Sensor fusion of GNSS and IMU data using Extended Kalman Filter for localization of vehicles

Ashwin Rajesh

Department of Electronics and Communication Engineering,
Government Engineering College, Thrissur

Introduction

- Localization is the task of finding the position of an object in its environment.
- It is an essential part of mobile robotic systems like **drones** and **self-driving cars**.
- The most common systems used for localization are the **GNSS** and **IMU** systems.
- These systems have their own individual limitations.
- They are complementary in nature.
- We can fuse data from both of these sensors using an estimator called the **Kalman filter**.

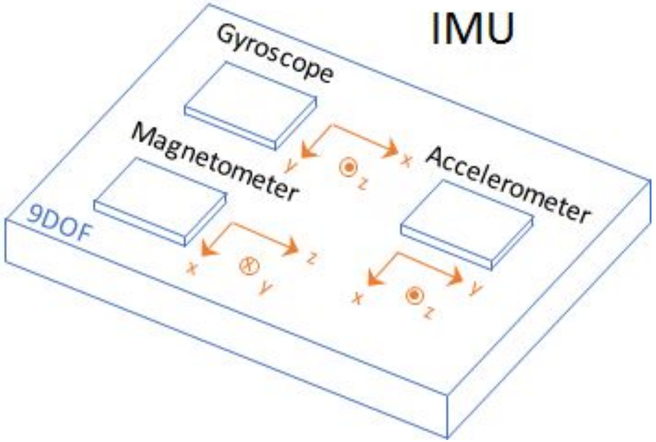
GNSS (Global Navigation Satellite System)

- Satellites in orbit send signals containing their locations in orbit and a timestamp
- Very accurate atomic clocks are used
- Difference in **time-of-flight** from multiple satellites to the receiver is used for trilateration to find position.
- **4** satellites are required to get latitude, longitude and altitude
- **3** satellites are required to get latitude and longitude data.
- Advantages
  - Location data from anywhere around the globe
- Disadvantages
  - Inaccurate (within 5-10m)
  - Low sampling rate (1-10Hz)
  - Unreliable inside tunnels

GNSS utilizes 89 satellites from all 4 satellite systems

IMU (Inertial Measurement Unit)

- Consists of 3 types of sensors
- **Gyroscopes** : Measures angular velocity
- **Accelerometers** : Measures linear acceleration
- **Magnetometers** : Measures magnetic field
- Implemented in silicon chips, as **MEMS** (Micro Electro-mechanical system) devices.
- Advantages
    - High refresh rate
    - Cheap and space-efficient
- Disadvantages
    - Linear position estimates accumulate errors



Double integration of IMU data

- For getting yaw angle, we numerically integrate angular velocity about the z axis.

$$\alpha = \int \dot{\alpha} dt \qquad \alpha_k = \alpha_{k-1} + \dot{\alpha}_k dt$$

- For getting linear positions, we integrate linear acceleration twice.

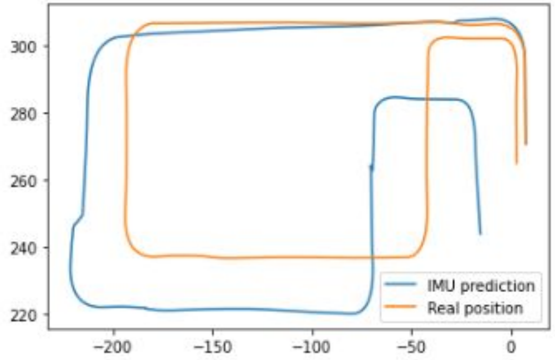
$$x = \int \int \ddot{x} dt \qquad \begin{aligned} \dot{x}_k &= \dot{x}_{k-1} + \ddot{x}_k dt \\ x_k &= x_{k-1} + \dot{x}_k dt \end{aligned}$$

$$y = \int \int \ddot{y} dt \qquad \begin{aligned} \dot{y}_k &= \dot{y}_{k-1} + \ddot{y}_k dt \\ y_k &= y_{k-1} + \dot{y}_k dt \end{aligned}$$

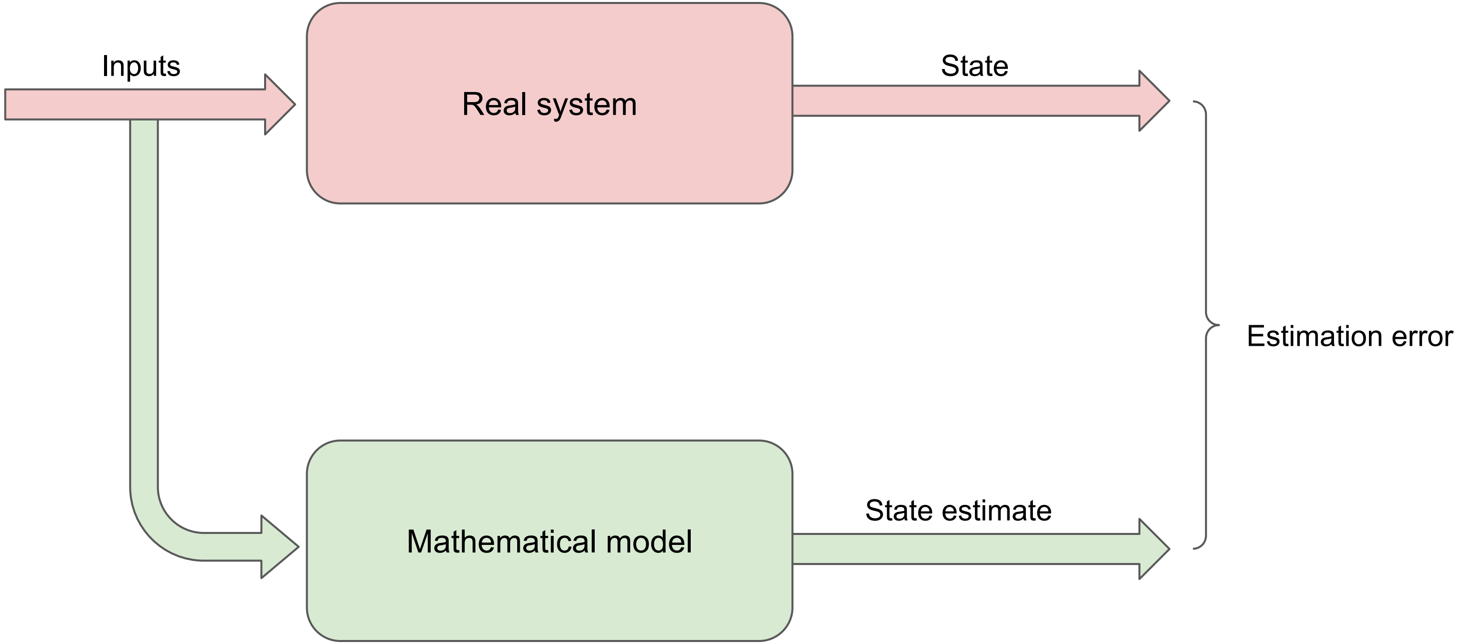
Drift in IMU estimates

$$x = \int \int (\ddot{x} + e)dt \qquad x = \int \int \ddot{x}dt + \frac{et^2}{2}$$

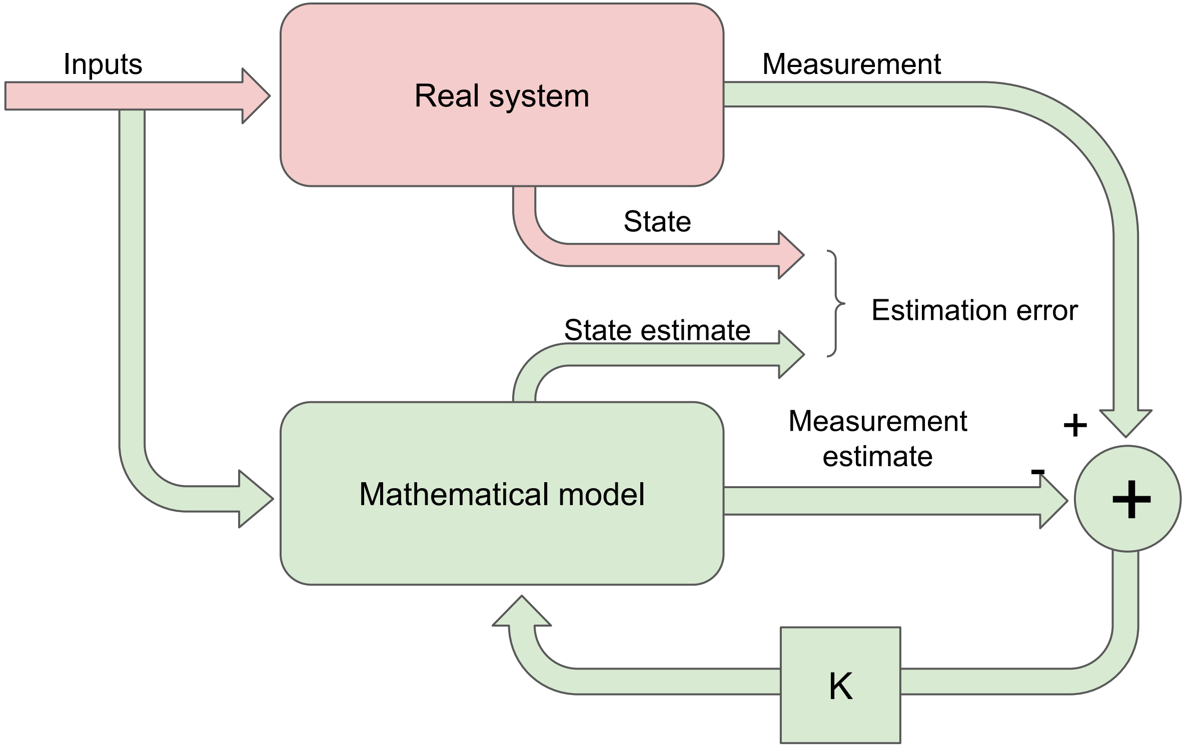
$$y = \int \int (\ddot{y} + e)dt \qquad y = \int \int \ddot{y}dt + \frac{et^2}{2}$$



Simple estimator



With feedback from measurements



What is a kalman filter?

- Invented by Rudolph E.Kalman in 1960
- Kalman filter is a stochastic estimator. Estimate the state x $\in$ R^n of a discrete-time controlled process that is governed by the linear stochastic difference equation

$$x_k = A \cdot x_{k-1} + B \cdot U_k + w_{k-1}$$

with a measurement y $\in$ R^m that is

$$y_k = H \cdot x_k + v_k$$

- The system is assumed to be linear and the measurement and process noise is assumed to be gaussian and independent of each other, with probability distributions

$$p(w) = N(0, Q) \qquad p(v) = N(0, R)$$

Modeling our system

- For vehicles, we assume they are on a 2-D plane. So, we include the global x and y position and velocity and the yaw angle as our state vector.
- We need to transform acceleration from local to global coordinates.

$$
\begin{bmatrix} x_k \\ y_k \\ \alpha_k \\ \dot{x}_k \\ \dot{y}_k \end{bmatrix}
= f\left(\begin{bmatrix} x_{k-1} \\ y_{k-1} \\ \alpha_{k-1} \\ \dot{x}_{k-1} \\ \dot{y}_{k-1} \end{bmatrix},
\begin{bmatrix} \ddot{x}_k \\ \ddot{y}_k \\ \dot{\alpha}_k \end{bmatrix}\right) =
\begin{bmatrix} x_{k-1} + \dot{x}_k dt \\ y_{k-1} + \dot{y}_k dt \\ \alpha_{k-1} + \dot{\alpha}_k dt \\ \dot{x}_{k-1} + (\ddot{x}_k cos(\alpha_{k-1}) - \ddot{y}_k sin(\alpha k - 1))dt \\ \dot{y}_{k-1} + (\ddot{x}_k sin(\alpha_{k-1}) + \ddot{y}_k cos(\alpha k - 1))dt \end{bmatrix}
$$

Next state (x_k) | Previous state (x_{k-1}) | Input vector (u_k)

Extended Kalman filter

- Prediction

$$x_k = f(x_{k-1}, u_k)$$

$$P_k = A_k P_{k-1} A_{k-1}^T + Q_k$$

$$A_k = \left.\frac{\partial f}{\partial x}\right|_x$$

- Measurement

$$v_k = y_k - h(x_k)$$

$$K_k = \frac{P_{k-1} H_k^T}{S_k}$$

$$S_k = H_k P_{k-1} H_k^T + R_k$$

$$H_k = \left.\frac{\partial h}{\partial x}\right|_x$$

$$x_k = x_{k-1} + K_k v_k$$

$$P_k = P_{k-1} - K_k S_k K_k^T$$

Linearization

$$A_k = \frac{\partial f}{\partial x}\bigg|_x = \begin{bmatrix} 1 & 0 & 0 & dt & 0 \\ 0 & 1 & 0 & 0 & dt \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & -dt(\ddot{x}_k sin(\alpha_{k-1}) + \ddot{y}_k cos(\alpha_{k-1})) & 1 & 0 \\ 0 & 0 & dt(\ddot{x}_k cos(\alpha_{k-1}) - \ddot{y}_k sin(\alpha_{k-1})) & 0 & 1 \end{bmatrix}$$

$$B_k = \frac{\partial f}{\partial u}\bigg|_x = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & dt \\ dtcos(\alpha_{k-1}) & -dtsin(\alpha_{k-1}) & 0 \\ dtsin(\alpha_{k-1}) & dtcos(\alpha_{k-1}) & 0 \end{bmatrix}$$

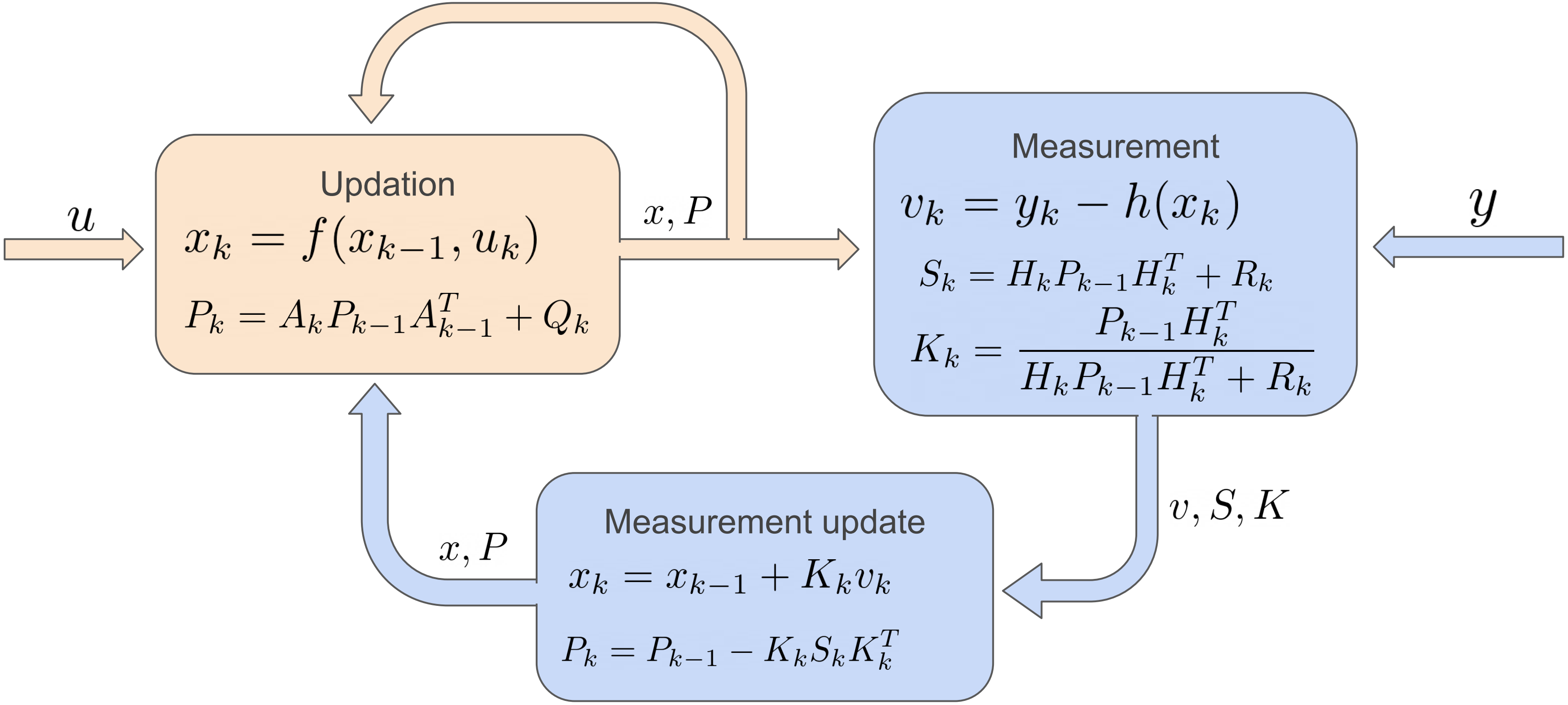
Measurement model

- The measurements are direct. We get global x and y coordinates from GNSS system

$$y_k = \begin{bmatrix} x_{meas} \\ y_{meas} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_k \\ y_k \\ \alpha_k \\ \dot{x}_k \\ \dot{y}_k \end{bmatrix}$$

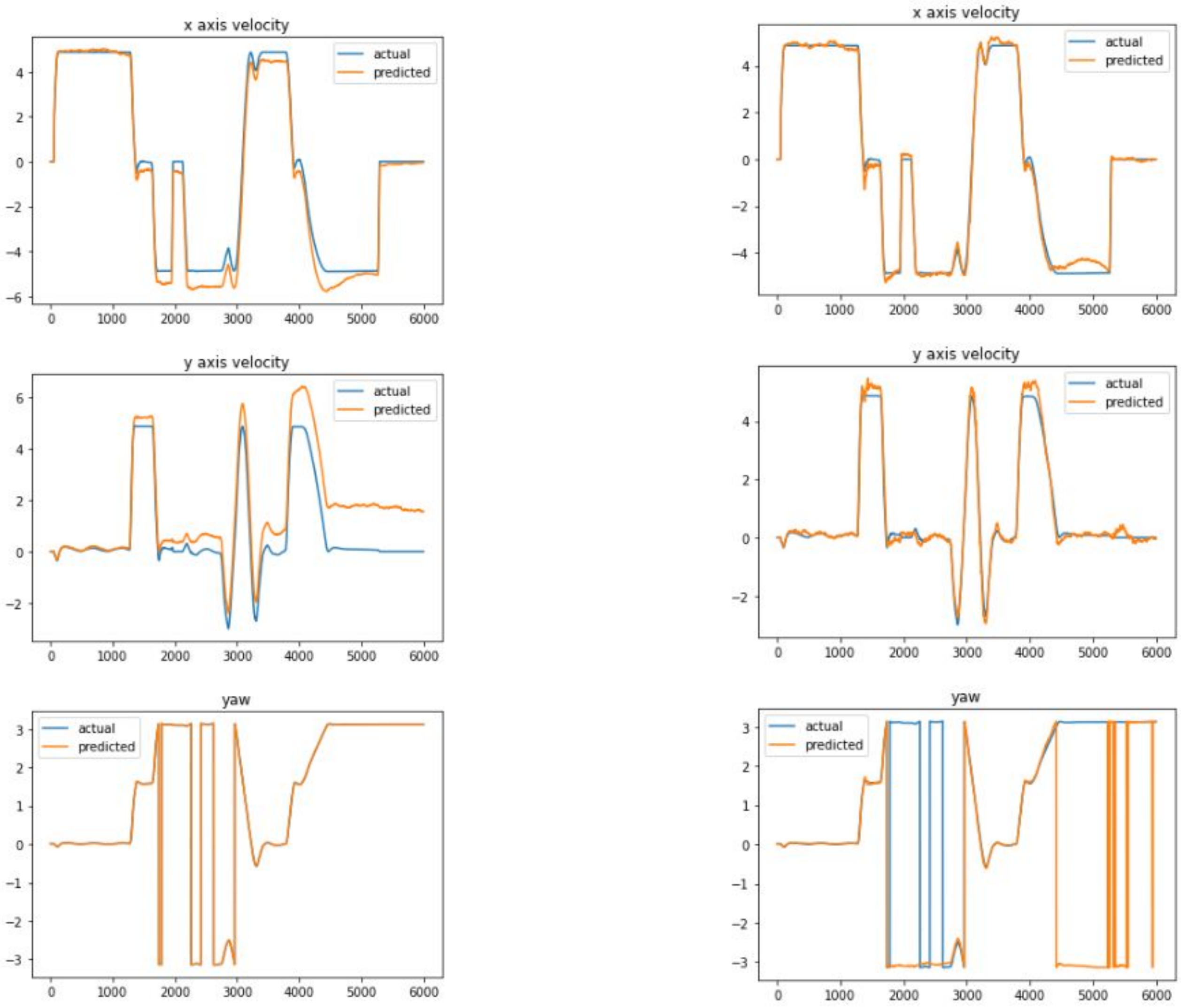
$$H_k = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

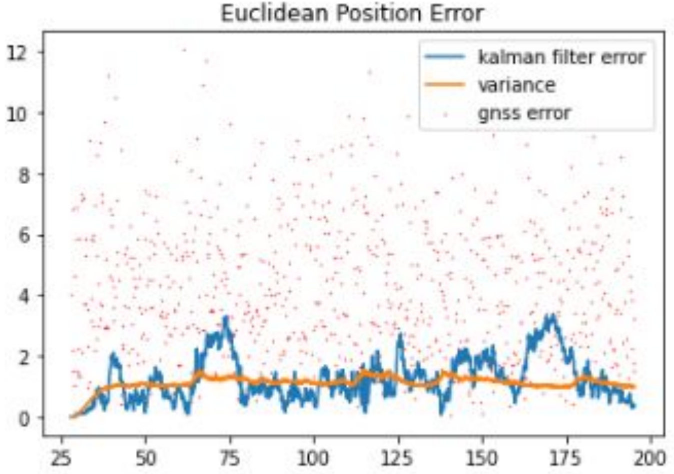
Extended kalman filter flowchart



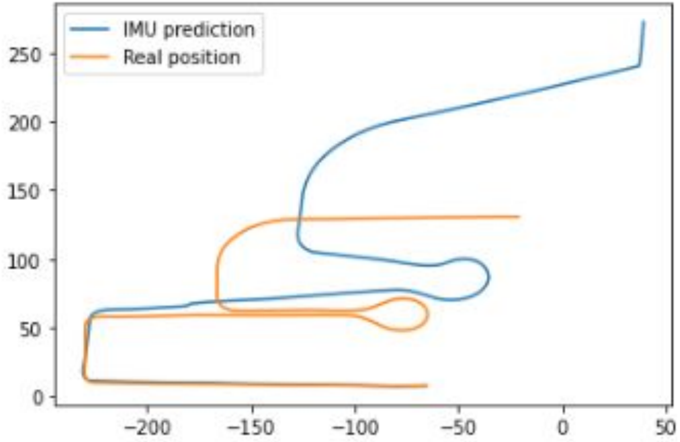
Implementation and simulation

- Implemented in **python**, and jupyter notebooks
- Used **numpy** library for linear algebra
- Used open-source self-driving simulator, **Carla** for simulation
- Code is available at : https://github.com/Ashwin-Rajesh/Kalman_filter_carla
- GNSS system with
  - 3.33 standard deviation
  - 5Hz sampling rate
- IMU with
  - 0.1 standard deviation
  - 60Hz sampling rate

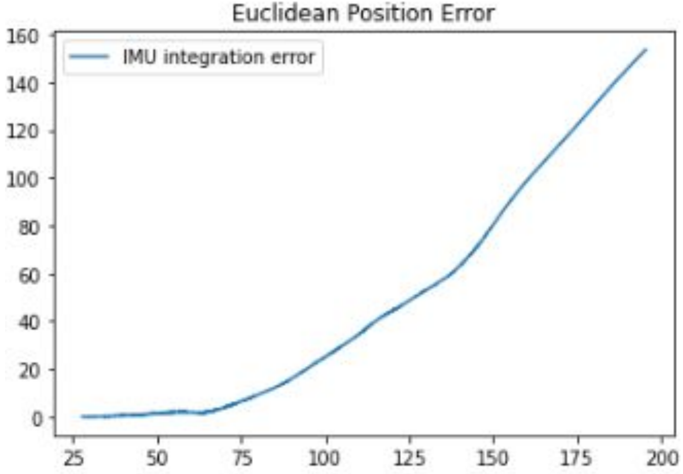

x axis velocity
actual
predicted
y axis velocity
actual
predicted
yaw
actual
predicted
x axis velocity
actual
predicted
y axis velocity
actual
predicted
yaw
actual
predicted



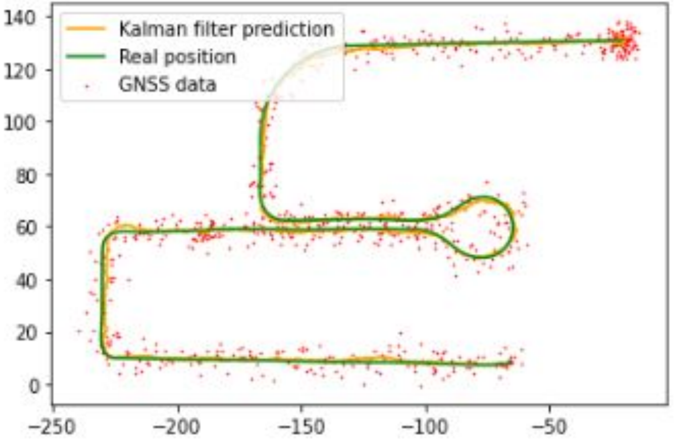
Euclidean Position Error
kalman filter error
variance
gnss error



IMU prediction
Real position



Euclidean Position Error
IMU integration error



Kalman filter prediction
Real position
GNSS data

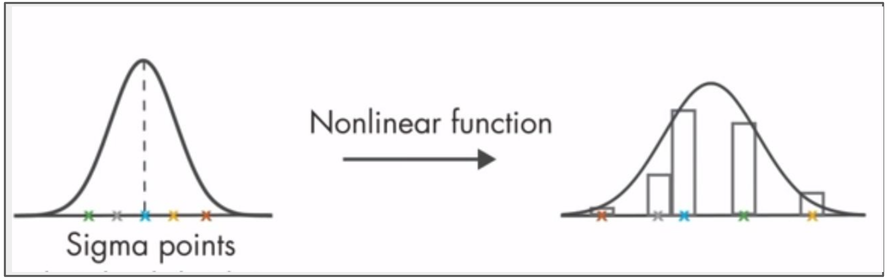
Improvements

- More sensors can be added to improve estimate, including
  - Depth cameras for visual odometry
  - Sensors reading values from the drivetrain and steering systems of the vehicle
  - LIDAR data

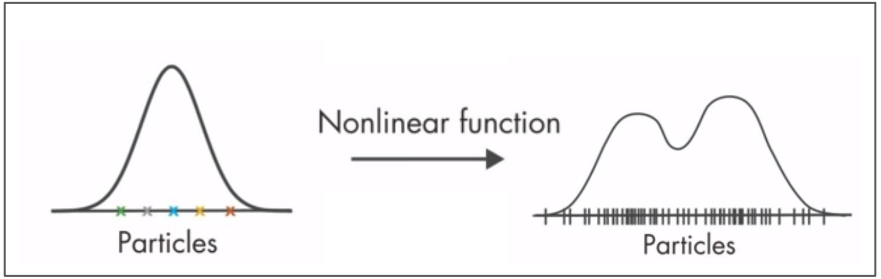
- LIDAR and depth data can be used in particle filters for localization. This is used in SLAM (simultaneous localization and mapping) systems.

Improvements

- Kalman filters are the optimal state estimators in the ideal world (linear systems, gaussian noise). But, for non-linear systems and outliers, they have terrible performance
- **Unscented kalman filters** work better for highly non-linear systems
- **Particle filters** work by generating random "particles" sampled from an initial distribution and passing each through the non-linear system model and getting the distribution for the next step.



Unscented kalman filter [4]

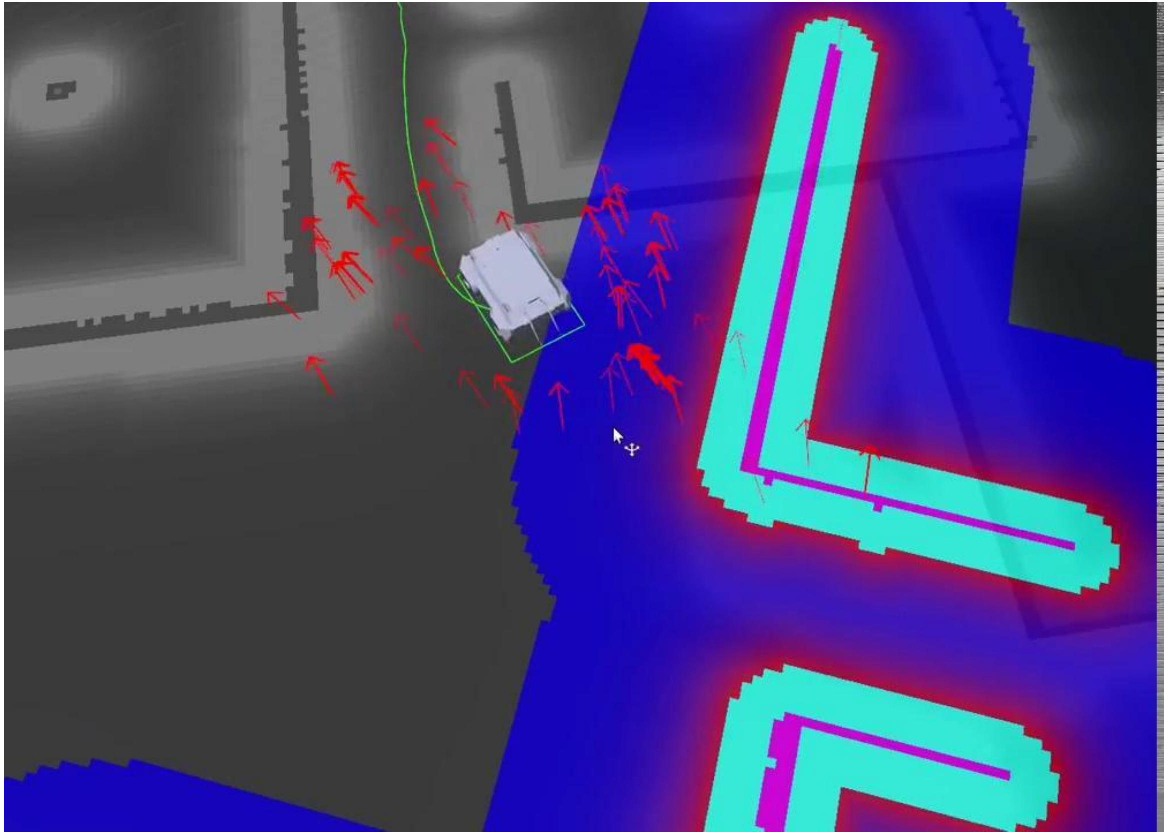


Particle filter [4]

Estimators compared

Filter	Model	Assumed distribution	Computational cost
Kalman filter	Linear	Gaussian	Low
Extended kalman filter	Linearized	Gaussian	Low - Medium
Unscented kalman filter	Non-linear	Gaussian	Medium
Particle filter	Non-linear	Non-gaussian	High

Particle filter localization in ROS - amcl



References and related work

[1] **Adam Werries, John M. Dolan**, Adaptive Kalman Filtering Methods for Low-Cost GPS/INS Localization for Autonomous Vehicles : https://kilthub.cmu.edu/articles/journal_contribution/Adaptive_Kalman_Filtering_Methods_for_Low-Cost_GPS_INS_Localization_for_Autonomous_Vehicles/6551687/files/12032720.pdf

[2] **Isaac Skog**, Sensor Fusion GPS + IMU : https://www.researchgate.net/profile/Mohamed_Mourad_Lafifi/post/how_can_I_get_the_linear_angular_velocity_from_an_IMU_and_GPS_data/attachment/5be6d103cfe4a7645501221a/AS%3A691397002805248%401541853299367/download/Sensor+Fusion+GPS%2BIMU+Localisation2018.pdf

[3] **Mohamed Laaraiedh**, Implementation of kalman filter in python : https://arxiv.org/pdf/1204.0375.pdf

[4] **Mathworks**, Understanding kalman filters : https://in.mathworks.com/videos/series/understanding-kalman-filters.html

[5] **Cihan H. Dagli**, Gravity compensation for IMUs : https://core.ac.uk/download/pdf/82791039.pdf

[6] **Yueming Zhao**, GNSS IMU integration for land vehicle navigation : https://www.diva-portal.org/smash/get/diva2:446078/FULLTEXT01.pdf

[7] **Hang Yan, Qi Shan, Yasutaka Furukawa**, IMU robust double integration : https://arxiv.org/abs/1712.09004

[8] **Analog Devices**, Precision MEMS gyroscopes : https://www.analog.com/en/technical-articles/mems-gyroscope-provides-precision-inertial-sensing.html

[9] **ROS wiki**, amcl package : http://wiki.ros.org/amcl

[10] **Carla simulator** : https://carla.org/

Thank You!