reference Systems," html: https://www.aspexit.com/coordinate-reference-systems/

[30] Sunny, "Art of Directional Drilling", html: https://directionaldrillingart.blogspot.com/2015/09/universal-transverse-mercator-what-is.html

[31] Zolfaghari, M.; Amini, E.; Mozaffari, H., "Analysis of Universal Transverse Mercator projection and Coordinate System," Amirkabir University of Technology (Tehran Polytechnic), December 2012 html:chrome-extension://oemmndcbldboiebfnladdacbdfmadadm/http://amini.zohosites.com/files/Asli.pdf

[32] Chen, J., "Projection Parameters" html:http://www.geography.hunter.cuny.edu/~jochen/gtech361/lectures/lecture04/concepts/Map%20coordinate%20systems/Projection%20parameters.htm

[33] Clynch J., "Coordinates", Naval Postgraduate School, 2002, html: chrome-extension://oemmndcbldboiebfnladdacbdfmadadm/https://www.oc.nps.edu/oc2902w/coord/coord.pdf

[34] Thomas, A., "Datums, Projections and Coordinate Systems", *Recorder,* Enerplus Corporation, Calgary, Canada,

html: https://csegrecorder.com/articles/view/datums-projections-and-coordinate-systems

[35] Clynch J., “Geodetic Coordinate Conversions”, Naval Postgraduate School, 2002, html:

chrome-extension://oemmndcbldboiebfnladdacbdfmadadm/https://www.oc.nps.edu/oc2902w/coord/coordcvt.pdf

[36] “Chapter 2 Coordinate Systems and Transformations “, html:

chrome-extension://oemmndcbldboiebfnladdacbdfmadadm/http://www.newbooks-services.de/MediaFiles/Texts/4/9780857296344_Excerpt_002.pdf

[37]Sanderson, Grant, “Visualizing quaternions (4d numbers) with stereographic projection”, *3Blue1Brown*, html:

https://www.youtube.com/watch?v=d4EgbgTm0Bg

[38]Sanderson, Grant, “Quaternions and 3d rotation, explained interactively”, *3Blue1Brown*, html:

https://www.youtube.com/watch?v=zjMuIxRvygQ

[39] Garcia, M., “Factored Quaternion Algorithm”, *AHRS,* 2019-2021, html:

https://ahrs.readthedocs.io/en/latest/filters/fqa.html

[40] VECTORNAV, “Educational Material”, html :

https://www.vectornav.com/resources/inertial-navigation-primer/math-fundamentals/math-refframes

[41] Mathworks, Matlab, “Capturing Azimuth, Pitch, and Roll Example”, html:

https://www.mathworks.com/matlabcentral/mlc-downloads/downloads/submissions/40876/versions/8/previews/sensorgroup/Examples/html/CapturingAzimuthRollPitchExample.html [date retrieved: 2/27/22]

[42] Mathworks, Matlab, “MPU9250 IMU Sensor”, html:

https://www.mathworks.com/help/supportpkg/arduino/ref/mpu9250imusensor.html [date retrieved: 2/27/22]

[43] Mathworks, Matlab, “Model IMU, GPS, and INS/GPS”, html:

https://www.mathworks.com/help/fusion/gs/model-imu-gps-and-insgps.html [date retrieved: 2/27/22]

APPENDICES

APPENDIX A:

1. Github repository link to Matlab codes: https://github.com/tbejaoui1/ME_Masters_Matlab_Code.git

APPENDIX B:

2. Github repository link to ROS1 codes: https://github.com/tbejaoui1/ME_Masters_ROS_SetUp_Code.git

[will be completed by Sept. 2022]