AMRITA VISHWA VIDYAPEETHAM
DEEMED TO BE UNIVERSITY

Bachelor of Technology In Artificial Intelligence

21MIS301 – Mathematics for Intelligent Systems

Observation Report

Topic:

Vehicle Tracking and Localisation using Kalman Filter

Done by:

V. Neeraj Chowdary – AIE21067

Thati Ayyappa Swamy – AIE21084

Professor/Guide:

Dr. Don S, Dr. Sunder Ram K

Table of content:

1 Introduction:

This project demonstrates the ability to detect and track multiple vehicles in real-time using a Kalman Filter, enabling a self-driving car to anticipate and react to surrounding vehicles.

1.1 Key Operations:

- Prediction of Current and Future Locations: Employ a Kalman Filter to predict a vehicle's current and future positions based on previous states, enhancing awareness of potential trajectories.
- Correction of Predictions: Incorporate new measurements from image frames to update and refine predicted states, ensuring accuracy and adaptability to changing conditions.
- Noise Reduction: Mitigate the impact of faulty detections by filtering out noise, resulting in smoother, more reliable tracking.

1.2 System Components:

- Camera: A camera mounted inside the self-driving car captures continuous image frames of the surrounding environment.
- Vehicle Detection Model: The pre-trained "ssd_mobilenet_v1_coco" model, leveraging the MS COCO dataset, identifies and localizes vehicles within each image frame.
- Bounding Box Generation: Bounding boxes are created around detected vehicles to precisely isolate their positions.
- State Estimation: The coordinates of bounding box corners are used to determine the state of each vehicle, including its position and velocity (assumed constant in this analysis).
- Kalman Filter: The Kalman Filter performs two primary operations:

  Prediction: Estimates the current state based on previous states and motion models.

  Update: Incorporates current measurements to correct the predicted state, reducing uncertainty and enhancing accuracy.

1.3 Data Flow:

- Image Capture: The camera continuously captures new image frames.
- Vehicle Detection: The model identifies vehicles within each frame and creates bounding boxes.
- State Estimation: Vehicle states are calculated from bounding box coordinates.
- Kalman Filter:

  Prediction: Predicts current states based on previous information.

  Update: Corrects predictions using current measurements.

- Tracking: The updated states are used to track vehicles across frames, generating smooth trajectories and enabling prediction of future positions.

2 Mathematical Implementation:

We use Kalman filter for tracking objects. Kalman filter has the following important features that tracking can benefit from:

- Prediction of object's future location.
- Correction of the prediction based on new measurement.
- Reduction of noise introduced by inaccurate detections.
- Facilitating the process of association of multiple objects to their tracks.

Kalman filter consists of two steps: prediction and update. The first step uses previous states to predict the current state. The second step uses the current measurement, such as detection bounding box location, to correct the state.

2.1 State equations:

The general linear dynamic system's state equation used is of the form:
$xk=xk-1+uk-1\Delta t+0.5*ak-1*\Delta t2$

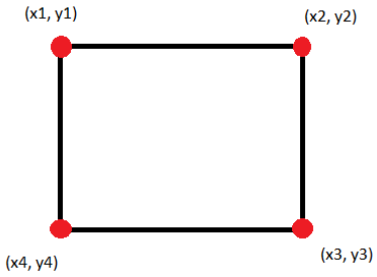
where,

- xk = state at a given time stamp 'k' (current state)
- xk−1 = = state at a given time 'k-1' (prior state)
- Δt is change in time
- u is the velocity at time 'k-1' (which is controlling the states)
- a is acceleration at time 'k-1'

In this analysis, we consider the object travelling in 2-D (both in X and Y direction).

Hence there will be state equations in terms of x and y.

Here, we are considering the four corners of the bounding box.



In a X-Y plane:

y1 = y2 ; y4 = y3 (since the pair of points are on same horizontal line)

x1 = x4 ; x2 = x3 (since the pair of points are on same vertical line)

Hence, there will be four state equations:

- $x1,k=x1,k-1+u1,k-1\Delta t$
- $x2,k=x2,k-1+u2,k-1\Delta t$
- $y1,k=y1,k-1+u1,k-1\Delta t$
- $y1,k=y1,k-1+u1,k-1\Delta t$

(As already mentioned, acceleration term is zero, since velocity is considered constant.)

Where u is velocity at k-1 timestamp.

The state equation is always paired with measurement equations, that describes the relationship between state and measurement at the current time stamp k.

The corresponding equations are obtained by taking the first derivative of states with respect to time:

$z_{x1,k} = u_{x1,k-1}$

$z_{x2,k} = u_{x2,k-1}$

$z_{y1,k} = u_{y1,k-1}$

$z_{y2,k} = u_{y2,k-1}$

Where u is velocity at k-1 timestamp

2.2 Notation:

X - State Mean

P - State Covariance

F - State Transition Matrix

Q - Process Covariance

B - Control Function

u - Control Input

Here, Q consists of the variances associated with each of the state estimates as well as the correlation between the errors in the state estimates.

$$Q = \begin{bmatrix} \Delta t^4/4 & \Delta t^3/2 \\ \Delta t^3/2 & \Delta t^2 \end{bmatrix}$$

All the variables and their respective coefficients are grouped in the form of a matrices:

2.2.1 State Transition matrix:

$$F = \begin{bmatrix} 1 & \Delta t & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & \Delta t & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & \Delta t & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & \Delta t \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$ his contains all the coefficients of state equation.

2.2.2 Measurement matrix:

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

This is also known as Control matrix, which contains all the coefficients of measurement equation

2.3 Prediction Phase equations

$\bar{x}=Fx+Bu$, where

u = control input matrix

$\bar{P}=FPF_T+Q$

Here,

$$Fx = \begin{bmatrix} y_{1,k-1} + u_{y1,k-1}\Delta t \\ u_{y1,k-1} \\ x_{1k-1} + u_{x1,k-1}\Delta t \\ u_{x1,k-1} \\ y_{2,k-1} + u_{y2,k-1}\Delta t \\ u_{y2,k-1} \\ x_{2,k-1} + u_{x2,k-1}\Delta t \\ u_{x2,k-1} \end{bmatrix},$$

$$Bu = \begin{bmatrix} y_{1k-1} \\ x_{1k-1} \\ y_{2k-1} \\ x_{2k-1} \end{bmatrix}.$$

2.4 Update phase equation:

H - Measurement matrix

z - measurement

R - measiurement noise covariance

y - reisdual

K - Kalman gain

Here,

H is a transformation matrix that transforms the state into the measurement domain.

Kalman gain is just a weight given to the measurements and the current state estimates).

Residual is difference between the measurement and the value predicted by the filter.

$$y = z - H\bar{x}$$
$$K = \bar{P}H^T(H\bar{P}H^T + R)^-1$$
$$x = \bar{x} + Ky$$
$$P = (I - KH)\bar{P}$$

3 Drawbacks or Novelty:

3.1 Drawbacks for existing model:

Considering the model(https://www.cs.montana.edu/techreports/1314/Le.pdf)

- The model assumes a linear relationship between the RSSI and the distance, which may not hold in complex environments with multipath propagation and shadowing effects.
- The model requires a Gaussian distribution for the measurement noise and the process noise, which may not be realistic for the RSSI data.
- The model relies on the availability and accuracy of the velocity information of the mobile node, which may not be always accessible or reliable.
- The model may not be able to handle dynamic changes in the number and location of the anchor nodes, which may affect the multiliterate algorithm.

3.2 Novelty for this model:

Kalman filters have been widely used in various applications due to their ability to provide optimal estimates for dynamic systems. Here are some novel applications of Kalman filters:

Navigation with a Global Navigation Satellite System (GNSS): The Kalman filter provides optimal estimates for linear models with additive Gaussian noises. An implementation example of the Kalman filter is navigation with a GNSS.

Bearing-Angle Target Tracking and Terrain-Referenced Navigation (TRN): The extended Kalman filter is utilized for nonlinear problems like bearing-angle target tracking and TRN.

Machine Learning: The Kalman filter is also expected to provide the optimal estimation of the state variable and parameters of quantum systems.

Optical Communication Systems: Kalman filters have been used in optical communication systems.

Robot Control and Trajectory Tracking: Sixty years after its creation, the Kalman filter is still used in autonomous navigation processes, robot control, and trajectory tracking.

Economics and Medicine: The filter is not only restricted to robotics but is also present in different fields, such as economics and medicine.

These applications demonstrate the versatility and effectiveness of Kalman filters in various fields. However, the specific novelty would depend on the particular application and how the Kalman filter is used or adapted for that application.

4 Pseudo Code:

4.1 Kalman Filter Implementation:

In this section, we describe the implementation of the Kalman filter in detail.

The state vector has eight elements as follows:

- [up, up_dot, left, left_dot, down, down_dot, right, right_dot]

That is, we use the coordinates and their first-order derivatives of the up left corner and lower right corner of the bounding box.

The process matrix, assuming the constant velocity (thus no acceleration), is:

```
self.F = np.array([[1, self.dt, 0,  0,  0,  0,  0, 0],
                   [0, 1,  0,  0,  0,  0,  0, 0],
                   [0, 0,  1,  self.dt, 0,  0,  0, 0],
                   [0, 0,  0,  1,  0,  0,  0, 0],
                   [0, 0,  0,  0,  1,  self.dt, 0, 0],
                   [0, 0,  0,  0,  0,  1,  0, 0],
                   [0, 0,  0,  0,  0,  0,  1, self.dt],
                   [0, 0,  0,  0,  0,  0,  0,  1]])
```

The measurement matrix, given that the detector only outputs the coordindate (not velocity), is:

```
self.H = np.array([[1, 0, 0, 0, 0, 0, 0, 0],
          [0, 0, 1, 0, 0, 0, 0, 0],
          [0, 0, 0, 0, 1, 0, 0, 0],
          [0, 0, 0, 0, 0, 0, 1, 0]])
```

The state, process, and measurement noises are :

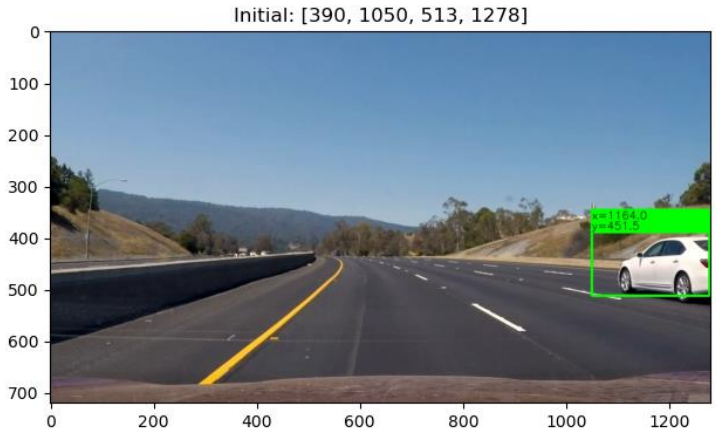
```
# Initialize the state covariance
 self.L = 100.0
 self.P = np.diag(self.L*np.ones(8))
 # Initialize the process covariance
 self.Q_comp_mat = np.array([[self.dt**4/2., self.dt**3/2.],
                             [self.dt**3/2., self.dt**2]])
 self.Q = block_diag(self.Q_comp_mat, self.Q_comp_mat,
                     self.Q_comp_mat, self.Q_comp_mat)
# Initialize the measurement covariance
```

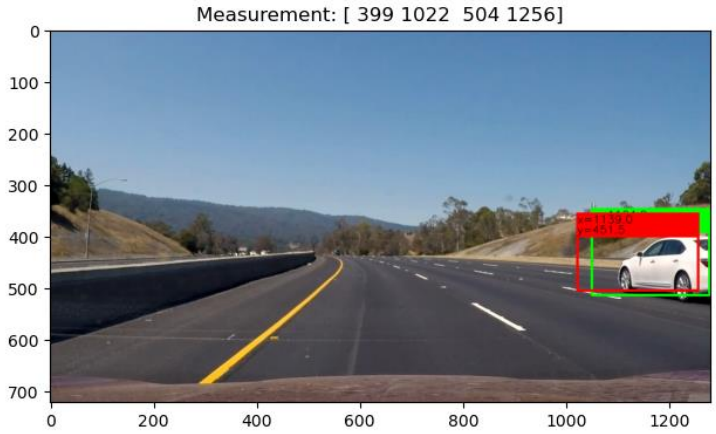
```
self.R_scaler = 1.0/16.0
self.R_diag_array = self.R_ratio * np.array([self.L, self.L, self.L, self.L])
self.R = np.diag(self.R_diag_array)
```

5 Sample Implementation:

Here **self.R_scaler** represents the "magnitude" of measurement noise relative to state noise. A low **self.R_scaler** indicates a more reliable measurement. The following figures visualize the impact of measurement noise to the Kalman filter process. The green bounding box represents the prediction (initial) state. The red bounding box represents the measurement. If measurement noise is low, the updated state (aqua coloured bounding box) is very close to the measurement (aqua bounding box completely overlaps over the red bounding box).

In contrast, if measurement noise is high, the updated state is very close to the initial prediction (aqua bounding box completely overlaps over the green bounding box).





From the above output images, we can visualize the measurement noises in the Kalman filter process.

The green bounding box gives the initial state of the car. The red bounding box gives the measurement values. The aqua coloured bounding box give the updated state.

Here, the initial state value is 390 The measurement state value is 399 after 1 second Updated state value is 398

If measurement noise is low, the updated state is very close to the measurement. So, the aqua bounding box completely overlaps the red bounding box.

Suppose, if measurement noise is high, the updated state is very close to the initial prediction. So, the aqua bounding box completely overlaps over the green bounding box.

From the output images, there is no complete overlapping of green and aqua bounding box. Which infers that the measurement noise is relatively lower.

6 GitHub links:

- VehicleVision

7 Functionality requirement:

A Functional Requirement (FR) is a description of the service that the software must offer. It describes a software system or its component. ... It can be a calculation, data manipulation, business process, user interaction, or any other specific functionality which defines what function a system is likely to perform.

7.1 Hardware-requirements:

Hardware Requirements:

1. RAM: 4GB
2. Storage: 1.5GB
3. CPU: 2 GHz or faster
4. GPU: 1 GHz or faster
5. Architecture: 64-bit