Table of Contents

Table Of Figures

Introduction

Sensor Fusion plays a vital role in modern navigation systems by fusing the data from multiple Sensors to provide accurate and robust estimates of a system's state, which can comprise position, velocity, and orientation. In the increasingly autonomous world, precise navigation is necessary for many applications, including unmanned aerial vehicle (UAV), Robotics, Aeronautics, and Augmented Reality. The Fusion of sensors, such as Inertial Measurement Units (IMUs), Global Positioning Systems(GPS), Magnetometers, radars, LIDAR , and cameras, allows for complementary information to be leveraged, mitigating the limitations and uncertainties of the individual sensors. This Report Explores the concept of sensor fusion for Better Position Estimation and the design of a reliable, Low-cost multi-sensor Fusion system.

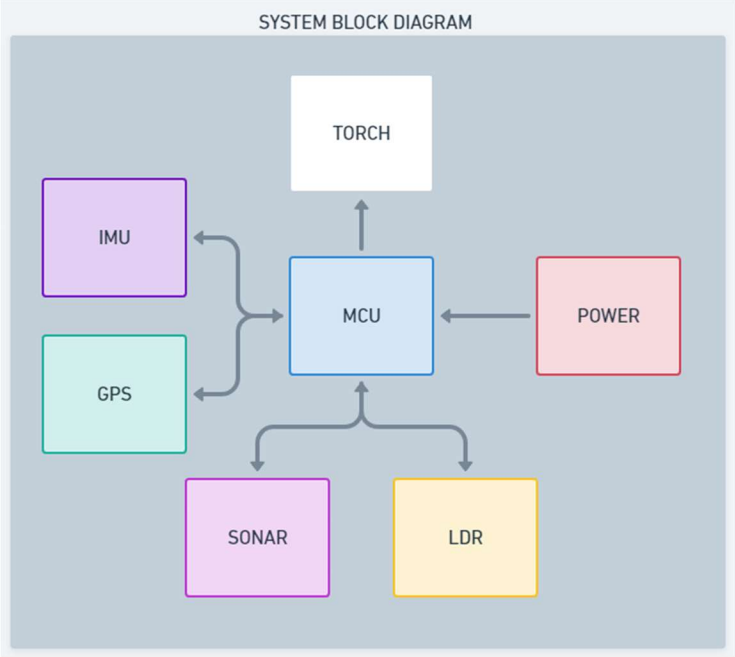

Figure 1: System Block Diagram

The System Block diagram consists of the components required for the fusion process that include sensors such as Ultrasonic, GPS, IMU, LDR and a Microcontroller to control and do the fusion, Powered by a Low Voltage Battery or PC. It also consists of an LDR that helps in the automatic switching of the Torch, Which is attached to the system this gives the ability of the system to work in day and night, if further cameras are added. Currently, This report discusses the use of Pixhawk and Arduino Microcontrollers usage.

Hardware and Sensors

i) MPU 6050 (Accelerometer and Gyroscope)

The MPU 6050 is a popular MEMS-based low-cost Inertial Measurement (IMU) that combines a 3-axis accelerometer and a 3-axis gyroscope into a single package. The accelerometer measures linear acceleration along the x,y, and z axes. Similarly, the Gyroscope detects the angular velocity along the three axes. The MPU 6050 provides crucial motion-related data like the position, and velocity by integration of the acceleration, which included drift over time. The MPU 6050 also comes with an ambient temperature sensor that measures the temperature in its surroundings, which helps tackle the IMU drift due to temperature.

Figure 2 : MPU 6050

Measured Quantities from the sensor are Roll, Pitch, and Yaw angles, calculated from the fusion of Accelerometer and Gyroscope Data. Acceleration in three axes, which is integrated to get position data.

ii) U-blox NEO M8N GPS Sensor

The U-blox NEO-M8N is a high-precision (2m) Global Positioning System(GPS) sensor that provides accurate global position and velocity information. GPS is essential for absolute positioning, especially in environments where satellite signals are available. The NEO-M8N integration with the MPU 6050 and other sensors through sensor fusion techniques allows for improved position and velocity estimates, compensating for GPS signal outages or inaccuracies in dense urban environments.

Figure 3 : GPS Module

Measured Quantities are Position, i.e., Latitude, Longitude, and Altitude, and Velocity, i.e., Velocity in the x,y, and z axes. These quantities help correct the predicted estimates from IMU using Kalman Fusion.

iii) Ultrasonic Sensor HC - SR04

The HC-SR04 ultrasonic sensor is a distance-measuring device based on the reflection of sound(echo) principle. It emits an ultrasonic pulse through one of the opening - Transmitters and measures the time it takes for the pulses to bounce back after hitting the object with the help of the other opening - Receiver. By Calculating the round trip time, the sensor can accurately determine the distance to the obstacle. The context of using the ultrasonic Sensor is to detect the obstacles or objects in the platform. The distance measurements in front of the HC-SR04 provide valuable input for collision avoidance and obstacle detection, enhancing the platform's safety.

Figure 4 : HC SR04

The measured quantity from the sensor is Distance (cm). Three Sensors are used for the accurate detection of the obstacle. The setup and implementation of this sensor will be discussed in the coming sections.

iv) Light-Dependent Resistor (LDR)

The Light-Dependent Resistor, or LDR, is a passive sensor that changes resistance based on the ambient light intensity. The principle of LDR is based on Photoconductivity. As the Surrounding light conditions vary, the resistance of the LDR changes accordingly. When the ambient light threshold is, the LDR detects the change and triggers the torch or light source, ensuring adequate visibility in low-light conditions or during darkness.

Figure 5 : LDR Sensor

The measured quantity from the sensor is the Resistance which will be measured in the ADC value from the MCU like Arduino UNO.

v) Arduino

The Arduino Uno is a versatile open-source microcontroller board with numerous I/O pins and an ATmega328P microcontroller. It's highly suitable for sensor fusion projects due to its ease of programming, compatibility with various sensors, and real-time data processing capabilities. The active Arduino community provides extensive support and resources for sensor fusion development. Moreover, its affordability makes it accessible for educational institutions and hobbyists. Users can customize and expand its functionality with shields and modules. Arduino Uno also offers scalability and compatibility with other Arduino boards for complex sensor fusion systems. In summary, it's a cost-effective, user-friendly solution for sensor fusion applications.

Figure 6: Arduino Uno

vi) Pixhawk

The Pixhawk 6C represents an advanced open-source autopilot system, well-suited for academic and research endeavors. It boasts robust processing capabilities and a modular architecture, enabling seamless integration with a wide array of sensors. This system excels in achieving high levels of autonomy, including autonomous navigation and obstacle avoidance, making it ideal for academic research and experimentation. Safety and redundancy features ensure the reliability and security of missions, while telemetry and data logging capabilities aid in real-time data analysis. Supported by a thriving community of experts, the Pixhawk 6C serves as an indispensable tool for academic pursuits in fields such as robotics, UAV technology, and autonomous systems.

Figure 7: Pixhawk 6c

Data Acquisition

LDR

In this report, the below Circuit is implemented in TINKERCAD Simulator to test the working of the LDR for switching the torch(LED strip) available in the system when low light is detected. The Light Dependent Resistors are connected with the Arduino UNO with a resistor connected to one of the terminals of LDR for Protection purposes.

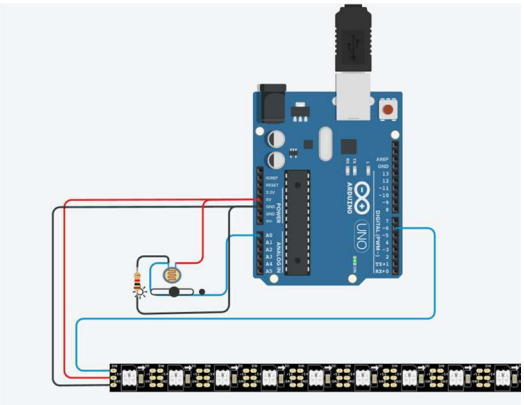

Figure 8 : LED in OFF State when sunlight is present

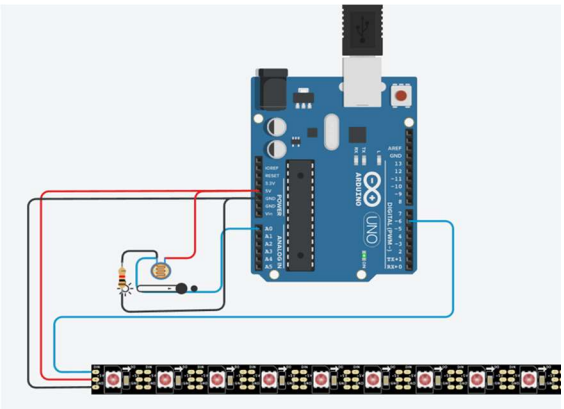

Figure 9 : LED in ON State when sunlight is absent

The color coding of the above simulation is Red - Power Supply, Black - Ground, and Blue - Signal. The Threshold Set for the Switching is 150 ADC Value. From the Above, we can observe the following points:
i) LED Strip Turned off while detection of light by LDR.
ii)LED Strip Glowing Red while detection of light by LDR.(i.e., when Analog value from LDR <150)

Filters

Kalman Filter

The Kalman Filter is a mathematical tool that estimates a system's state based on incomplete and noisy measurements. It is an estimator to predict any state or part in the signal which contains the signal. The result of the estimation process is similar to reducing the noise from the signal, which is one of the reason it is called as Kalman Filter.[3] Initially developed for aerospace applications, the Kalman Filter has found applications in diverse domains like robotics, aerospace, and sensor fusion. The Kalman Filter is also known as BLUE (Best Linear Unbiased Estimator). This is one of the reason for selecting Kalman Filter for Sensor Fusion and others include it's consistency, lowest variance estimation etc.

The Kalman Filter combines predictions from a dynamic model with noisy measurements to produce a filtered estimate of the actual system state. This process involves two essential steps.

They are the Prediction step (also known as the estimation step) and the correction step (also known as the update step). In the prediction step, the Kalman Filter Uses the system's dynamics to predict the current state estimate into the future. In the correction step, the predicted estimate is corrected based on the incoming measurements (from GPS), incorporating both the measurement and the model's uncertainty. The equations for these states can be found below.

Implementation:

The Kalman Filter in One Dimension has been implemented on the Sonars averaged Data[4]. Three HC- SR04 are connected to the Arduino UNO as shown in the picture and the data from the all the three sensors have been averaged for noise elimination. And this Averaged Data is fed into the Kalman Filter which again reduced the Noise in the Averaged Data. The Following Simulation was also done in TINKERCAD and also it was Practically Implemented and the results of the practical test are shown in the graph.

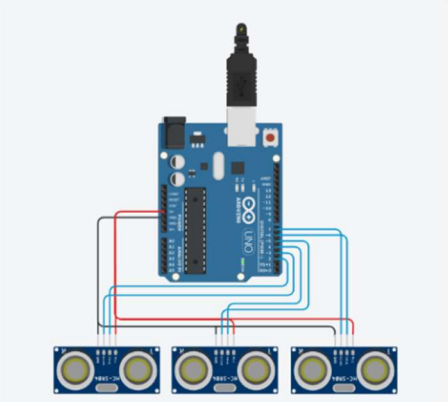

Figure 10 : Simulation of HC SR04

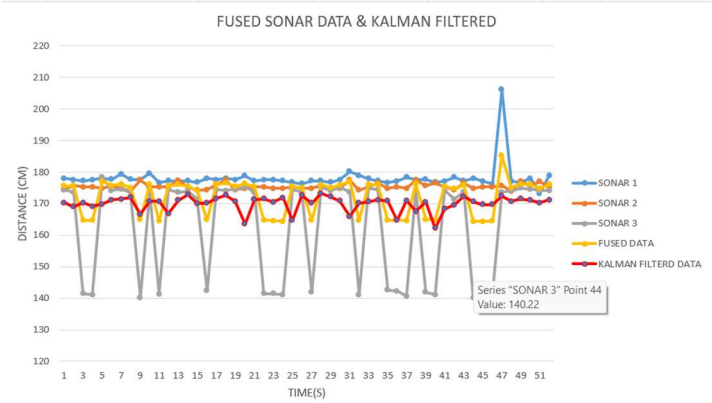

Figure 11 : Results with fused output

In the Above graph visualization the Color Code is as follows, Red for Power, Black for Ground and Blue for Signal line. In Fig9, The sensor three is quite noisy and the overall measurement From the sensors is the averaged data. The Kalman filter gets its input from the Averaged Data. The Above setup assumes the system is in Rest condition and the three sensors are detecting a single object in front of them.

Kalman Filter In Multi Dimension

The Multi dimension Kalman Filter extends the principles of the scalar Kalman Filter to handle vectors of state variables. Here, We can be able to fuse the data with the help of multi sensors. This Could include systems of multiple physical quantities or attributes that need to be estimated simultaneously. For example, In a robotic System, the state might include position, velocity, orientation etc all of which are interrelated.

This Kalman Filter Requires the motion and measurement models, similar to the one dimensional Kalman Filter this will also require two major steps, they are 1) Prediction 2) Correction.

The Mathematics of the Multidimensional Kalman Filter is as follows [3][1]

1) The Predicted State

$$x_{kp} = Ax_{k-1} + Bu_k + w_k$$

Where, $x_{kp} = \begin{bmatrix} positionx \\ velocityx \end{bmatrix}$ Predicted State Matrix

$x_{k-1} = \begin{bmatrix} positionxk-1 \\ velocityxk-1 \end{bmatrix}$ Previous State Matrix

$A = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix}$, known as State Transition Matrix

$B = \begin{bmatrix} \frac{1}{2} * \Delta t^2 \\ \Delta t \end{bmatrix}$, known as Control Matrix

$u_k = [g]$, known as Control Vector.

w_k = State Noise

2) Initial Process Covariance

$$P_{k-1} = \begin{bmatrix} variance\_position & 0 \\ 0 & variance\_velocity \end{bmatrix}$$

Here considering Cross Covariance as '0'

3) Predicted Process Covariance Matrix

$$P_{kp} = AP_{k-1}A^T + Q_k$$

Where, P_{kp} = Predicted Process Covariance Matrix

P_{k-1} = Previous Process Covariance Matrix

Q_k = Covariance Noise

A = State Transition Matrix

4) Kalman Gain

$$K = P_{kp}H^T/(HP_{kp}H^T + R)$$

Where, $H = \begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}$, known as Observation Matrix

$R = \begin{bmatrix}var\_imu & 0\\ 0 & var\_gps\end{bmatrix}$, Estimated Measurement Error Covariance Matrix

K = Kalman Gain

5) New Observation

$$Y_k = CY_{km} + Z_k$$

Where, $Y_k =$ Measurement Matrix

Y_{km} = Measurements from External Sensor (GPS)

Z_k = External Delays

$C = \begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}$, known as Transformation Matrix

6) Calculation the Current State

$$X_k = X_{kp} + K[Y_k - HX_{kp}]$$

Where, X_k = Current/Updated State Matrix

7) Updating the Process Covariance Matrix

$$P_k = (I - KH)(P_{kp})$$

Where, P_k = Updated Process Covariance Matrix

I = Identity Matrix

8) Assign the Updated Matrices

$$X_{k-1} = X_k$$
$$P_{k-1} = P_k$$

Here, The updated Matrices will again go through iterative process of the Kalman Filter and gets updated again for the best possible estimates.

The Visual Representation of the Above Equations and their flow is represented with the help of Block Diagram.

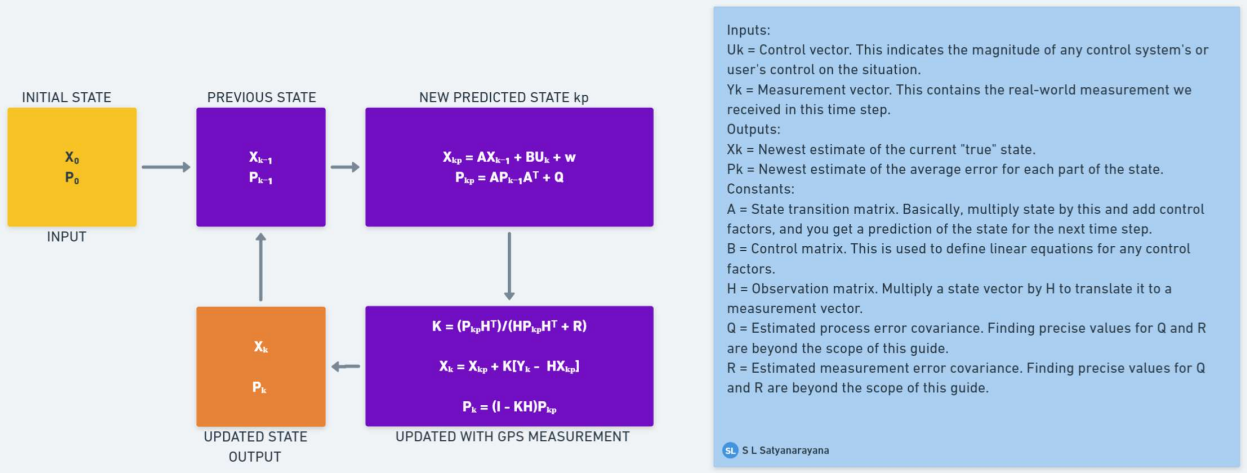

Figure 12 : Kalman Filter Block Diagram

The Python Code for the Above Kalman Filter is implemented with GPS Connected to the Pixhawk and using the Integrated IMU Sensor in the Pixhawk.

Extended Kalman Filter [2]

Inertial Measurement Units (IMUs) and Global Navigation Satellite Systems (GNSS) are widely used sensors in navigation and localization applications. IMUs provide high-frequency measurements of acceleration and angular velocity, while GNSS offers absolute position and velocity information. However, both sensor types have inherent limitations and errors due to noise, biases, and environmental conditions. Sensor fusion techniques, particularly the Extended Kalman Filter (EKF), have proven effective in combining these disparate data sources to enhance navigation accuracy, especially in challenging environments.

The Extended Kalman Filter is an extension of the traditional Kalman Filter, designed to handle nonlinear system models. It addresses situations where the state transition and observation models are nonlinear. The EKF operates iteratively, performing prediction and update steps to estimate the system's state while incorporating sensor measurements and reducing uncertainty. The key innovation of the EKF lies in linearizing the nonlinear functions using Jacobian matrices, allowing the filter to operate in the Gaussian domain. Similar to the Kalman Filter this Filter also includes the Prediction and Update Step for the fusion of IMU and GPS Sensors.

The Table shows the comparision of Tightly Coupled EKF, Loosely coupled EKF:

Attribute	Tightly Coupled	Loosely Coupled
GNSS Measurement	Pseudo Ranges to Satellites	Only Position
Accuracy	HIgher	Lower
Complexity	Higher	Lower

For a Simplistic approach and due to time constraints, this report will discuss the implementation of Error State Extended Kalman Filter in a Loosely Coupled System.

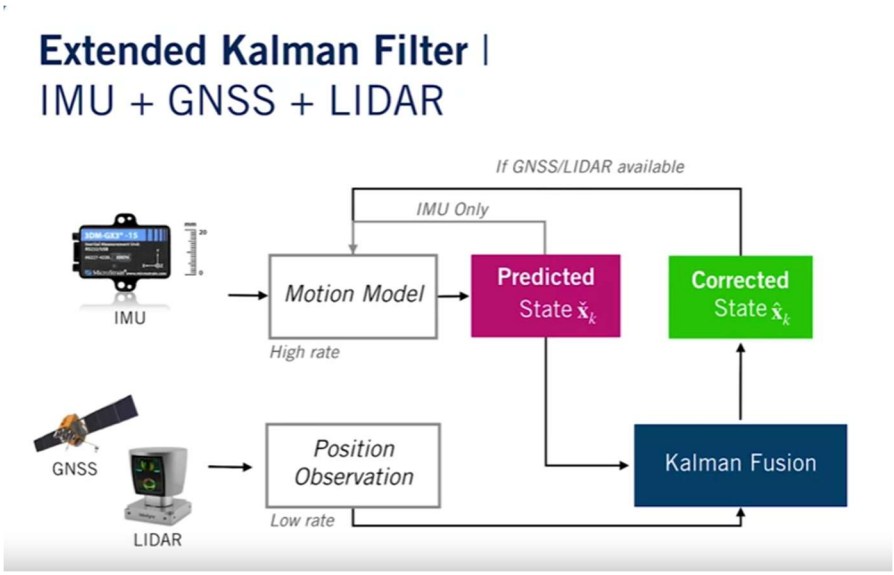

Figure 13 : Extended Kalman Filter Block Diagram of Loosely Coupled system [2]

The Motion Model of the Loosely Coupled system contains the following,

$$\text{Position} \qquad \mathbf{p}_k = \mathbf{p}_{k-1} + \Delta t\, \mathbf{v}_{k-1} + \frac{\Delta t^2}{2}\left(\mathbf{C}_{ns}\mathbf{f}_{k-1} + \mathbf{g}\right)$$

$$\text{Velocity} \qquad \mathbf{v}_k = \mathbf{v}_{k-1} + \Delta t\left(\mathbf{C}_{ns}\mathbf{f}_{k-1} + \mathbf{g}\right)$$

$$\text{Orientation} \qquad \mathbf{q}_k = \mathbf{q}_{k-1} \otimes \mathbf{q}(\boldsymbol{\omega}_{k-1}\Delta t) = \boldsymbol{\Omega}\left(\mathbf{q}(\boldsymbol{\omega}_{k-1}\Delta t)\right)\mathbf{q}_{k-1}$$

where...

$$\mathbf{C}_{ns} = \mathbf{C}_{ns}(\mathbf{q}_{k-1}) \qquad \boldsymbol{\Omega}\left(\begin{bmatrix} q_w \\ \mathbf{q}_v \end{bmatrix}\right) = q_w \mathbf{1} + \begin{bmatrix} 0 & -\mathbf{q}_v^T \\ \mathbf{q}_v & -[\mathbf{q}_v]_\times \end{bmatrix} \qquad \mathbf{q}(\boldsymbol{\theta}) = \begin{bmatrix} \cos\frac{|\boldsymbol{\theta}|}{2} \\ \frac{\boldsymbol{\theta}}{|\boldsymbol{\theta}|}\sin\frac{|\boldsymbol{\theta}|}{2} \end{bmatrix}$$

And also, For the Extended Kalman Filter to work, the system should be a linear system, so the linearized motion model will be

Error State

$$\delta \mathbf{x}_k = \begin{bmatrix} \delta \mathbf{p}_k \\ \delta \mathbf{v}_k \\ \delta \boldsymbol{\phi}_k \end{bmatrix} \in R^9$$

3x1 rotation error

Error Dynamics

$$\delta \mathbf{x}_k = \mathbf{F}_{k-1} \delta \mathbf{x}_{k-1} + \mathbf{L}_{k-1} \mathbf{n}_{k-1}$$

measurement noise

where...

$$\mathbf{F}_{k-1} = \begin{bmatrix} \mathbf{1} & \mathbf{1}\Delta t & 0 \\ 0 & \mathbf{1} & -[\mathbf{C}_{ns}\mathbf{f}_{k-1}]_\times \Delta t \\ 0 & 0 & \mathbf{1} \end{bmatrix} \qquad \mathbf{L}_{k-1} = \begin{bmatrix} 0 & 0 \\ \mathbf{1} & 0 \\ 0 & \mathbf{1} \end{bmatrix}$$

$$\mathbf{n}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_k) \sim \mathcal{N}\left(\mathbf{0}, \Delta t^2 \begin{bmatrix} \sigma^2_{\text{acc}} & \\ & \sigma^2_{\text{gyro}} \end{bmatrix}\right)$$

$\mathbf{1}$ is the 3×3 identity matrix

For the Implementation of the EKF , the mathematics involved is discussed as below:

1) Update the state with the IMU inputs

$$\check{\mathbf{x}}_k = \begin{bmatrix} \check{\mathbf{p}}_k \\ \check{\mathbf{v}}_k \\ \check{\mathbf{q}}_k \end{bmatrix}$$

$$\check{\mathbf{p}}_k = \mathbf{p}_{k-1} + \Delta t \mathbf{v}_{k-1} + \frac{\Delta t^2}{2}\left(\mathbf{C}_{ns}\mathbf{f}_{k-1} + \mathbf{g}_n\right)$$

$$\check{\mathbf{v}}_k = \mathbf{v}_{k-1} + \Delta t\left(\mathbf{C}_{ns}\mathbf{f}_{k-1} + \mathbf{g}_n\right)$$

$$\check{\mathbf{q}}_k = \boldsymbol{\Omega}\left(\mathbf{q}(\boldsymbol{\omega}_{k-1}\Delta t)\right)\mathbf{q}_{k-1}$$

2) Propagate Uncertainty

$$\check{\mathbf{P}}_k = \mathbf{F}_{k-1}\mathbf{P}_{k-1}\mathbf{F}_{k-1}^T + \mathbf{L}_{k-1}\mathbf{Q}_{k-1}\mathbf{L}_{k-1}^T$$

Can either be $\hat{\mathbf{P}}_{k-1}$ or $\check{\mathbf{P}}_{k-1}$

3) If GNSS or GPS Position is available:
   i. Compute the Kalman Gain

$$\mathbf{K}_k = \check{\mathbf{P}}_k\mathbf{H}_k^T(\mathbf{H}_k\check{\mathbf{P}}_k\mathbf{H}_k^T + \mathbf{R})^{-1}$$

One of $\mathbf{R}_{\text{GNSS}}$,

ii. Compute the Error State

$$\delta \mathbf{x}_k = \mathbf{K}_k(\mathbf{y}_k - \check{\mathbf{p}}_k)$$

iii. Correct Predicted State

$$\hat{\mathbf{p}}_k = \check{\mathbf{p}}_k + \delta \mathbf{p}_k$$
$$\hat{\mathbf{v}}_k = \check{\mathbf{v}}_k + \delta \mathbf{v}_k$$
$$\hat{\mathbf{q}}_k = \mathbf{q}(\delta \boldsymbol{\phi}) \otimes \check{\mathbf{q}}_k \longleftarrow \text{global orientation error}$$

iv. Compute Corrected Covariance Matrix

$$\hat{\mathbf{P}}_k = (\mathbf{1} - \mathbf{K}_k \mathbf{H}_k)\check{\mathbf{P}}_k$$

In implementation of the Extended Kalman Filter we computed the states with the help of error state which reduces the computation complexity of the algorithm, this method is called as Error State Extended Kalman Filter. The assumptions considered in this Algorithm are

- IMU has no biases (Where in real life IMU comes with biases).
- GNSS has no delay.
- State Initialization is Provided by the User.

Methodology

This Section explains the Algorithm of the Kalman Filter and Error State Extended Kalman Filter through flowchart

i) Kalman Filter

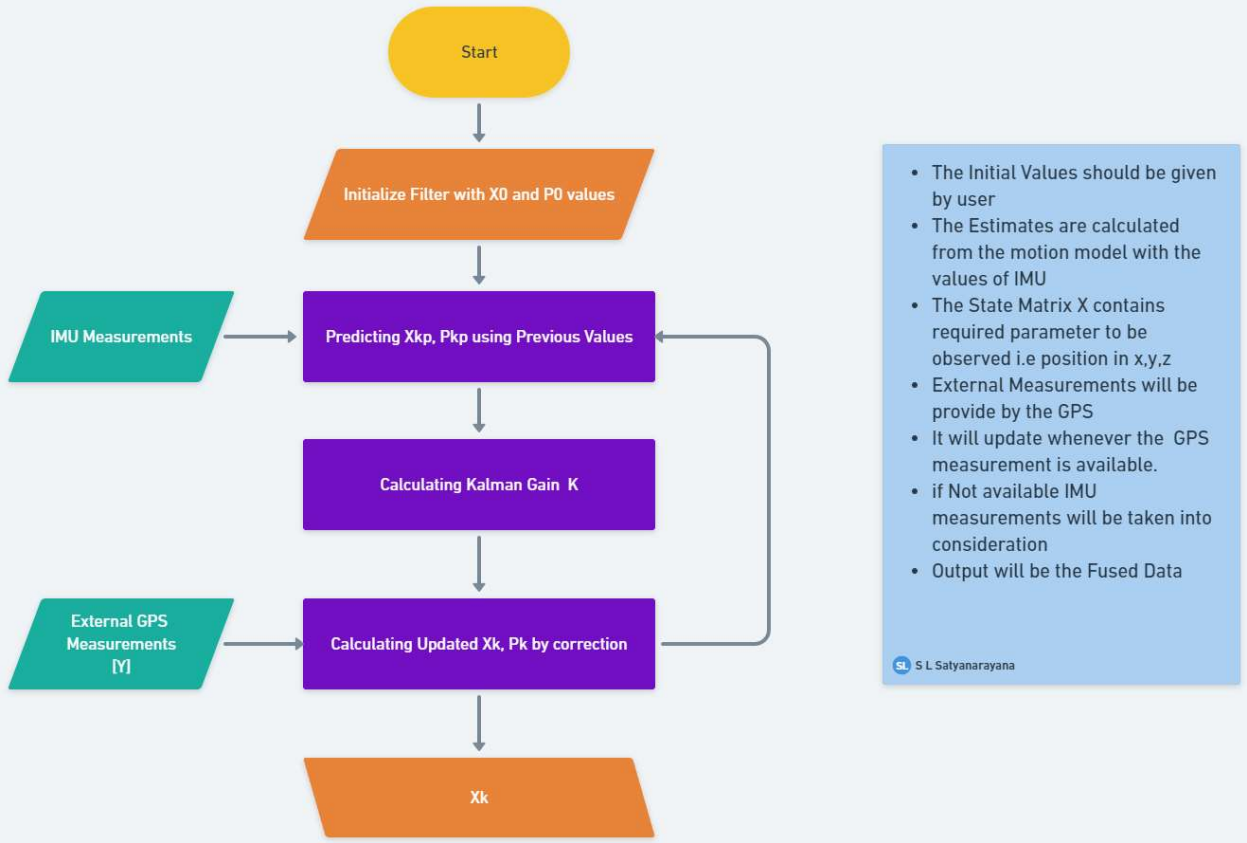

Figure 14 : Kalman Filter Flowchart

The Kalman Filter is implemented using Pixhawk Flight Controller and GPS Sensor with python language. The Flowchart of the Code Implemented is as shown.

The code takes the Initial inputs from the user and predicts the state matrix with available measurements from IMU and initializes Covariance Matrix P using the initial value given by the user. Using the Predicted Matrices the Kalman Gain is calculated which acts as weighting factor between the Predicted and External Measurement (GPS). Finally the updated state matrix is the output from the code which contains position and velocity data. All the Variables shown in the Flowchart was discussed Filter Section. By Implementing this code the Fused data is obtained. The Results obtained through the Kalman Filter implementation is discussed in the Results Section.

ii) Error State Extended Kalman Filter

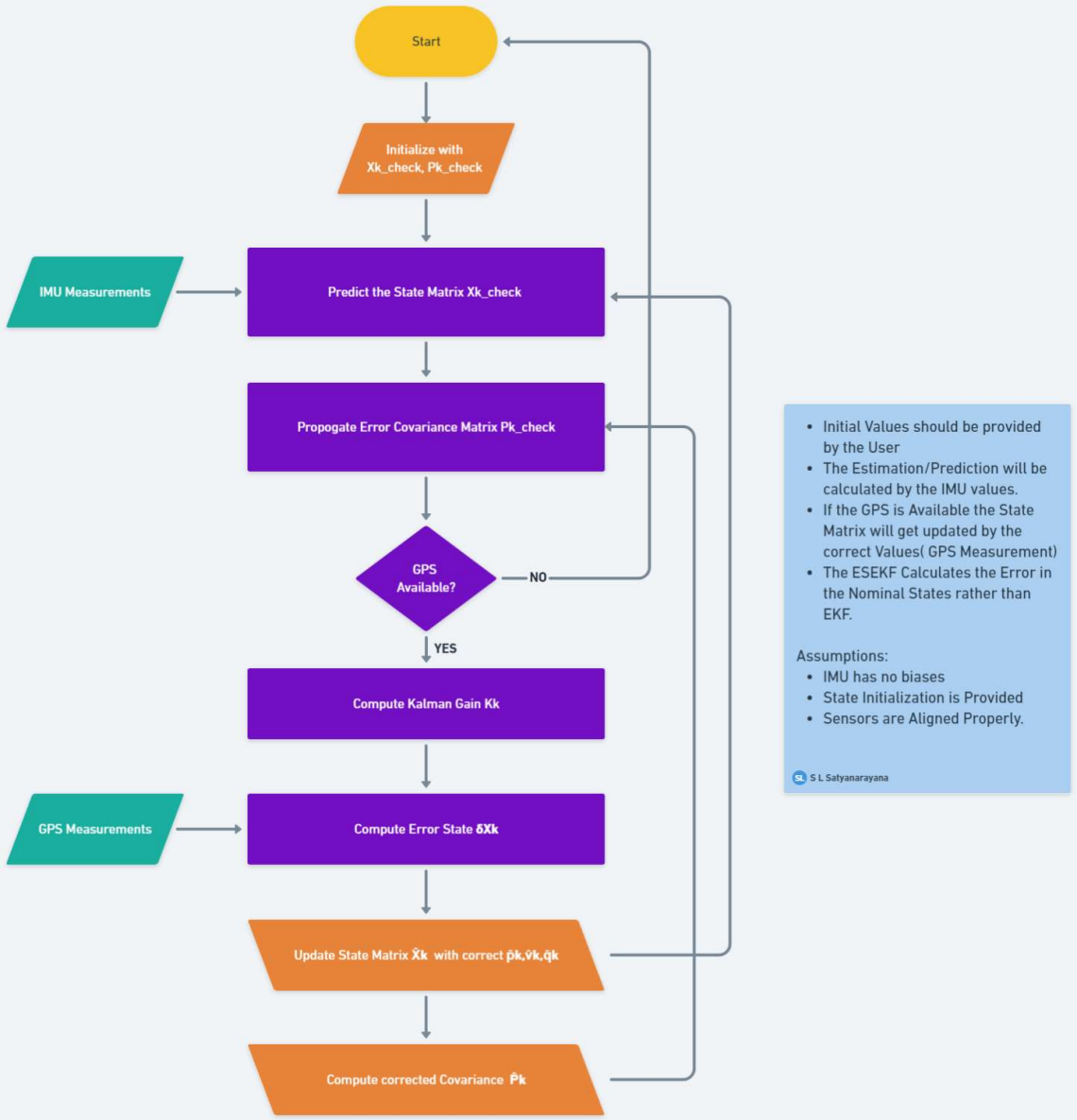

Figure 15 : ESEKF Flow Chart

The Error State Extended Kalman Filer is implemented using the IMU, GPS Data from the CARLA simulator. Where the Ground truth data is also generated from the CARLA Simulator. Using the Ground truth Data we can compare our output data to optimize and tune the code. The code is also implemented using Python and the flow chart of the code is as shown below.

The Code follows the Mathematics of the EKF discusses earlier in the Filter Section. It takes inputs from the IMU continuously as it has high data rate and corrects it's prediction whenever the Data from GPS is available. The output of the code will be discussed in the Results Section

Results

i) Kalman Filter

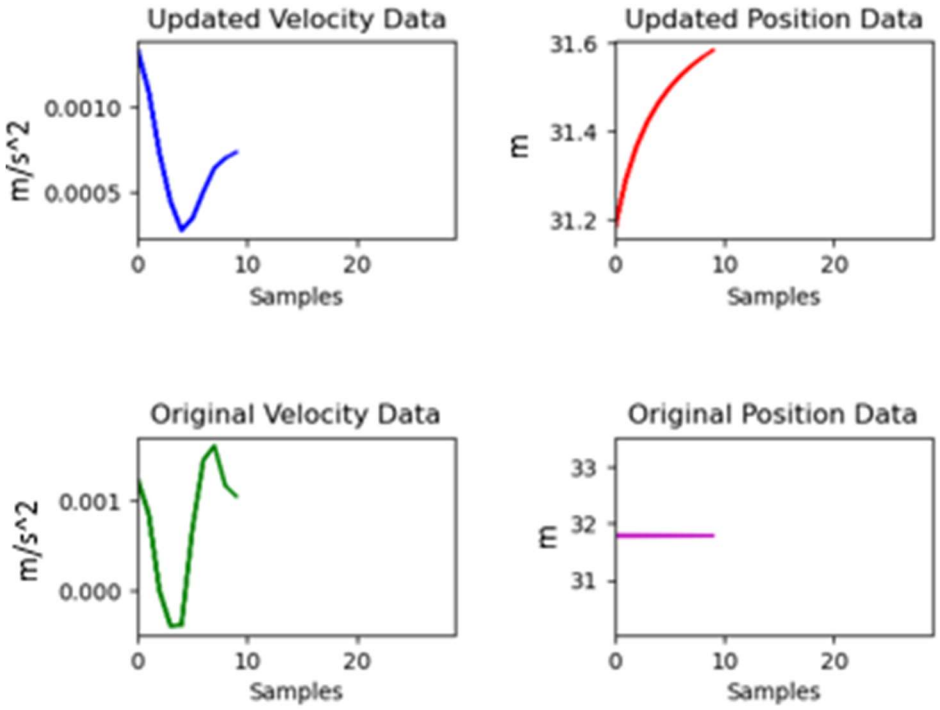

Figure 16 : Kalman Filter Results

These Results were obtained from the Pixhawk which is connected to the GPS Module. Here the Implementation is done for the first ten samples from the Pixhawk. It was initialized with Position, Velocity and acceleration Data which are inserted such that the settling time of the Kalman Filter Output will be less. The Velocity and Position Data are considered along a single Axes. Here the Updated Velocity data is initialized properly and it was with good coordination with the Original Velocity Data till some extent and after the peak value of Original Velocity data the Kalman Filter Updated it with less noise i.e. less peak at sample no 10. For the Position Data the updated value is not initialized to the resolution of 0.1 so it is taking some time to settle according to the Original Position Data that is 31.8 m. The Above Data was recorded while the Pixhawk controller setup was in rest condition, i.e. Acceleration and Velocity were in zero.

From the above we can observe that Kalman Filter performs well but it involves some error due to improper initialization and improper Calibration

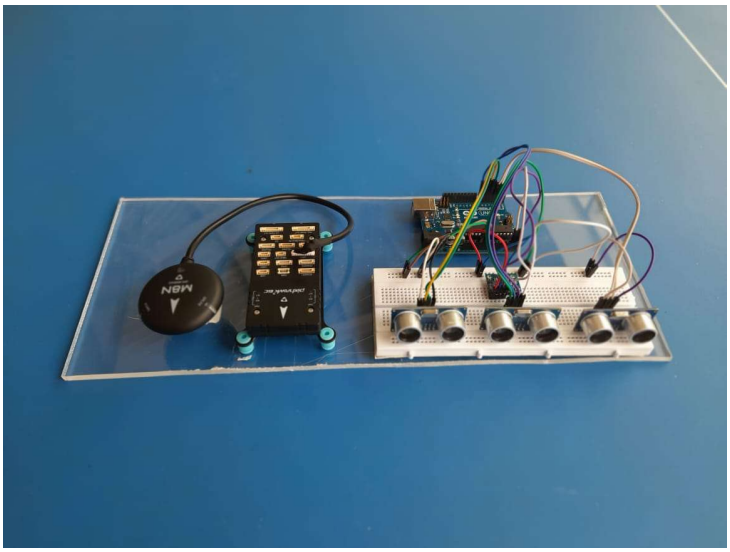

Figure 17: Fusion Setup

The Picture shows the Practical Setup of the Kalman Filter, which combines the GPS with Pixhawk, SONAR with the Arduino, As discussed Multi Dimension Kalman Fusion is Applied on Pixhawk and GPS , whereas Single Dimension Kalman Filter is applied on the SONAR Data for smoothening of the data.

ii) Error State Extended Kalman Filter

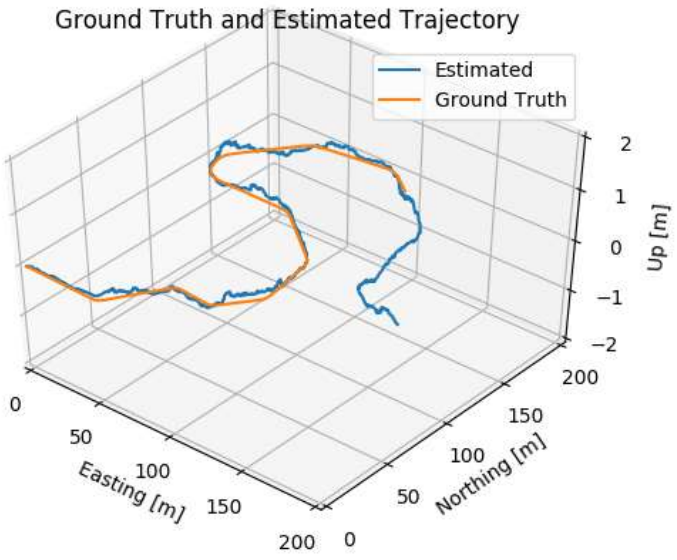

Figure 18 : ESEKF Estimation Results

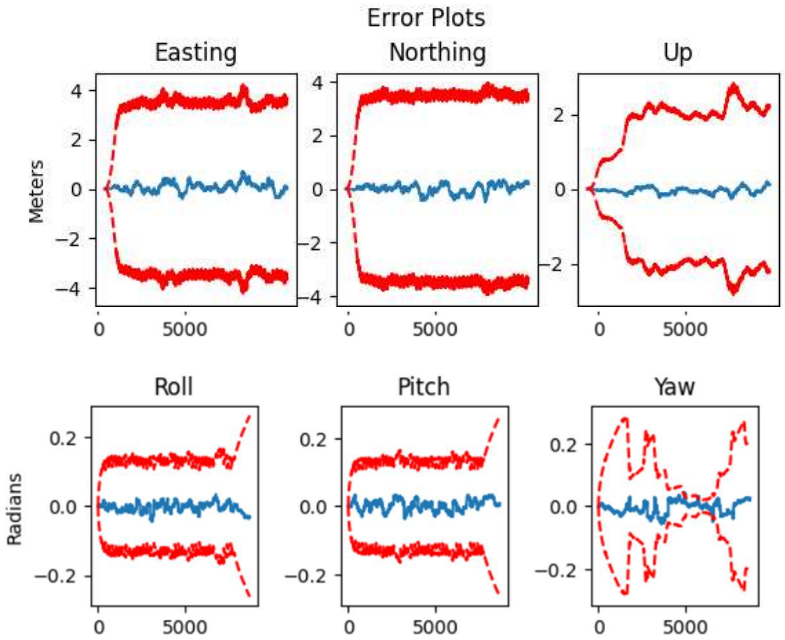

Figure 19 : ESEKF Error Results

These Results are obtained using the Simulated IMU and GPS Data that is generated from the CARLA Simulator. The Final Estimated Trajectory was obtained after a series of iterations by tuning the variables in the Code and input values. The error plots shows the error involved in estimating the trajectory and orientation. The Easting, Northing and Up represents the Position of the System in ENU Reference Frame. The Red Colored pattern represents the 3 sigma deviation of the values and the Blue pattern represents the Actual Estimation Error of the system

References

1. Van Biezen, M. (n.d.). Linear algebra [YouTube Playlist]. YouTube
2. Kelly, J., & Waslander, S. (n.d.). State estimation and localization for self-driving cars [MOOC]. Coursera1
3. Simon, D. (2006). The discrete-time Kalman filter. In Optimal State Estimation, D. Simon (Ed.). https://doi.org/10.1002/0470045345.ch5
4. Ma'arif, Alfian & Iswanto, & Nuryono, Aninditya & Alfian, Rio. (2020). Kalman Filter for Noise Reducer on Sensor Readings. Signal and Image Processing Letters. 1. 11-22. 10.31763/simple.v1i2.2.