SENSOR FUSION-BASED OBJECT TRACKING FOR SELF-DRIVING CARS USING KALMAN FILTER

Basavaraj Navalgund - 01FE16BEC232

UNDER THE GUIDANCE OF

Dr. Nalini Iyer

Prof. Akash Kulkarni

KLE Technological University

Vidyanagar, BVB Campus, Hubli - 580031

Karnataka, India

CONTENTS

PROBLEM STATEMENT

Environmental perception using LiDAR and RADAR sensors with data combination and inference of data using Kalman filter, Extended Kalman filter and Unscented Kalman filter techniques.

AIM OF THE PROJECT

- Fusion of LiDAR and RADAR sensor data.
- Reduce sensor error.
- Track object’s (vehicle, bicycle, person, etc) position, velocity, turning angle (yaw angle) and turning rate (yaw rate) accurately.

INTRODUCTION

- Object (vehicle, bicycle, person, etc) Tracking is an estimate object’s position, velocity, turning angle (yaw angle) and turning rate (yaw rate). It is a important parameter which needs to be satisfied for a vehicle to be autonomous.
- Here Object Tracking is achieved through fusion of LiDAR and RADAR sensors using Kalman filter, Extended Kalman filter and Unscented Kalman filter.

METHODOLOGY

- Standard Kalman filter
- Kalman filter & Extended Kalman filter
- Unscented Kalman filter

ALGORITHM: 1

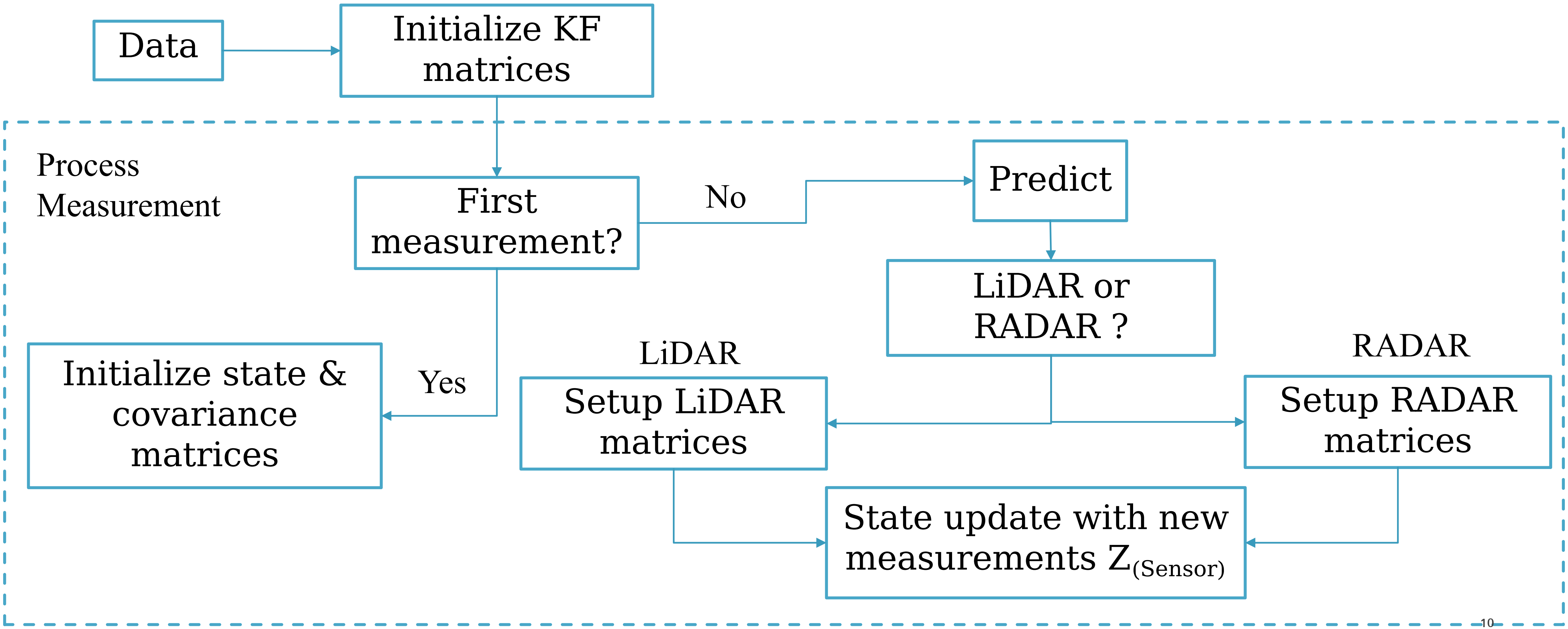
- Standard Kalman Filter
  - It is an algorithm, which uses measurements containing noise & produces estimations, which are more accurate than the sensor measurements. It can handle only linear equation.
  - It involves 3 steps:
    - Initialization
    - Prediction
    - Update

DATA FORMAT

Sensor measurement = meas_
Kalman filter estimated = est_
Ground truth values = gt_

- Input data format:
  - L (for LiDAR) meas_p_x meas_p_y timestamp gt_p_x gt_p_y gt_v_x gt_v_y
  - R (for RADAR) meas_rho meas_phi meas_rho_dot timestamp gt_p_x gt_p_y gt_v_x gt_v_y
- Output data format:
  - est_p_x est_p_y est_v_x est_v_y meas_p_x meas_p_y gt_p_x gt_p_y gt_v_x gt_v_y
- LiDAR gives position (i.e. p_x and p_y).
- RADAR gives radial distance, radial velocity and yaw angle.

BLOCK DIAGRAM



BLOCK DIAGRAM

Initialize state

X_0

P_0

Predict state for LiDAR

$X_{kp} = F \cdot X_{k-1} + u_{k-1}$

$P_{kp} = F \cdot P_{k-1} \cdot F^T + Q_{k-1}$

LiDAR input state

$Y_k = Z_{k(LiDAR)} - H \cdot X_{kp}$

Update state

$K_k = (P_{kp} \cdot H^T) \;/$

$(H \cdot P_{kp} \cdot H^T + R_L)^{-1}$

Outputs of LiDAR Kalman equation

$X_k = X_{kp} - K_k \cdot Y_k$

$P_k = (I - K_k \cdot H)\, P_{kp}$

Predict state for RADAR

$X_{(k+1)p} = F \cdot X_k + u_k$

$P_{(k+1)p} = F \cdot P_k \cdot F^T + Q_k$

RADAR input state

$Y_k = Z_{k(RADAR)} - H \cdot X_{(k+1)p}$

Update state

$K_{k+1} = (P_{(k+1)p} \cdot H^T) \;/$

$(H \cdot P_{(k+1)p} \cdot H^T + R_R)^{-1}$

Outputs of RADAR Kalman equation

$X_{k+1} = X_{(k+1)p} - K_k \cdot Y_{k+1}$

$P_{k+1} = (I - K_{k+1} \cdot H)\, P_{(k+1)p}$

- Abbreviation's
  - **X** represents object state vector
  - **P** represents object state covariance matrix
  - **F** represents state transition matrix
  - **u** represents process noise
  - **Q** represents process noise covariance matrix
  - **Y** represents innovation matrix or cross-correlation matrix
  - **Z** represents sensor measurement matrix
  - **R** represents sensor measurement covariance matrix
  - **K** represents Kalman gain
  - **I** represents identity matrix

$\boxed{\begin{matrix} X_0 \\ P_0 \end{matrix}}$

- Initialize state
  - Used **Constant Speed (CS) motion model**, hence considered position & velocity (**p_x, p_y, v_x, v_y**) in state vector.

State vector

$$\begin{bmatrix} p_x \\ p_y \\ v_x \\ v_y \end{bmatrix}$$

- $\mathbf{X_0}$ represents initialized object state vector.

$$X_0 = \begin{bmatrix} p_x \\ p_y \\ 0 \\ 0 \end{bmatrix}$$

- We initialize:
  - Object's position i.e. p_x, p_y using LiDAR and RADAR sensor.
  - Object's velocity i.e. v_x, v_y to 0.
- $\mathbf{P_0}$ represents the initialized state covariance matrix.
- We initialize diagonal elements of $\mathbf{P_0}$ to unity, because variables of state vector are independent to each other.

$$P_0 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$X_{kp} = F \cdot X_{k\text{-}1} + u_{k\text{-}1}$$

$$P_{kp} = F \cdot P_{k\text{-}1} \cdot F^{T} + Q_{k\text{-}1}$$

- Predict state
  - $\mathbf{X}_{\mathbf{kp}}$ represents predicted object state.
  - **F** represents state transition matrix, it is designed based on state transition equations of linear motion model.
  - $\mathbf{P}_{\mathbf{kp}}$ represents the predicted state covariance matrix.

State transition equations

$$\begin{aligned}
p_x' &= p_x + v_x \Delta t + \tfrac{1}{2} a_x \Delta t^2 \\
p_y' &= p_y + v_y \Delta t + \tfrac{1}{2} a_y \Delta t^2 \\
v_x' &= v_x + a x \Delta t \\
v_y' &= v_y + a_y \Delta t
\end{aligned}$$

$$F = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- $\mathbf{U_{k-1}}$ represents process noise, it gives us information about the car we are tracking is accelerating, decelerating or braking.
- $\mathbf{Q_{k-1}}$ represents process noise covariance matrix, it gives information about the error in the acceleration.

State transition equations

$$
\begin{aligned}
p_x' &= p_x + v_x\Delta t + \tfrac{1}{2}a_x\Delta t^2 \\
p_y' &= p_y + v_y\Delta t + \tfrac{1}{2}a_y\Delta t^2 \\
v_x' &= v_x + a_x\Delta t \\
v_y' &= v_y + a_y\Delta t
\end{aligned}
$$

$$
u_k = \begin{bmatrix} \frac{1}{2}a_x\Delta t^2 \\ \frac{1}{2}a_y\Delta t^2 \\ a_x\Delta t \\ a_y\Delta t \end{bmatrix} = \begin{bmatrix} \frac{1}{2}\Delta t^2 & 0 \\ 0 & \frac{1}{2}\Delta t^2 \\ \Delta t & 0 \\ 0 & \Delta t \end{bmatrix} \begin{bmatrix} a_x \\ a_y \end{bmatrix} = G \cdot a
$$

$$
Q_v = \begin{bmatrix} \sigma^2_{ax} & 0 \\ 0 & \sigma^2_{ay} \end{bmatrix}
$$

$$
Q_k = G \cdot Q_v \cdot G^T
$$

$$Y_k = Z_k - H \cdot X_{kp}$$

- Update state
  - **Y_k** represents innovation matrix.
  - **Z_k** represents LiDAR and RADAR sensor measurements.
  - **H** represents measurement matrix, it is a changing unit of state vector, in order to match sensor measured values.

$$Z_{LiDAR} = \begin{bmatrix} P_x \\ P_y \end{bmatrix}$$

$$H_{LiDAR} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

$$Z_{RADAR} = \begin{bmatrix} P_x = \rho \cdot cos\varphi \\ P_y = \rho \cdot sin\varphi \\ v_x = \dot{\rho} \cdot cos\varphi \\ v_y = \dot{\rho} \cdot sin\varphi \end{bmatrix}$$

$$H_{RADAR} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$K_k = \frac{P_{kp} \cdot H^T}{H \cdot P_{kp} \cdot H^T + R}$$

- Update state
  - **K_k** represents Kalman gain.
  - **R** represents sensor measurement covariance matrix, whose values are given by sensor manufacturers in data sheet.

$$R_{LiDAR} = \begin{bmatrix} \sigma^2_{px} & 0 \\ 0 & \sigma^2_{py} \end{bmatrix}$$

$$R_{RADAR} = \begin{bmatrix} \rho \cdot cos\varphi & 0 & 0 & 0 \\ 0 & \rho \cdot sin\varphi & 0 & 0 \\ 0 & 0 & \dot{\rho} \cdot cos\varphi & 0 \\ 0 & 0 & 0 & \dot{\rho} \cdot sin\varphi \end{bmatrix}$$

Standard Radar

$$Z_{RADAR} = \begin{bmatrix} \rho \\ \varphi \\ \dot{\rho} \end{bmatrix}$$

$$R_{RADAR} = \begin{bmatrix} \sigma^2_{\rho} & 0 & 0 \\ 0 & \sigma^2_{\varphi} & 0 \\ 0 & 0 & \sigma^2_{\dot{\rho}} \end{bmatrix}$$

$$X_k = X_{kp} + K_k \cdot Y_k$$

$$P_k = (I - K_k \cdot H)\, P_{kp}$$

- Update state
  - **I** represents identity matrix.
  - If **R** is larger and $\mathbf{P_{kp}}$ is smaller,then algorithm gives more weightage to predicted values rather than sensor measured values. Vice-Versa.

ALGORITHM: 2

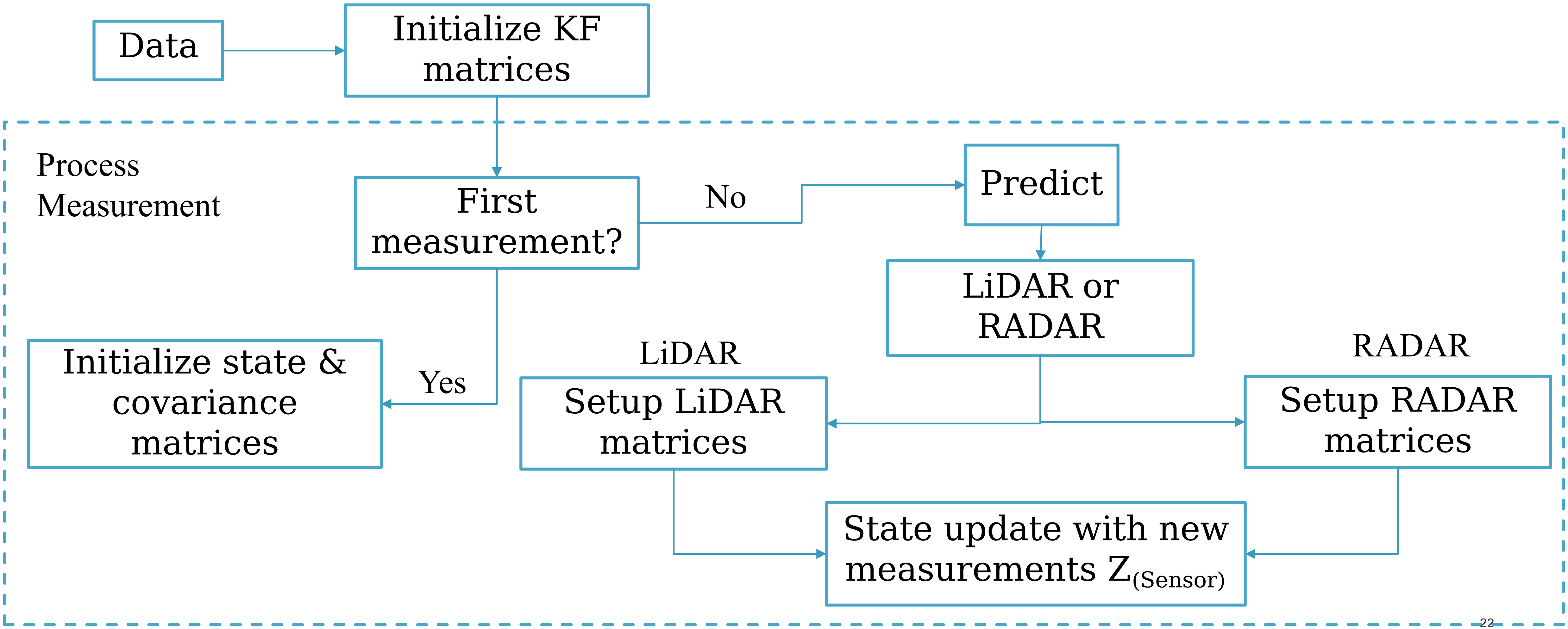
- Kalman Filter & Extended Kalman Filter
  - It is an algorithm, which uses measurements containing noise & produces estimations, which are more accurate than the sensor measurements. Kalman Filter can handle only equations, while Extended Kalman Filter can handle both linear and non-linear equations.
  - It involves 3 steps:
    - Initialization
    - Prediction
    - Update

DATA FORMAT

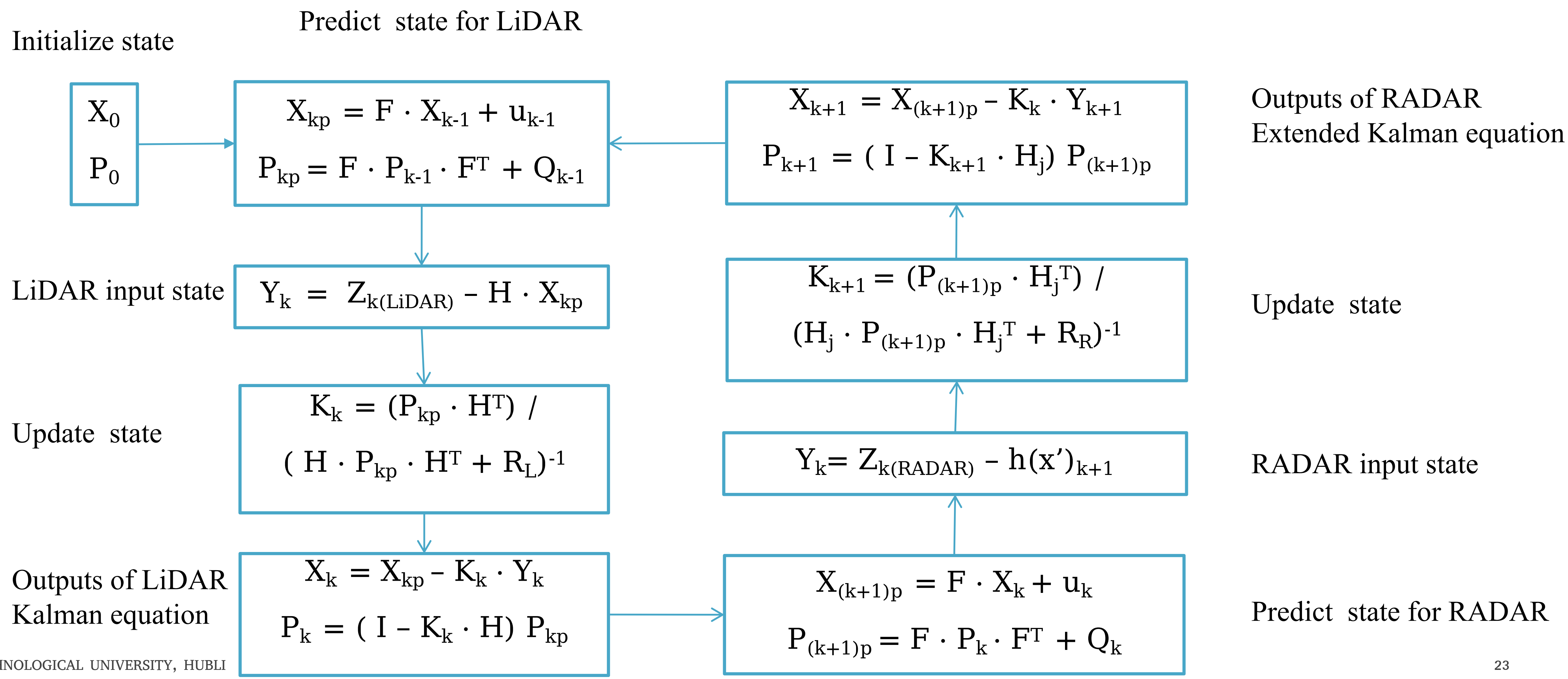
Sensor measurement = meas_
Kalman filter estimated = est_
Ground truth values = gt_

- Input data format:
  - L (for LiDAR) meas_p_x meas_p_y timestamp gt_p_x gt_p_y gt_v_x gt_v_y
  - R (for RADAR) meas_rho meas_phi meas_rho_dot timestamp gt_p_x gt_p_y gt_v_x gt_v_y
- Output data format:
  - est_p_x est_p_y est_v_x est_v_y meas_p_x meas_p_y gt_p_x gt_p_y gt_v_x gt_v_y
- LiDAR gives position (i.e. p_x and p_y).
- RADAR gives radial distance, radial velocity and yaw angle.

BLOCK DIAGRAM



BLOCK DIAGRAM



- Abbreviation's
  - **X** represents object state vector
  - **P** represents object state covariance matrix
  - **F** represents state transition matrix
  - **u** represents process noise
  - **Q** represents process noise covariance matrix
  - **Y** represents innovation matrix or cross-correlation matrix
  - **Z** represents sensor measurement matrix
  - **R** represents sensor measurement covariance matrix
  - **K** represents Kalman gain
  - **I** represents identity matrix
  - **h(x')** represents non-linear function
  - $\mathbf{H_j}$ represents Jacobian matrix

$\boxed{\begin{array}{c} X_0 \\ P_0 \end{array}}$

- Initialize state
  - Used **Constant Speed (CS) motion model**, hence considered position & velocity (**p_x, p_y, v_x, v_y**) in state vector.

State vector

$$\begin{bmatrix} p_x \\ p_y \\ v_x \\ v_y \end{bmatrix}$$

- $\mathbf{X_0}$ represents initialized object state vector.
- We initialize:
  - Object's position i.e. p_x, p_y using LiDAR and RADAR sensor.
  - Object's velocity i.e. v_x, v_y to 0.
- $\mathbf{P_0}$ represents the initialized state covariance matrix.
  - We initialize diagonal elements of $\mathbf{P_0}$ to unity, because variables of state vector are independent to each other.

$$X_0 = \begin{bmatrix} p_x \\ p_y \\ 0 \\ 0 \end{bmatrix}$$

$$P_0 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$X_{kp} = F \cdot X_{k-1} + u_{k-1}$$

$$P_{kp} = F \cdot P_{k-1} \cdot F^{T} + Q_{k-1}$$

- Predict state
  - $\mathbf{X_{kp}}$ represents predicted object state.
  - **F** represents state transition matrix, it is designed based on state transition equations of linear motion model.
  - $\mathbf{P_{kp}}$ represents the predicted state covariance matrix.

State transition equations

$$p_x' = p_x + v_x\Delta t + \tfrac{1}{2}a_x\Delta t^2$$
$$p_y' = p_y + v_y\Delta t + \tfrac{1}{2}a_y\Delta t^2$$
$$v_x' = v_x + ax\Delta t$$
$$v_y' = v_y + a_y\Delta t$$

$$F = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- **U_{k-1}** represents process noise, it gives us information about the car we are tracking is accelerating, decelerating or braking.
- **Q_{k-1}** represents process noise covariance matrix, it gives information about the error in the acceleration.

State transition equations

$$
\begin{aligned}
p_x' &= p_x + v_x\Delta t + \tfrac{1}{2}a_x\Delta t^2 \\
p_y' &= p_y + v_y\Delta t + \tfrac{1}{2}a_y\Delta t^2 \\
v_x' &= v_x + a_x\Delta t \\
v_y' &= v_y + a_y\Delta t
\end{aligned}
$$

$$
u_k = \begin{bmatrix} \frac{1}{2}a_x\Delta t^2 \\ \frac{1}{2}a_y\Delta t^2 \\ a_x\Delta t \\ a_y\Delta t \end{bmatrix} = \begin{bmatrix} \frac{1}{2}\Delta t^2 & 0 \\ 0 & \frac{1}{2}\Delta t^2 \\ \Delta t & 0 \\ 0 & \Delta t \end{bmatrix} \begin{bmatrix} a_x \\ a_y \end{bmatrix} = G \cdot a
$$

$$
Q_v = \begin{bmatrix} \sigma^2_{ax} & 0 \\ 0 & \sigma^2_{ay} \end{bmatrix}
$$

$$
Q_k = G \cdot Q_v \cdot G^T
$$

$$Y_k = Z_{k(LiDAR)} - H \cdot X_{kp}$$

- Update state
  - $\mathbf{Y_k}$ represents innovation matrix.
  - $\mathbf{Z_{k(LiDAR)}}$ represents LiDAR sensor measurements.
  - **H** represents measurement matrix, it is a changing unit of state vector, in order to match sensor measured values.

$$Z_{LiDAR} = \begin{bmatrix} P_x \\ P_y \end{bmatrix}$$

$$H_{LiDAR} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

- Update state

$$K_k = \frac{P_{kp} \,.\, H^T}{H \cdot P_{kp} \cdot H^T + R_L}$$

  - $\mathbf{K_k}$ represents Kalman gain.
  - $\mathbf{P_{kp}}$ represents the state covariance matrix.
  - **H** represents measurement matrix, it is a changing unit of state vector, in order to match sensor measured values.
  - $\mathbf{R_L}$ represents LiDAR sensor covariance matrix, whose values are given by sensor manufacturers in data sheet.

$$K_k = \frac{P_{kp} \cdot H^T}{H \cdot P_{kp} \cdot H^T + R_L}$$

- Update state
  - $\mathbf{K_k}$ represents Kalman gain.
  - $\mathbf{R_L}$ represents LiDAR measurement covariance matrix, whose values are given by sensor manufacturers in data sheet.

$$R_{LiDAR} = \begin{bmatrix} \sigma^2{}_{px} & 0 \\ 0 & \sigma^2{}_{py} \end{bmatrix}$$

$$X_k = X_{kp} - K_k \cdot Y_k$$

$$P_k = (I - K_k \cdot H)\, P_{kp}$$

- Update state
  - **I** represents identity matrix.
  - If **R** is larger and $\mathbf{P_{kp}}$ is smaller, then algorithm gives more weightage to predicted values rather than sensor measured values. Vice-Versa.

- Update state
  - $\mathbf{Y_{k+1}}$ represents innovation matrix.
  - $\mathbf{Z_{k+1(RADAR)}}$ represents RADAR sensor measurements.
  - $\mathbf{h(x')_{k+1}}$ represents non-linear function, it is a contains changing unit of state vector, in order to match sensor measured values.

$$\boxed{Y_{k+1} = Z_{k+1(RADAR)} - h(x')_{k+1}}$$

$$Z_{RADAR} = \begin{bmatrix} \rho \\ \varphi \\ \dot{\rho} \end{bmatrix} \qquad h(x') = \begin{bmatrix} \sqrt{p_x{}^2 + p_y{}^2} \\ arctan\frac{p_y}{p_x} \\ \frac{p_x v_x + p_y v_y}{\sqrt{p_x{}^2 + p_y{}^2}} \end{bmatrix}$$

- Update state
  - $\mathbf{K_{k+1}}$ represents Kalman gain.
  - $\mathbf{H_j}$ represents Jacobian matrix, to approximately linearize the non-linear function h(x'), in this matrix we partially differentiate each variable of state vector with the sensor measurement variables.
  - $\mathbf{R_{RADAR}}$ represents RADAR sensor covariance matrix, whose values are given by sensor manufacturers in data sheet.

$$K_{k+1} = \frac{P_{(k+1)p} \cdot H_j^T}{H_j \cdot P_{(k+1)p} \cdot H_j^T + R_{RADAR}}$$

$$R_{RADAR} = \begin{bmatrix} \sigma^2_{\rho} & 0 & 0 \\ 0 & \sigma^2_{\varphi} & 0 \\ 0 & 0 & \sigma^2_{\dot{\rho}} \end{bmatrix}$$

$$H_j = \begin{bmatrix} \frac{\partial \rho}{\partial p_x} & \frac{\partial \rho}{\partial p_y} & \frac{\partial \rho}{\partial v_x} & \frac{\partial \rho}{\partial v_y} \\ \frac{\partial \varphi}{\partial p_x} & \frac{\partial \varphi}{\partial p_x} & \frac{\partial \varphi}{\partial v_x} & \frac{\partial \varphi}{\partial v_y} \\ \frac{\partial \dot{\rho}}{\partial p_x} & \frac{\partial \dot{\rho}}{\partial p_y} & \frac{\partial \dot{\rho}}{\partial v_x} & \frac{\partial \dot{\rho}}{\partial v_y} \end{bmatrix}$$

$$X_{k+1} = X_{(k+1)p} - K_{k+1} \cdot Y_{k+1}$$

$$P_{k+1} = (I - K_{k+1} \cdot H_j) \, P_{(k+1)p}$$

- Update state
  - **I** represents identity matrix.
  - If **R** is larger and $\mathbf{P_{(k+1)p}}$ is smaller, then algorithm gives more weightage to predicted values rather than sensor measured values. Vice-Versa.

ALGORITHM: 3

- Unscented Kalman Filter
  - It is an algorithm, which uses measurements containing noise & produces estimations, which are more accurate than the sensor measurements. It can handle both linear and non-linear equations.
  - It involves 3 steps:
    - Initialization
    - Prediction
    - Update

DATA FORMAT

Sensor measurement = meas_
Kalman filter estimated = est_
Ground truth values = gt_

- Input data format:
  - L (for LiDAR) meas_p_x meas_p_y Timestamp gt_p_x gt_p_y gt_v_x gt_v_y gt_yaw gt_yawrate
  - R (for RADAR) meas_rho meas_phi meas_rho_dot Timestamp gt_p_x gt_p_y gt_v_x gt_v_y gt_yaw gt_yawrate
- Output data format:
  - Timestamp est_p_x est_p_y est_v est_yaw est_yawrate sensor_type NIS meas_p_x meas_p_y gt_p_x gt_p_y gt_v_x gt_v_y
- LiDAR gives position (i.e. p_x and p_y).
- RADAR gives radial distance, radial velocity and yaw angle.

BLOCK DIAGRAM FOR PREDICTION STEP

Generate sigma points

$$X_{k|k} = \quad x_{k|k} \cdot \quad x_{k|k} + \sqrt{(\lambda + n_x)\ P_{k|k}} \quad \cdot \quad x_{k|k} - \sqrt{(\lambda + n_x)\ P_{k|k}}$$

Rules for sigma point matrix

Predict sigma points

$$X_{k+1|k} = f(x_k\ ,\ v_k)$$

Process model

Predict state mean and covariance

$$x_{k+1|k} = \sum_{i=0}^{2n_a} w_i \cdot X_{k+1|k,i}$$

Predicted state mean

$$P_{k+1|k} = \sum_{i=0}^{2n_a} w_i \cdot (X_{k+1|k,i} - x_{k+1|k}) \cdot (X_{k+1|k,i} - x_{k+1|k})^T$$

Predicted state covariance

BLOCK DIAGRAM FOR UPDATE STEP

Predict measurement

$$\mathrm{Z}_{k+1|k} = h(x_{k+1})$$

Measurement model

$$z_{k+1|k} = \sum_{i=0}^{2n_a} w_i \cdot \mathrm{Z}_{k+1|k,i}$$

Predicted measurement mean

$$S_{k+1|k} = \sum_{i=0}^{2n_a} w_i \cdot (\mathrm{Z}_{k+1|k,i} - z_{k+1|k}) \cdot (\mathrm{Z}_{k+1|k,i} - z_{k+1|k})^T + R$$

Predicted measurement covariance

BLOCK DIAGRAM FOR UPDATE STEP CONTINUED….

Update state

$$T_{k+1|k} = \sum_{i=0}^{2n_a} w_i \,(X_{k+1|k,i} - x_{k+1|k}) \cdot (Z_{k+1|k,i} - z_{k+1|k})^T$$

Cross-correlation between sigma points in state space and measurement space

$$K_{k+1|k} = T_{k+1|k} \cdot S^{-1}{}_{k+1|k}$$

Kalman gain

$$X_{k+1|k+1} = x_{k+1|k} + K_{k+1|k} \cdot (Z_{k+1(\text{Sensor})} - z_{k+1|k})$$

State update

$$P_{k+1|k+1} = P_{k+1|k} - K_{k+1|k} \cdot S_{k+1|k} \cdot K^T{}_{k+1|k}$$

Covariance matrix update

- Abbreviation’s
  - **X** represents matrix with predicted sigma points
  - **x** represents object state mean vector
  - **P** represents object state covariance matrix
  - **λ** represents scaling factor
  - $\mathbf{n_a}$ represents augment state vector dimension.
  - **v** represents process noise.
  - **w** represents weights
  - $\mathbf{h(x)}$ represents non-linear function
  - **Z** represents matrix with measured sigma points
  - **z** represents measurement mean vector
  - **S** represents measurement covariance matrix
  - **T** represents cross-correlation matrix.
  - $\mathbf{Z_{(Sensor)}}$ represents LiDAR and RADAR sensor measurement vector
  - **R** represents sensor covariance matrix
  - **K** represents Kalman gain

$$X_{k|k} = \left[\; x_{k|k} \quad x_{k|k} + \sqrt{(\lambda + n_x)\, P_{k|k}} \quad x_{k|k} - \sqrt{(\lambda + n_x)\, P_{k|k}} \;\right]$$

Rules for sigma point matrix

- Generate sigma points
  - Used **Constant Turn Rate Speed (CTRS)** motion model, hence considered position & velocity, yaw angle and yaw rate ($\mathbf{p_x, p_y, v, \psi, \dot{\psi}}$).
  - $\mathbf{X_{k|k}}$ represents matrix with sigma points at time k.

State vector

$$\begin{bmatrix} p_x \\ p_y \\ v \\ \psi \\ \dot{\psi} \end{bmatrix}$$

- $\mathbf{x}_{k|k}$ represents object state mean.
- We initialize:
  - Object's position i.e. p_x, p_y using LiDAR and RADAR sensor.
  - Object's velocity, yaw angle and yaw rate i.e. v, ψ, $\dot{\psi}$ to 0.
- $\boldsymbol{\lambda}$ represents scaling factor, it decided how far we need to spread sigma points.
- $\mathbf{n}_x$ represents state vector dimension.
- $\mathbf{P}_{k|k}$ represents state covariance matrix.
  - We initialize diagonal elements of itto unity, because variables of state vector are independent to each other.

$$x_{k|k} = \begin{bmatrix} p_x \\ p_y \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

$$\lambda = 3 - n_x$$

Where, $n_x = 5$

$$P_{k|k} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$X_{a,k|k} = \left[x_{a,k|k} \quad x_{a,k|k} + \sqrt{(\lambda + n_x)\, P_{a,k|k}} \quad x_{a,k|k} - \sqrt{(\lambda + n_x)\, P_{a,k|k}} \right]$$

Generate Augment sigma points

- Augment sigma points
  - $\boldsymbol{\nu_{a,\,k}}$ represents longitudinal acceleration noise.
  - $\boldsymbol{\nu_{\ddot{\psi},\,k}}$ represents yaw angle acceleration noise

Augment State

$$\begin{bmatrix} p_x \\ p_y \\ v \\ \psi \\ \dot{\psi} \\ \nu_{a,\,k} \\ \nu_{\ddot{\psi},\,k} \end{bmatrix}$$

$$Q = \begin{bmatrix} \sigma^2{}_a & 0 \\ 0 & \sigma^2{}_\varphi \end{bmatrix} \qquad P_{a,k|k} = \begin{bmatrix} P_{k|k} & 0 \\ 0 & Q \end{bmatrix}$$

$$X_{k+1|k} = f(x_k , v_k)$$

Process model

$$X_{k+1|k} = X_{k|k} + \int_{t_k}^{t_{k+1}} g(x)\, dt + v_k$$

- Predict sigma points
  - $\mathbf{x_k}$ represents object state mean vector.

$$x_k = \begin{bmatrix} p_x \\ p_y \\ v \\ \psi \\ \dot{\psi} \end{bmatrix}$$

- $\mathbf{v_k}$ represents process noise.
- After passing each generated sigma points through process model, hence we get matrix with sigma points at time k+1 i.e. $X_{k+1|k}$ (i.e. predicted set of sigma points in x-y plane).

Differential equation

$$\dot{x} = g(x)$$

$$g(x) = \begin{bmatrix} \dot{P}_x \\ \dot{P}_y \\ \dot{v} \\ \dot{\psi} \\ \ddot{\psi} \end{bmatrix} = \begin{bmatrix} \cos\psi \cdot v \\ \sin\psi \cdot v \\ 0 \\ \dot{\psi} \\ 0 \end{bmatrix}$$

$$v_k = \begin{bmatrix} \frac{1}{2} \cdot \Delta t^2 \cdot \cos\psi_k \cdot v_{a,k} \\ \frac{1}{2} \cdot \Delta t^2 \cdot \sin\psi_k \cdot v_{a,k} \\ \Delta t \cdot v_{a,k} \\ \Delta t^2 \cdot v_{\ddot{\psi},k} \\ \Delta t \cdot v_{\ddot{\psi},k} \end{bmatrix}$$

$$\mathrm{x}_{k+1|k} = \sum_{i=0}^{2n_a} w_i \cdot \mathrm{X}_{k+1|k,i}$$

Predicted state mean

$$\mathrm{P}_{k+1|k} = \sum_{i=0}^{2n_a} w_i \cdot (\mathrm{X}_{k+1|k,i} - \mathrm{x}_{k+1|k}) \cdot (\mathrm{X}_{k+1|k,i} - \mathrm{x}_{k+1|k})^T$$

Predicted state covariance

- Predict mean & covariance.
  - $\mathbf{X}_{k+1|k,i}$ represents matrix with predicted sigma points at time k+1.
  - $\mathbf{w}_i$ represents weights.
    - Where, $w_i = \frac{\lambda}{\lambda + n_x}$, $i = 0,1,2, , , , , n_x$
  - $\mathbf{n}_a$ represents augment state vector dimension.

$$Z_{k+1|k} = h(x_{k+1})$$

Measurement model

- Predict measurement
  - $\mathbf{h}(\mathbf{x}_{k+1})$ represents non-linear function, it is a contains changing unit of state vector, in order to match sensor measured values.

Function for LiDAR

$$h(x_{k+1}) = \begin{bmatrix} p_x \\ p_y \end{bmatrix}$$

Non-linear function for Radar

$$h(x_{k+1}) = \begin{bmatrix} \sqrt{{p_x}^2 + {p_y}^2} \\ arctan\frac{p_y}{p_x} \\ \frac{p_x v_x + p_y v_y}{\sqrt{{p_x}^2 + {p_y}^2}} \end{bmatrix}$$

- Now, we pass earlier generated each sigma points through measurement model, hence we get matrix with sigma points at time k+1 i.e. $Z_{k+1|k}$ (i.e. measured set of sigma points in x-y plane).

$$z_{k+1|k} = \sum_{i=0}^{2n_a} w_i \cdot Z_{k+1|k,i}$$

Predicted measurement mean

- Predict measurement
  - $\mathbf{Z_{k+1|k,i}}$ represents matrix with measured sigma points at time k+1.
  - $\mathbf{w_i}$ represents weights.
    - Where, $w_i = \dfrac{\lambda}{\lambda + n_x}$, $i = 0,1,2,,,,,\ n_x$
  - $\mathbf{n_a}$ represents augment state vector dimension.

$$S_{k+1|k} = \sum_{i=0}^{2n_a} w_i \cdot (Z_{k+1|k,i} - z_{k+1|k}) \cdot (Z_{k+1|k,i} - z_{k+1|k})^T + R_{sensor}$$

Predicted measurement covariance

- Predict measurement
  - $\mathbf{Z_{k+1|k,i}}$ represents matrix with measured sigma points at time k+1.
  - $\mathbf{z_{k+1|k}}$ represents predicted measurement mean.
  - $\mathbf{R_{sensor}}$ represents sensor covariance matrix, whose values are given by sensor manufacturers in data sheet.

$$R_{LiDAR} = \begin{bmatrix} \sigma^2{}_{px} & 0 \\ 0 & \sigma^2{}_{py} \end{bmatrix} \qquad R_{RADAR} = \begin{bmatrix} \sigma^2{}_{\rho} & 0 & 0 \\ 0 & \sigma^2{}_{\varphi} & 0 \\ 0 & 0 & \sigma^2{}_{\dot{\rho}} \end{bmatrix}$$

$$T_{k+1|k} = \sum_{i=0}^{2n_a} w_i \, (X_{k+1|k,i} - x_{k+1|k}) \cdot (Z_{k+1|k,i} - z_{k+1|k})^T$$

Cross-correlation between sigma points
in state space and measurement space

- Update state
  - $\mathbf{X_{k+1|k,i}}$ represents matrix with predicted sigma points at time k+1.
  - $\mathbf{x_{k+1|k}}$ represents predicted state mean.
  - $\mathbf{Z_{k+1|k,i}}$ represents matrix with measured sigma points at time k+1.
  - $\mathbf{z_{k+1|k}}$ represents predicted measurement mean.

$$K_{k+1|k} = T_{k+1|k} \cdot S^{-1}{}_{k+1|k}$$

Kalman gain

$$X_{k+1|k+1} = x_{k+1|k} + K_{k+1|k} \cdot (Z_{k+1(Sensor)} - z_{k+1|k})$$

State update

- Update state
  - $\mathbf{T}_{\mathbf{k+1|k}}$ represents cross-correlation.
  - $\mathbf{x}_{\mathbf{k+1|k}}$ represents predicted state mean.
  - $\mathbf{S}_{\mathbf{k+1}}$ represents predicted measurement covariance.
  - $\mathbf{Z}_{\mathbf{k+1(Sensor)}}$ represents LiDAR and RADAR sensor measurements.
  - $\mathbf{z}_{\mathbf{k+1|k}}$ represents predicted measurement mean.

$$Z_{LiDAR} = \begin{bmatrix} p_x \\ p_y \end{bmatrix} \qquad Z_{RADAR} = \begin{bmatrix} \rho \\ \varphi \\ \dot{\rho} \end{bmatrix}$$

$$P_{k+1|k+1} = P_{k+1|k} - K_{k+1|k} \cdot S_{k+1|k} \cdot K^{T}_{k+1|k}$$

Covariance matrix update

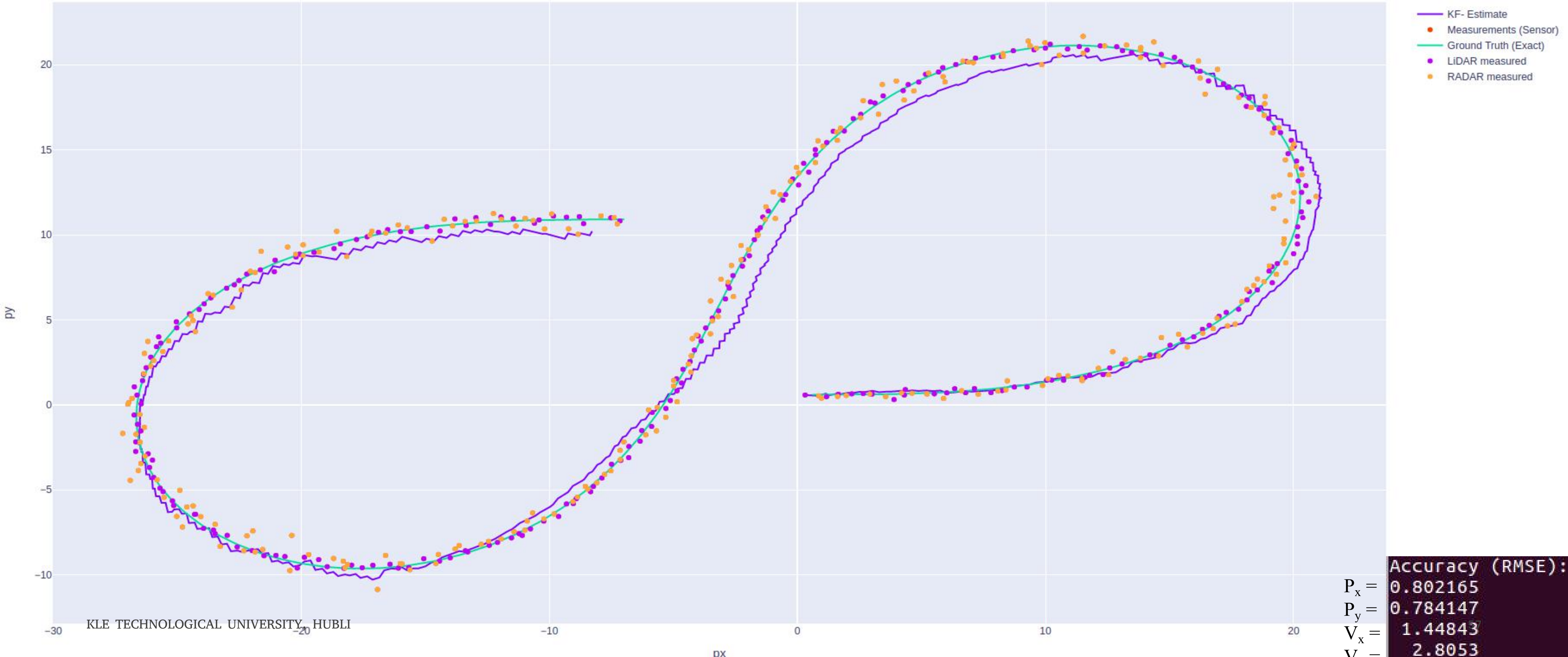
- Update state
  - $\mathbf{P}_{k+1|k}$ represents predicted state covariance.
  - $\mathbf{S}_{k+1}$ represents predicted measurement covariance.
  - $\mathbf{K}_{k+1|k}$ represents Kalman gain.

RESULT'S

PERFORMANCE MEASUREMENT OF FILTER

- Root Mean Square Error (RMSE)
  - RMSE = $\sqrt{\sum_{i=1}^{n} \frac{(Y_{i(predicted)} - Y_{i(ground\ truth)})^2}{n}}$
    - Where, n = Number of observations

KF IMPLEMANTATION FOR ROAD SENARIO:1



Accuracy (RMSE):

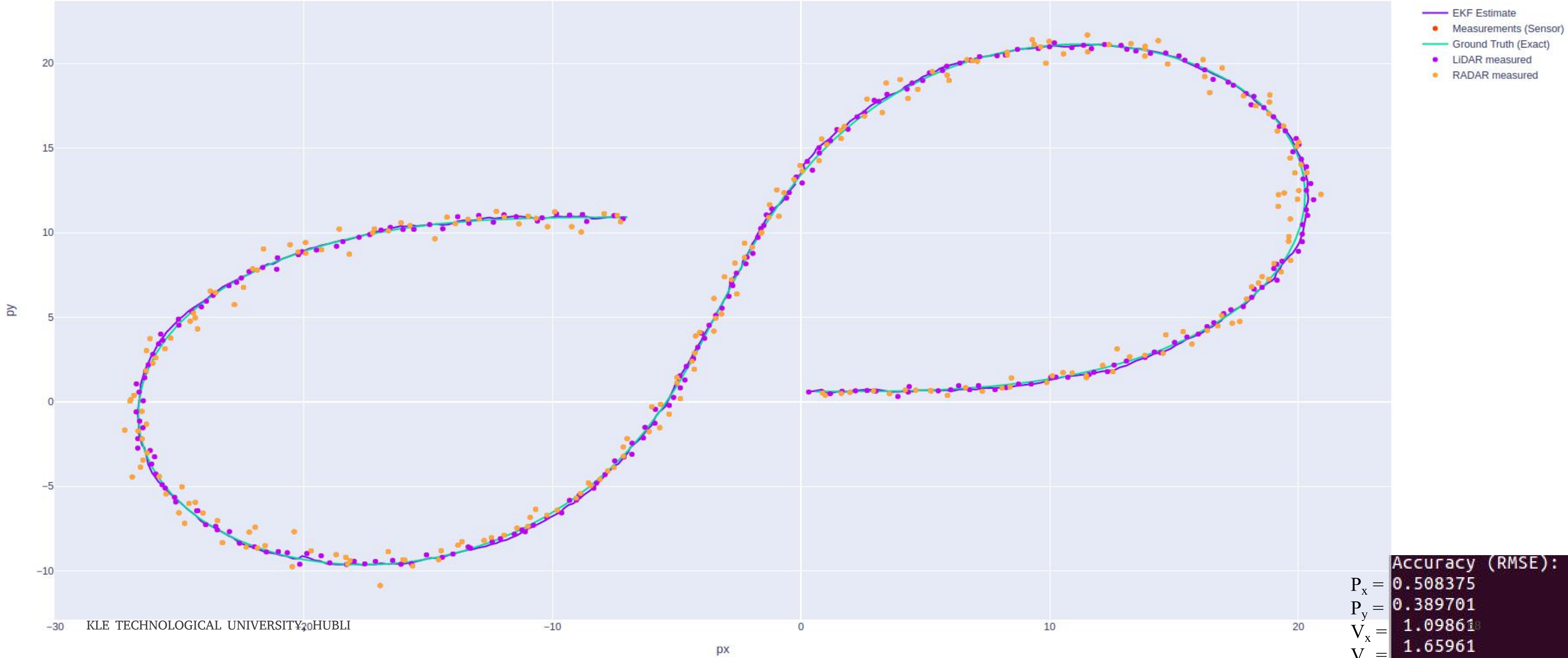
$P_x =$ 0.802165

$P_y =$ 0.784147

$V_x =$ 1.44843

$V_y =$ 2.8053

EKF IMPLEMANTATION FOR ROAD SENARIO:1



Accuracy (RMSE):

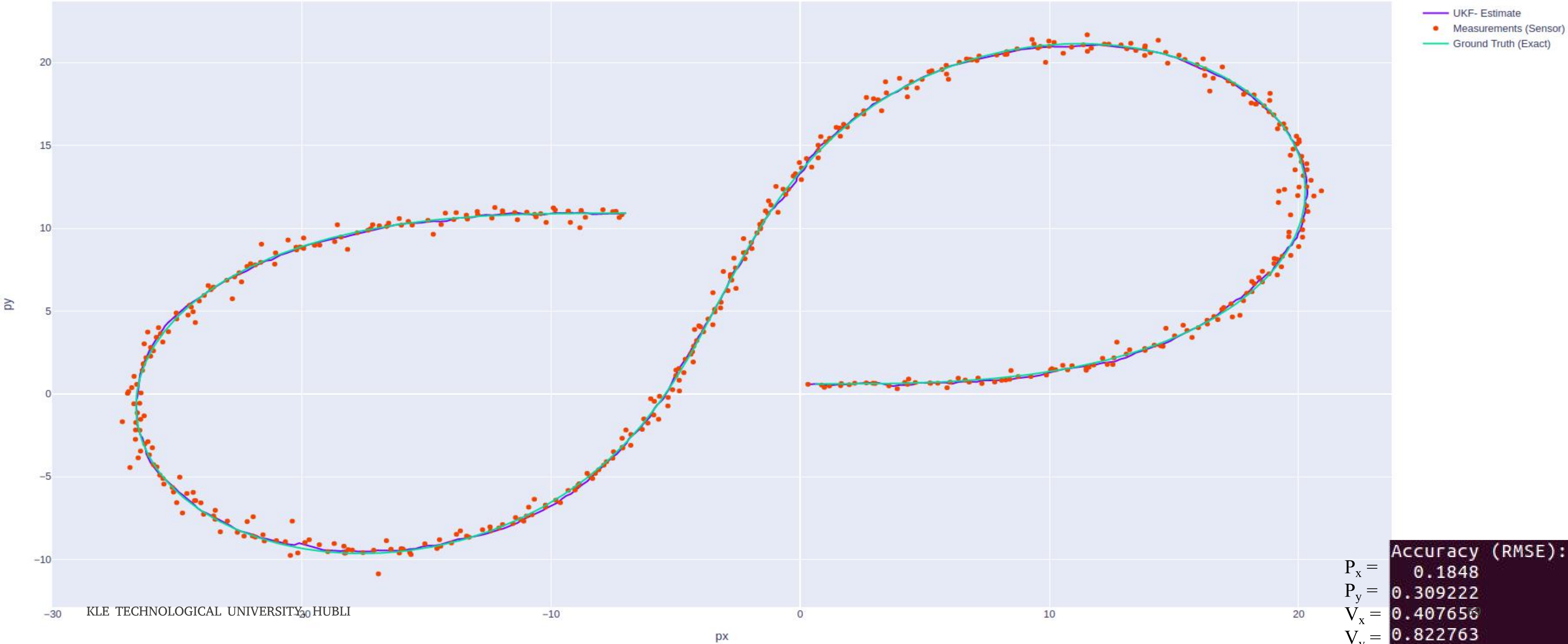
P_x = 0.508375

P_y = 0.389701

V_x = 1.09861

V_y = 1.65961

UKF IMPLEMANTATION FOR ROAD SENARIO:1



Accuracy (RMSE):

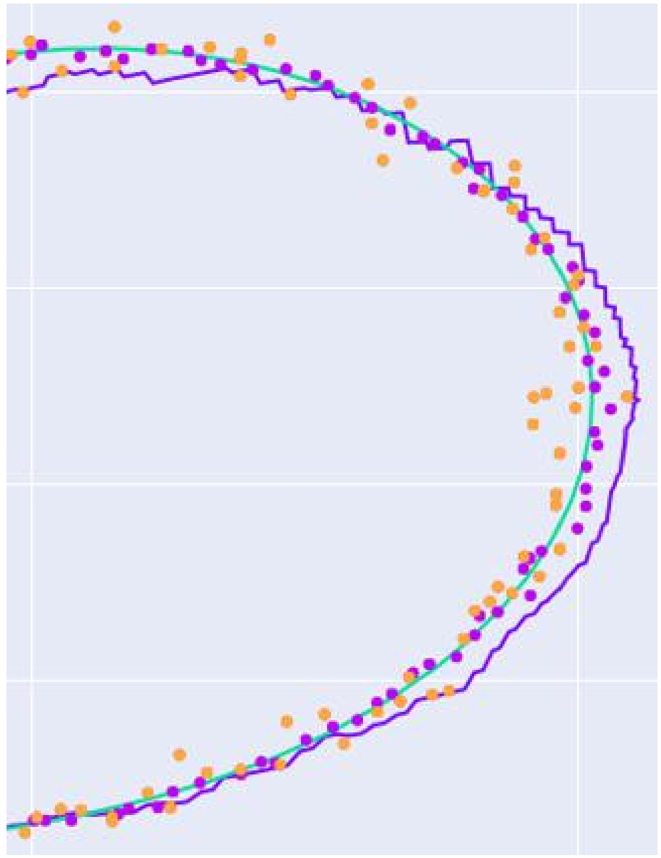
$P_x =$ 0.1848

$P_y =$ 0.309222

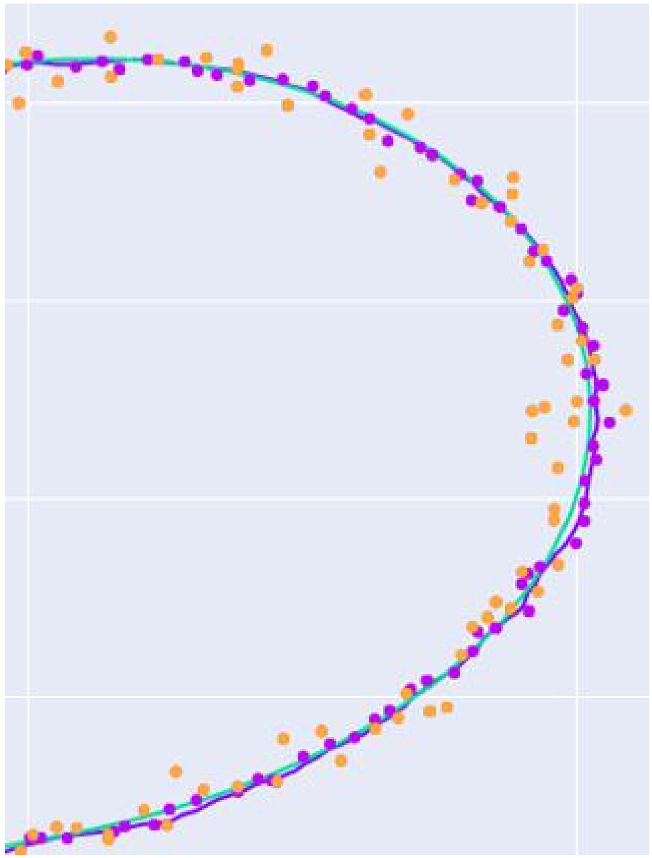
$V_x =$ 0.407656

$V_y =$ 0.822763

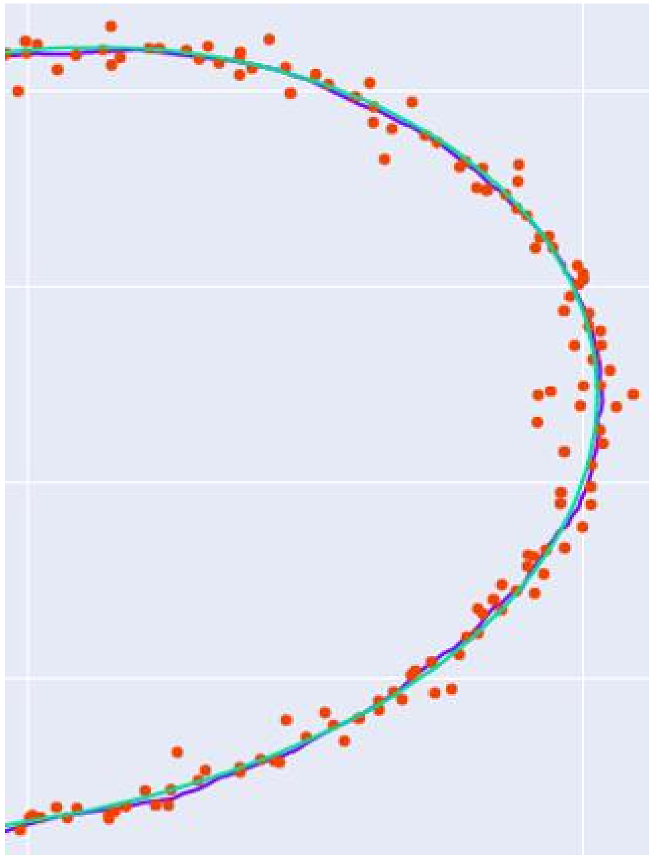
KF, EKF & UKF IMPLEMANTATION FOR ROAD SENARIO:1



KF Plot

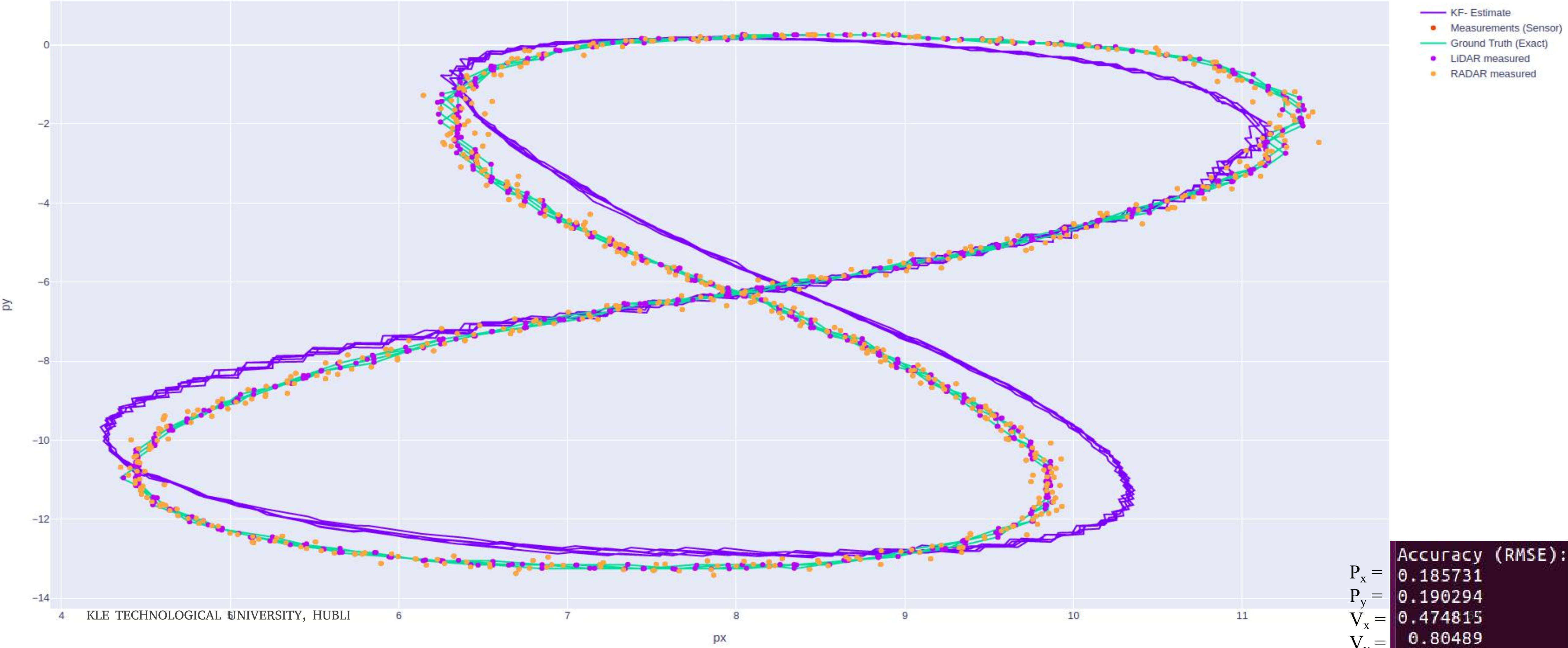


EKF Plot



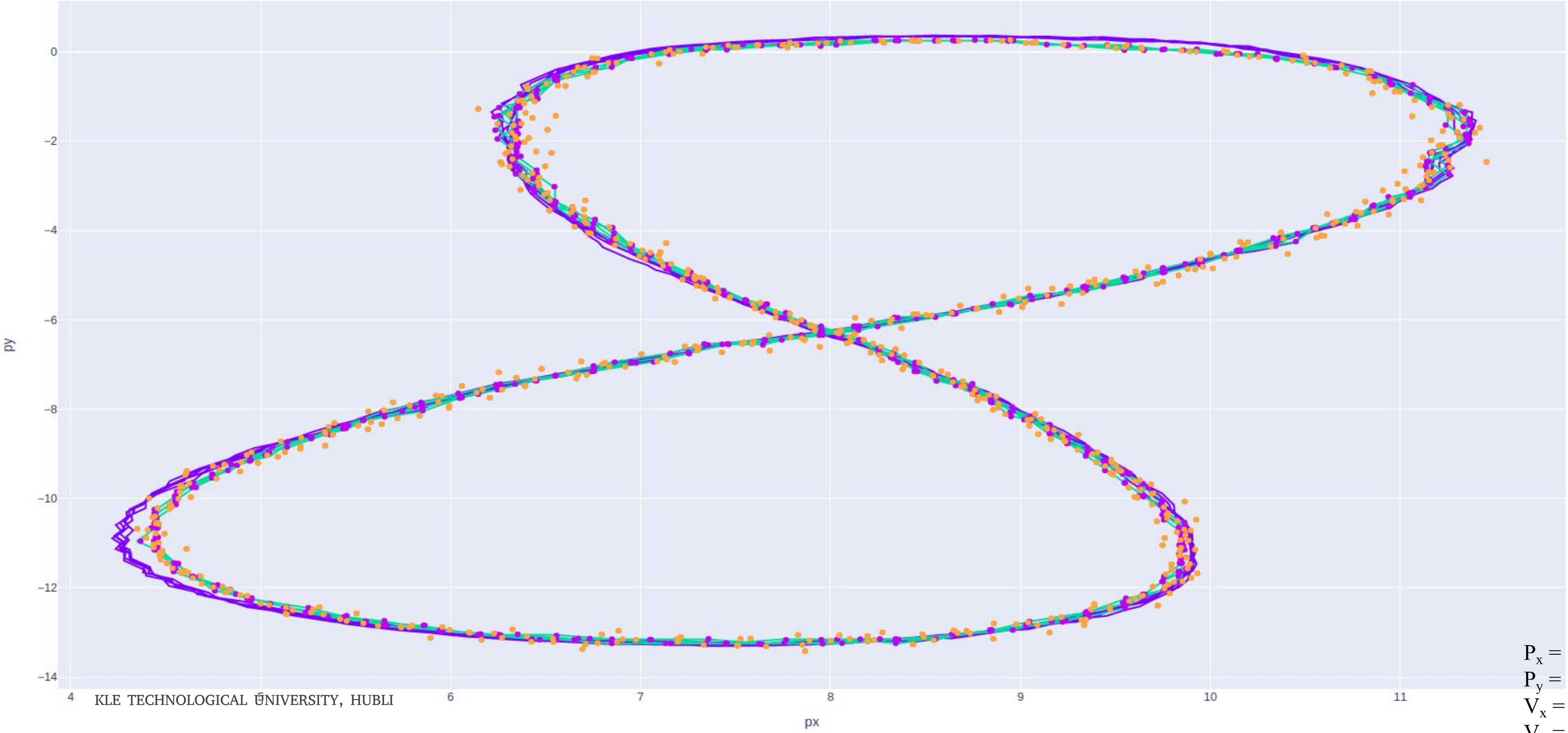
UKF Plot

KF IMPLEMANTATION FOR ROAD SENARIO:2

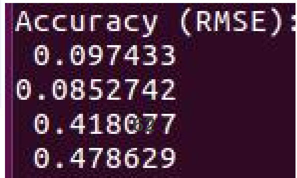


	Accuracy (RMSE):
P_x =	0.185731
P_y =	0.190294
V_x =	0.474815
V_y =	0.80489

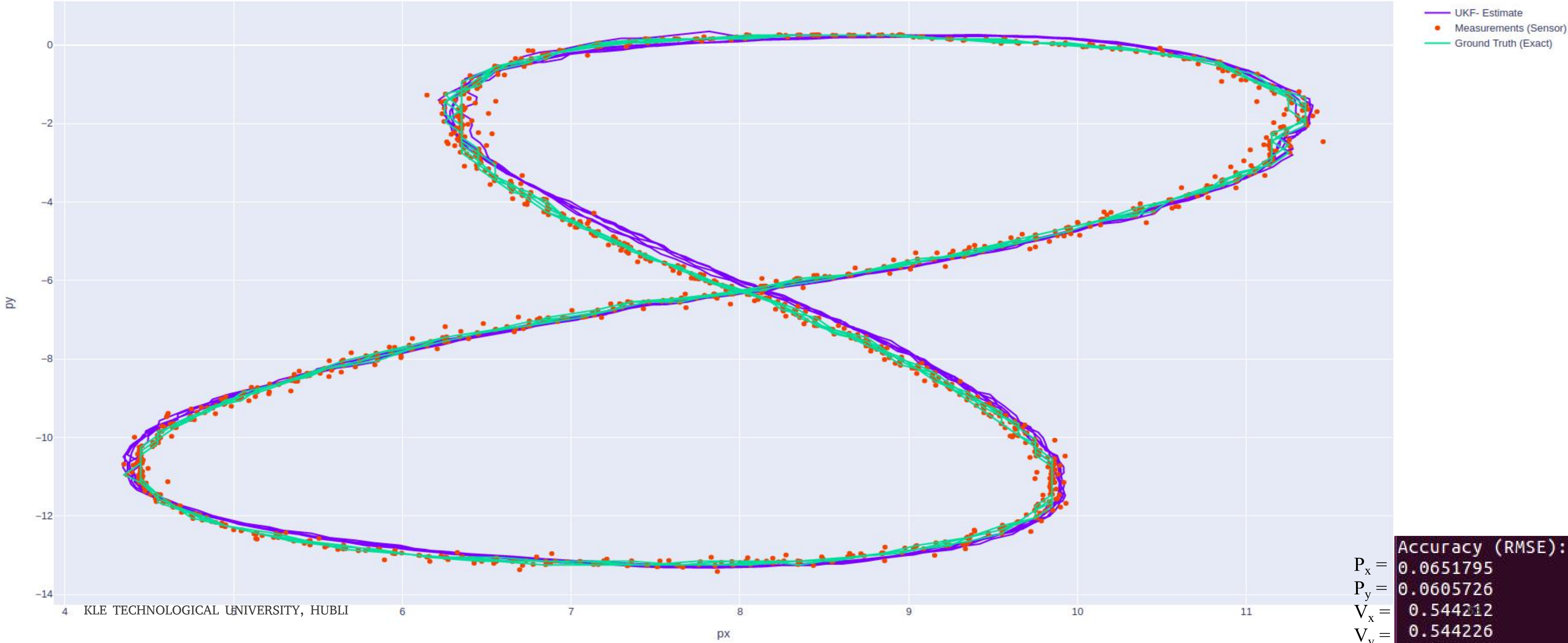
EKF IMPLEMANTATION FOR ROAD SENARIO:2



$P_x =$
$P_y =$
$V_x =$



UKF IMPLEMANTATION FOR ROAD SENARIO:2



Accuracy (RMSE):

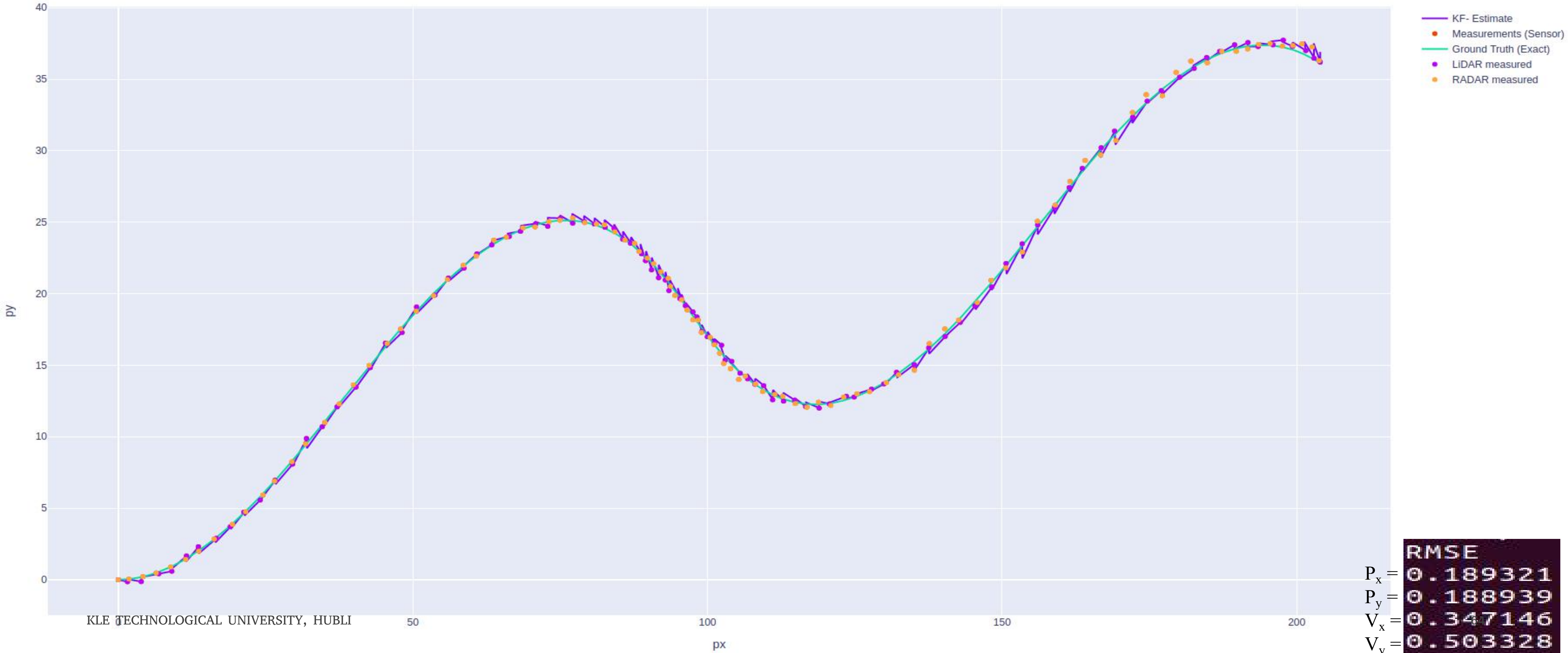
$P_x =$ 0.0651795

$P_y =$ 0.0605726

$V_x =$ 0.544212

$V_y =$ 0.544226

KF IMPLEMANTATION FOR ROAD SENARIO:3



RMSE

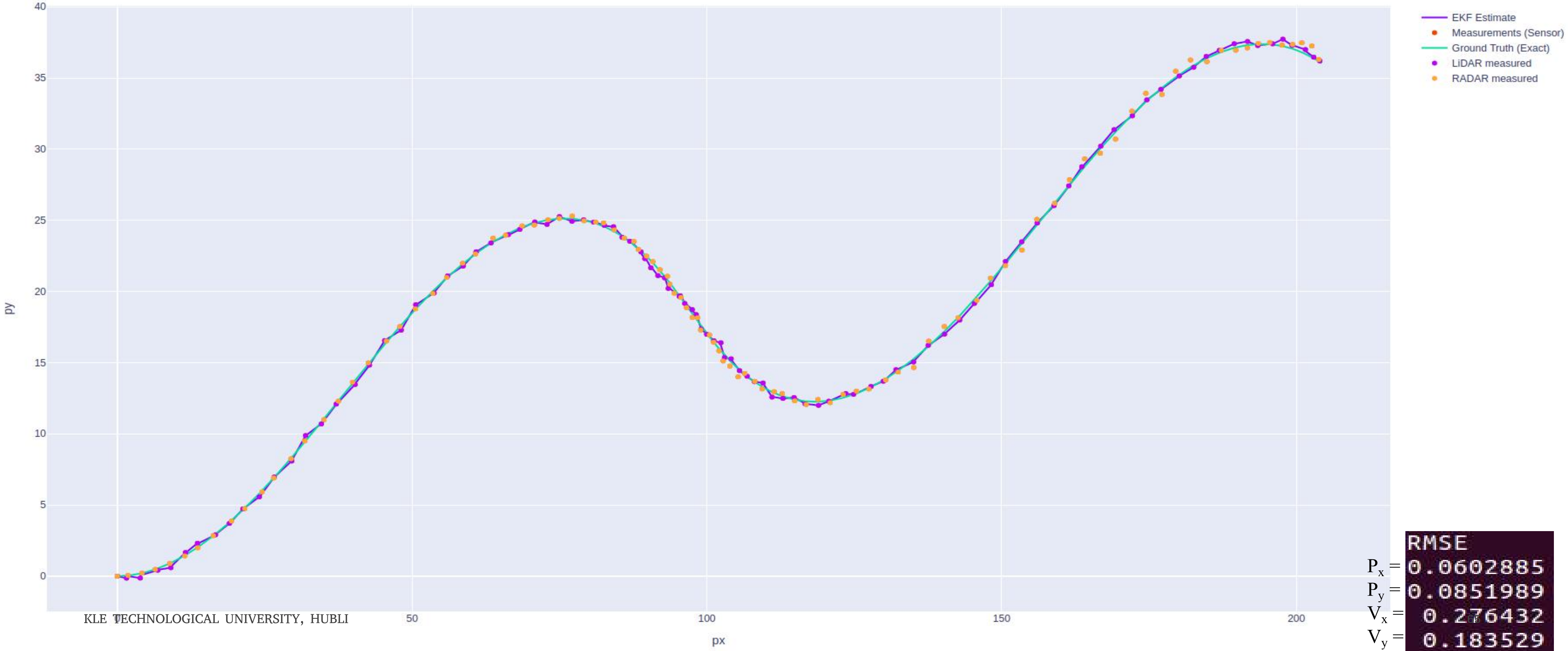
$P_x =$ 0.189321

$P_y =$ 0.188939

$V_x =$ 0.347146

$V_y =$ 0.503328

EKF IMPLEMANTATION FOR ROAD SENARIO:3



RMSE

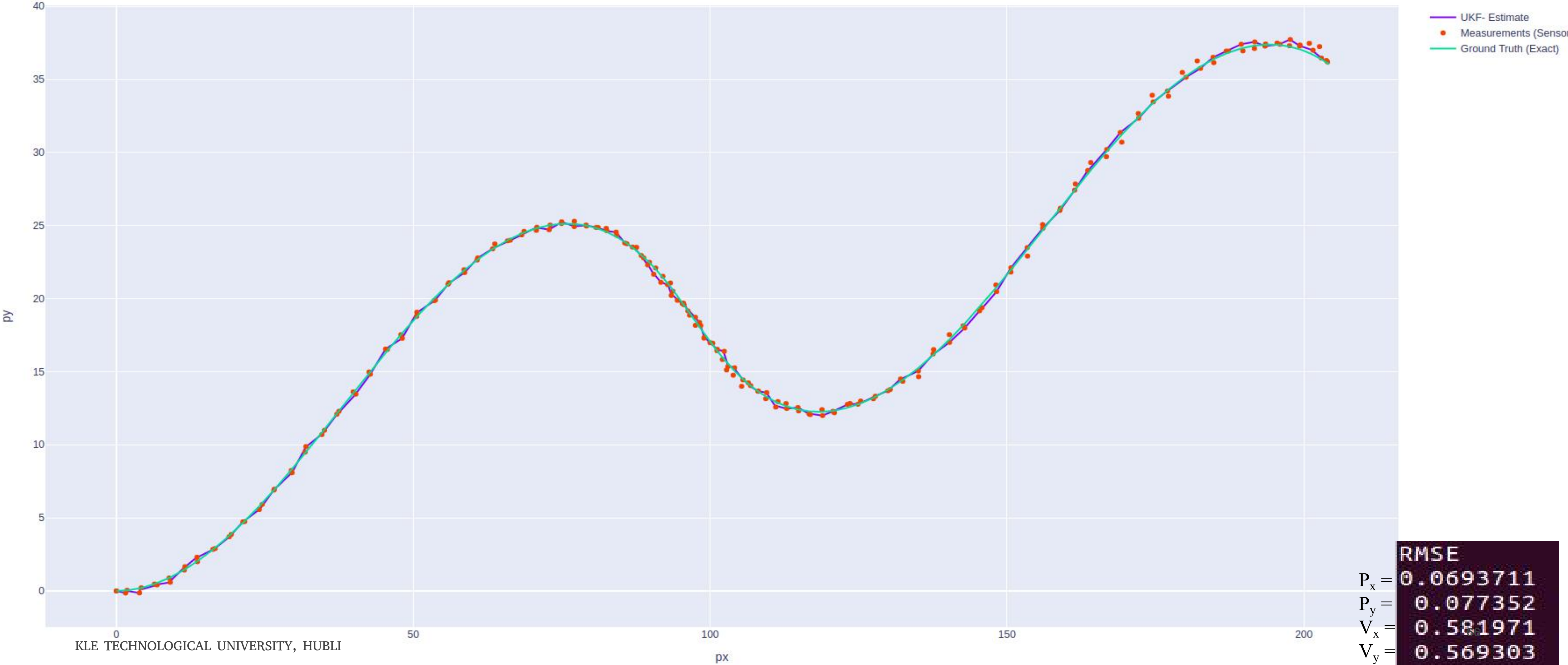
P_x = 0.0602885

P_y = 0.0851989

V_x = 0.276432

V_y = 0.183529

UKF IMPLEMANTATION FOR ROAD SENARIO:3



CONCLUSION

	Road 1					Road 2					Road 3				
	px	py	Max. ROC	vx	vy	px	py	Max. ROC	vx	vy	px	py	Max. ROC	vx	vy
KF	0.802	0.784	3250	1.448	2.805	0.185	0.19	512.27	0.474	0.804	0.189	0.188	86.66	0.347	0.503
EKF	0.508	0.389		1.098	1.659	0.097	0.852		0.418	0.478	0.0693	0.077		0.581	0.569
UKF	0.184	0.309		0.407	0.822	0.0651	0.0605		0.544	0.544	0.06	0.854		0.276	0.183

From the above table we can derive certain results:

- When the radius of curvature i.e ROC is smaller RMSE values of KF for position and velocity is smaller i.e KF able to make estimations accurately, when the ROC of the road is larger in such case RMSE values are larger i.e KF estimations are not so accurate this is because KF can handle only linear equations.
- While even though the ROC is larger EKF estimations are better than the KF estimations for position and velocity i.e smaller RMSE values for position and velocity this is because EKF can handle non-linear equations using the jacobian matrix, which is used to approximately linearize the non-linear function.
- But for real-world autonomous driving, it is important and necessary to have still more accurate estimations i.e very smaller RMSE values for position and velocity, hence we go for UKF estimations, where we estimate the position and velocity much accurately with negligible error in estimations i.e smaller RMSE values for position and velocity compared to KF and EKF estimations.

CONCLUSION

- A standard Kalman filter can only handle linear equations. Both the extended Kalman filter and the unscented Kalman filter allow us to use non-linear equations.
- Unscented Kalman filter yields better results compared to Kalman filter and extended Kalman filter algorithm.

REFERENCE

[1] Hojoon Lee, Heungseok Chae, Kyongsu Yi,``A Geometric Model based 2D LiDAR/Radar Sensor Fusion for Tracking Surrounding Vehicles", IFAC-PapersOnLine, Volume 52, Issue 8, 2019, Pages 130-135, ISSN:2405-8963.

[2] Hatem Hajri and Mohamed-Cherif Rahal,``Real Time Lidar and Radar High-Level Fusion for Obstacle Detection and Tracking with evaluation on a ground truth", July 2018.

[3] Shengbo Eben Li, Guofa Li, Jiaying Yu, Chang Liu, Bo Cheng, Jianqiang Wang, Keqiang Li, ``Kalman filter-based tracking of moving objects using linear ultrasonic sensor array for road vehicles", Mechanical Systems and Signal Processing, Volume 98, 2018, Pages 173-189, ISSN 0888-3270.

[4] Y. Maalej, S. Sorour, A. Abdel-Rahim and M. Guizani, ``Tracking 3D LIDAR Point Clouds Using Extended Kalman Filters in KITTI Driving Sequences," 018 IEEE Global Communications Conference (GLOBECOM), Abu Dhabi, United Arab Emirates, 2018, pp. 1-6, doi: 10.1109/GLOCOM.2018.8647803.

[5] M. Chiani, A. Giorgetti and E. Paolini, ``Sensor Radar for Object Tracking," in Proceedings of the IEEE, vol. 106, no. 6, pp. 1022-1041, June 2018, doi:10.1109/JPROC.2018.2819697 .

[6] Manjunath, Y. Liu, B. Henriques and A. Engstle, ``Radar Based Object Detection and Tracking for Autonomous Driving", 2018 IEEE MTT-S International Conference on Microwaves for Intelligent Mobility (ICMIM), Munich, 2018, pp. 1-4, doi: 10.1109/ICMIM.2018.8443497

[7] Mobus and U. Kolbe , ``Multi-target multi-object tracking, sensor fusion of radar and infrared," IEEE Intelligent Vehicles Symposium, 2004, Parma, Italy, 2004, pp. 732-737, doi: 10.1109/IVS.2004.1336475

Sensor fusion-based object tracking for self-driving cars using kalman filter

Basavaraj Navalgund

Under the guidance of Dr. Nalini C. Iyer and Prof. Akash Kulkarni

School of Electronics and Communication Egineering

basavarajnavalgund97@gmail.com

Problem statement

Environmental perception using LiDAR and RADAR sensors with Data combination and inference of data using Kalman filter, Extended Kalman filter and Unscented Kalman filter techniques.

Objectives

- Fusion of LiDAR and RADAR sensor data.
- Reduce sensor error.
- Track object's (vehicle, bicycle, person, etc) position, velocity, turning angle (yaw angle) and turning rate (yaw rate) accurately.

Contributions

Design of Kalman filter, Extended Kalman filter and Unsented Kalman filter:

- Fusion of heterogeneous sensors
- Optimal design of matrices
- Reducing individual sensor errors

System Model: Second Equation of Motion

$$S = u \cdot t + (1/2) \cdot a \cdot t^2$$
$$V = u + a \cdot t$$

Object State Vector:

$$X_{kp} = \begin{bmatrix} P_x \\ p_y \\ V \\ \emptyset \\ \emptyset\_dot \end{bmatrix}$$

Used Constant Turn Rate Speed (CTRS) motion model, hence considered position, velocity, yaw angle, and yaw rate.

Methodology

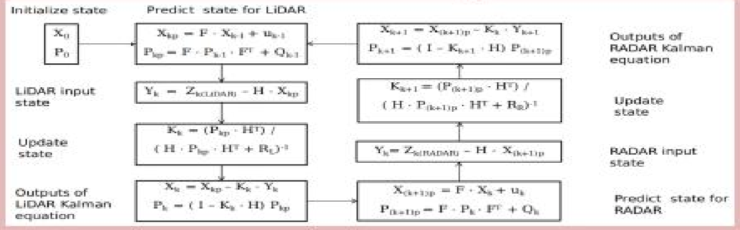


Fig. 1. Kalman filter block diagram

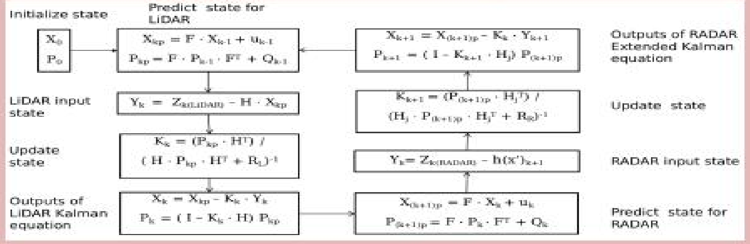
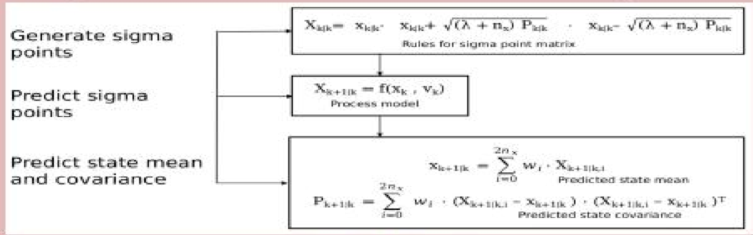


Fig. 2. Kalman & Extended Kalman filter block diagram



$$Z_{k+1|k} = h(x_{k+1})$$

$$z_{k+1|k} = \sum_{i=0}^{2nx} w_i \cdot Z_{k+1|k,i}$$

$$S_{k+1|k} = \sum_{i=0}^{2nx} w_i \cdot (Z_{k+1|k,i} - z_{k+1|k}) \cdot (Z_{k+1|k,i} - z_{k+1|k})^T + R$$

$$T_{k+1|k} = \sum_{i=0}^{2nx} w_i \, (X_{k+1|k,i} - x_{k+1|k}) \cdot (Z_{k+1|k,i} - z_{k+1|k})^T$$

$$K_{k+1|k} = T_{k+1|k} \cdot S_{k+1|k}^{-1}$$

$$X_{k+1|k+1} = x_{k+1|k} + K_{k+1|k} \cdot (Z_{k+1(Sensor)} - z_{k+1|k})$$

$$P_{k+1|k+1} = P_{k+1|k} - K_{k+1|k} \cdot S_{k+1|k} \cdot K_{k+1|k}^T$$

Fig. 3. Unscented Kalman filter block diagram

Results

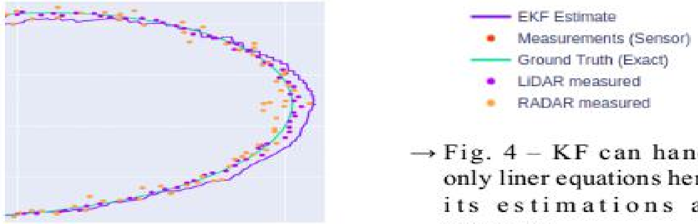


Fig. 4 - KF Plot

→ Fig. 4 – KF can handle only liner equations hence, its estimations are inaccurate.

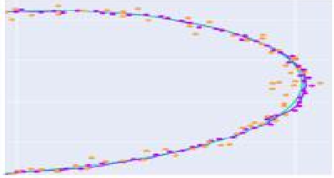


Fig. 5- EKF Plot

→ Fig. 5 - EKF can handle both liner & non linear equations hence, its estimations are better than KF.

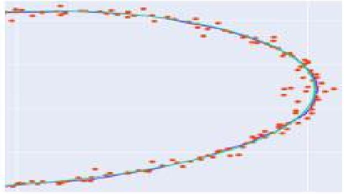


Fig. 6 - UKF Plot

→ Fig. 6 - UKF can handle both liner & non linear equations hence, its estimations are accurate than KF and EKF.

Conclusions

A standard Kalman filter handles linear equations. Both the extended Kalman and the Unscented Kalman filter allow us to use non-linear equations.Unscented Kalman filter yields better results compared to other presented algorithms.

Publications

Applied for 5th edition of Signal Processing and Communication conference 2020 for the topic Autonomous Navigation and Robotics.

THANK YOU