Technitute_

Advanced Kalman Filtering and Sensor Fusion

2D Vehicle Extended Kalman Filter: Update Step

EKF Exercise 2

2D Vehicle EKF:
Update Step Exercise

Overview

Implement the ***Kalman Filter Update Equations*** and the ***Lidar Measurement Model***.

Step 1 (Setup)

- Open your last kalman filter file from the previous exercise which had the prediction step completed.
- Review the code that is already been written for the Update Steps as part of the EKF exercises.
    - ***handleLidarMeasurements()*** is called after the prediction step whenever LIDAR measurements have been made by the sensor. This function then sequentially calls the ***handleLidarMeasurement()*** function which is the main function you will modify to fuse the Lidar measurements ***one*** at a time.

```
void KalmanFilter::handleLidarMeasurements(const std::vector<LidarMeasurement>& dataset, const BeaconMap& map)
{
    // Assume No Correlation between the Measurements and Update Sequentially
    for(const auto& meas : dataset) {handleLidarMeasurement(meas, map);}
}

void KalmanFilter::handleLidarMeasurement(LidarMeasurement meas, const BeaconMap& map)
{
    if (isInitialised())
    {
        VectorXd state = getState();
        MatrixXd cov = getCovariance();

        // Implement The Kalman Filter Update Step for the Lidar Measurements in the
        // section below.
        // HINT: use the wrapAngle() function on angular values to always keep angle
        // values within correct range, otherwise strange angle effects might be seen.
        // HINT: You can use the constants: LIDAR_RANGE_STD, LIDAR_THETA_STD
        // ----------------------------------------------------------------------- //
        // ENTER YOUR CODE HERE

        BeaconData map_beacon = map.getBeaconWithId(meas.id); // Match Beacon with built in Data Association Id
        if (meas.id != -1 && map_beacon.id != -1)
        {

        }

        // ----------------------------------------------------------------------- //

        setState(state);
        setCovariance(cov);
    }
}
```

2D Vehicle EKF:
Update Step Exercise

Step 2 (Implement the Lidar Measurement Model)

- Modify the function ***handleLidarMeasurement()***

$$\begin{bmatrix} \hat{r} \\ \hat{\theta} \end{bmatrix} = \begin{bmatrix} \sqrt{(\overset{\texttt{map\_beacon.x}}{L_x} - \hat{p}_x)^2 + (\overset{\texttt{map\_beacon.y}}{L_y} - \hat{p}_y)^2} \\ \arctan\left(\frac{L_y - \hat{p}_y}{L_x - \hat{p}_x}\right) - \hat{\psi} \end{bmatrix}$$

NOTE:
The code assumes that the state vector has the following form!!

$$\hat{x} = \begin{bmatrix} p_x \\ p_y \\ \psi \\ V \end{bmatrix}$$

Step 3 (Implement the Innovation Calculations)

- Modify the function ***handleLidarMeasurement()***
- Calculate the Measurement Innovation (make sure you normalise the angle innovation!)
- Calculate the Measurement Jacobian Matrix (H matrix)
- Calculate the Measurement Innovation Covariance (S matrix)
- Assume the range and theta measurement std are LIDAR_RANGE_STD and LIDAR_THETA_STD

2D Vehicle EKF:
Update Step Exercise

Step 3 (Implement the Innovation Calculations)

- Modify the function ***handleLidarMeasurement()***
- Calculate the Measurement Innovation (make sure you normalise the angle innovation!)

```
y(1) = wrapAngle(y(1)); // Wrap the Heading Innovation
```

- Calculate the Measurement Jacobian Matrix (H matrix)
- Calculate the Measurement Innovation Covariance (S matrix)
- Assume the range and theta measurement std are LIDAR_RANGE_STD and LIDAR_THETA_STD.

$$\nu = \begin{bmatrix} r \\ \theta \end{bmatrix}_{\text{meas}} - \begin{bmatrix} \hat{r} \\ \hat{\theta} \end{bmatrix}$$

$$\mathbf{R} = \begin{bmatrix} \sigma_r^2 & 0 \\ 0 & \sigma_\theta^2 \end{bmatrix}$$

$$\mathbf{H} = \nabla h_x = \begin{bmatrix} \frac{1}{d}(\hat{p}_x - L_x) & \frac{1}{d}(\hat{p}_y - L_y) & 0 & 0 \\ \frac{-1}{d^2}(\hat{p}_y - L_y) & \frac{1}{d^2}(\hat{p}_x - L_x) & -1 & 0 \end{bmatrix}$$

$$d = \sqrt{(L_x - \hat{p}_x)^2 + (L_y - \hat{p}_y)^2}$$

$$\mathbf{S} = \nabla h_x \mathbf{P}_k^- \nabla h_x^T + \mathbf{R}$$

2D Vehicle EKF:
Update Step Exercise

Step 4 (Implement the Kalman Filter Update Step Equations)

$$\hat{x}_k^+ = \hat{x}_k^- + \mathbf{K}_k \nu_k$$
$$\mathbf{K}_k = \mathbf{P}_k^- \nabla h_x^T \mathbf{S}_k^{-1}$$
$$\mathbf{P}_k^+ = (\mathbf{I} - \mathbf{K}_k \nabla h_x) \mathbf{P}_k^-$$

2D Vehicle EKF:
Update Step Exercise

Step 5 (Run the Simulation in the following configurations)

- Profile 1 (Constant Speed/Heading, Zero Initial Conditions)
- Profile 2 (No-Zero Initial Conditions)
- Profile 3 (Constant Speed, Changing Headings)
- Profile 4 (Changing Speed, Changing Headings)
- Profile 5 (Profile 1 + LIDAR)
- Profile 6 (Profile 2 + LIDAR)
- Profile 7 (Profile 3 + LIDAR)
- Profile 8 (Profile 4 + LIDAR)

Step 6 (Compare the Simulation Results between the LKF and EKF with/without Lidar)