Mapping and Perception for an Autonomous Robot
Project 2

Nadav Marciano 305165698

Part A: Kalman Filter

a. In accordance with the table, I downloaded the data of driver 20 from the KITTI dataset.
b. The data is extracted using the given function.
c. we get the plots for the LLA trajectory and the matched ENU path:

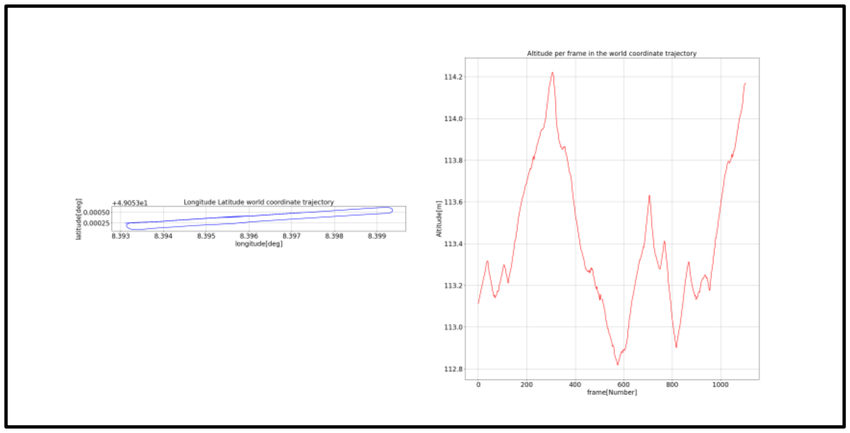


Figure 1 - Ground-truth GPS trajectory LLA

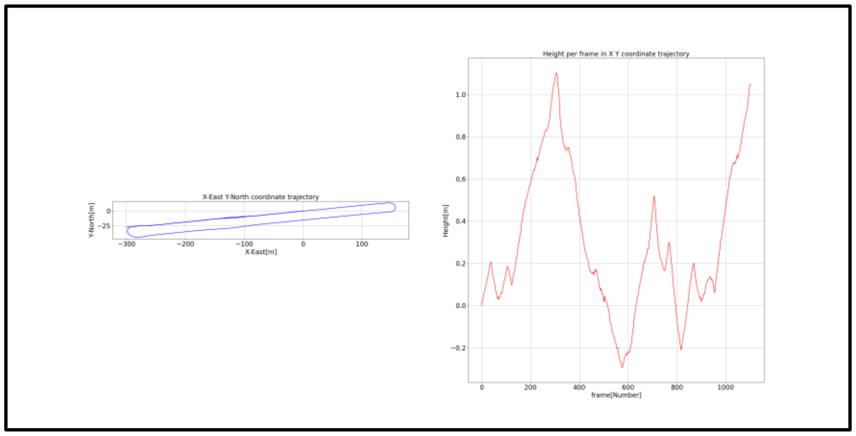


Figure 2 - Ground- truth GPS trajectory ENU

In our analysis, we have plotted the ground truth (GT) path using both LLA coordinates and ENU coordinates to represent the vehicle's trajectory. By comparing these graphs, we can verify the data's consistency and accuracy. Both graphs should depict the same trajectory, confirming our coordinate transformations are correct. This ensures accurate representations for further analyses, such as Kalman filtering.

According to Google Maps, this is the route:

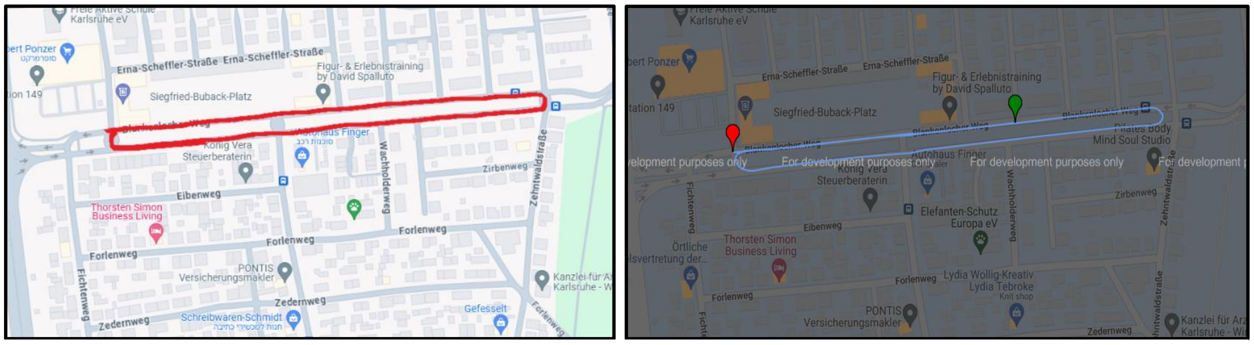


Figure 3 - Vehicle Route on Google Maps

The drive takes place on a one-way road along an intercity route near a residential neighborhood. The trajectory follows a closed-loop path. There are signs along the roadside, and dynamic objects include cars also traveling on the road.

d. Adding Gaussian noise to the ground-truth GPS data generates noisy observations intended for input into the Kalman filter:

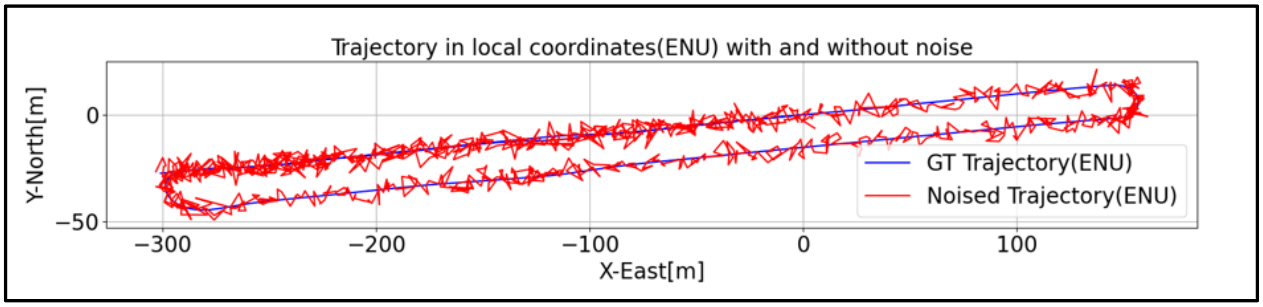


Figure 4 - Comparison of the ENU path with and noise

e. The Kalman Filter operates on a linear system where the state evolves over time according to a known model, and observations are made at each step.

A linear process model: $X_t = A_t X_{t-1} + B_t u_t + \epsilon_t \qquad \epsilon_t \sim N(0, R)$

$$\begin{bmatrix} x_t \\ v_{x_t} \\ y_t \\ v_{y_t} \end{bmatrix} = \begin{bmatrix} 1 & \Delta t & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \Delta t \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{t-1} \\ v_{x_{t-1}} \\ y_{t-1} \\ v_{y_{t-1}} \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} + \epsilon_t$$

Observations: $\hat{Z}_t = C X_t + \delta_t \qquad \delta_t \sim N(0, Q)$

$$\begin{bmatrix} Z_{x_t} \\ Z_{y_t} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_t \\ v_{x_t} \\ y_t \\ v_{y_t} \end{bmatrix} + \delta_t$$

Kalman Filter Flow:

1. Initialize Parameters: initial state, covariance, measurement covariance Q, and noise covariance R.

Initial state $x_0 = \begin{bmatrix} x_0 \\ v_{x_0} \\ y_0 \\ v_{y_0} \end{bmatrix}$ Initial Covariance $P_0 = \begin{bmatrix} k\sigma_{x_0}^2 & 0 & 0 & 0 \\ 0 & k\sigma_{xv_0}^2 & 0 & 0 \\ 0 & 0 & k\sigma_{y_0}^2 & 0 \\ 0 & 0 & 0 & k\sigma_{yv_0}^2 \end{bmatrix}$

Noise covariance $R = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & \Delta t & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \Delta t \end{bmatrix} \sigma_n^2$ Measurement covariance $Q = \begin{bmatrix} \sigma_x^2 & 0 \\ 0 & \sigma_y^2 \end{bmatrix}$

2. Build Matrices: Construct matrices A, B, and C. In our case, these matrices define the relationship for 2D position and speed in both axes.

$$A = \begin{bmatrix} 1 & \Delta t & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \Delta t \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad B = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \qquad C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

We initialize the initial position using the noisy GPS data. For the initial velocity in both the x and y directions, we select a value of 10 m/s. This setup corresponds to the 2D case discussed in the lecture. The necessary matrices are defined accordingly. Since there is no control input in this section, the control matrix B is set to zero.

3. Iterate Over Frames (Data Measurements):

- Predict: Using the constant motion model to forecast the vehicle's state and compute the filter's covariance matrix Σ.

$\bar{\mu}_t = A_t \mu_{t-1} + B_t u_t$

$\bar{\Sigma}_t = A_t \Sigma_{t-1} A_t^T + R_t$

Where μ represents the state vector x, Σ denotes the covariance matrix P, and R is the process noise covariance matrix Q.

- Kalman Gain: Calculation of the Kalman Gain, which determines how much weight to assign to the current measurement versus the predicted state estimate during the update.

$K_t = \bar{\Sigma}_t C_t^T (C_t \bar{\Sigma}_t C_t^T + Q_t)^{-1}$

- Correct: Adjust the state estimate and covariance based on the Kalman Gain to refine the estimation.

$\mu_t = \bar{\mu}_t + K_t(z_t - C_t \bar{\mu}_t)$

$\Sigma_t = (I - K_t C_t)\bar{\Sigma}_t$

Ultimately, we are left with two parameters, k and σ_n, for conducting the empirical test. I explored k values ranging from 1 to 5, with a fixed parameter of σ_n=0.5. The effect of k was minimal, and the best result was achieved with k=1. Subsequently, I set k=1 and varied σ_n from 0 to 2 in steps of 0.1 to determine the optimal parameter. Here are the results of σ_n as a function of RMSE and maxE:

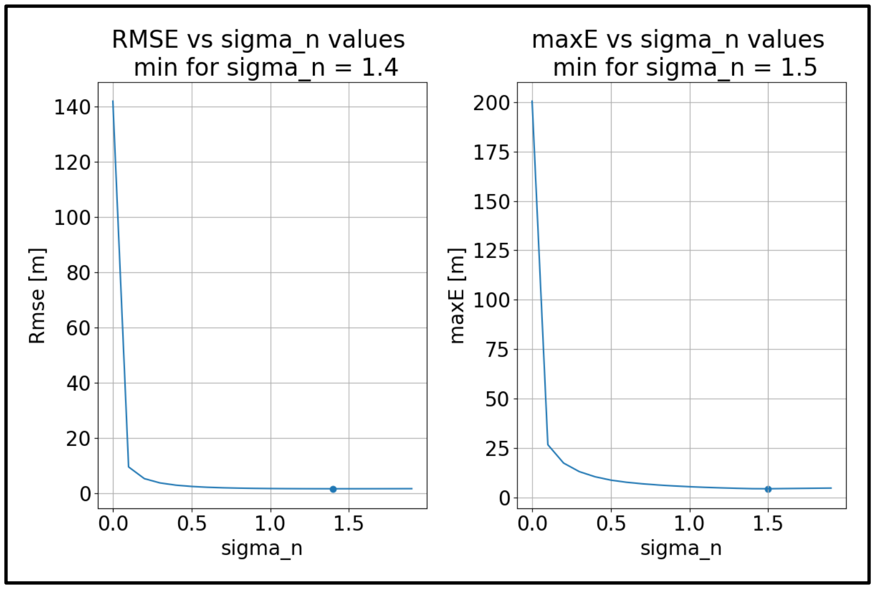


Figure 5 – Kalman filter calibration

Based on these results, I chose σ_n=1.5.

f. Result analysis ground-truth and estimated results:

The values achieved in our test were as follows:
maxE: 4.304851892707902 , **RMSE:** 1.6260814977429983

The maximum error represents the largest single discrepancy between the true position and the estimated position by the Kalman filter. Achieving a maxE of 4.304851892707902 indicates that the largest deviation in our estimates was about 4.3 from the true path.

The Root Mean Square Error is a measure of the average magnitude of the errors between the estimated positions and the true positions, providing an overall sense of how well the Kalman filter is performing. An RMSE of 1.6260814977429983 indicates that, on average, our position estimates were about 1.63 away from the true positions.

These results demonstrate that the chosen parameters (with σ_n=1.5) effectively minimized both the maximum error and the average error, indicating a well-calibrated Kalman filter for our application.

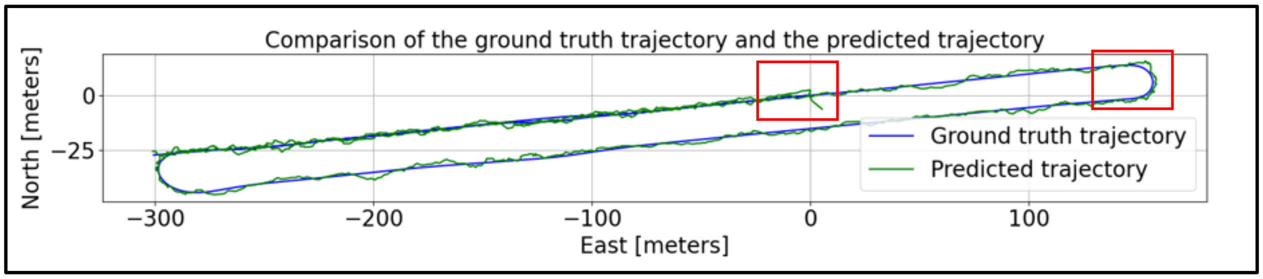
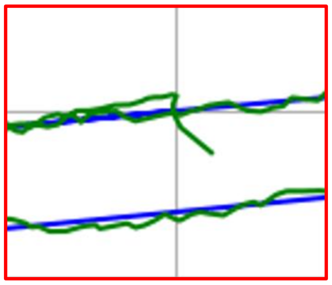
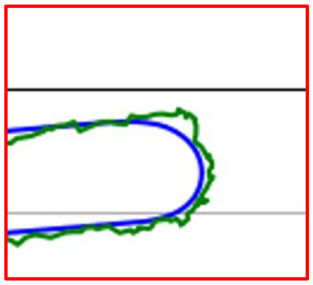


Figure 6 – Comparison of the ground truth trajectory and the filtered trajectory (KF)

Let's focus on the interesting parts. As expected, it takes a while for the filter to adjust initially, resulting in a noisy path. It's notable that our trajectory challenges the filter's accuracy, particularly during slight turns, where errors noticeably increase.





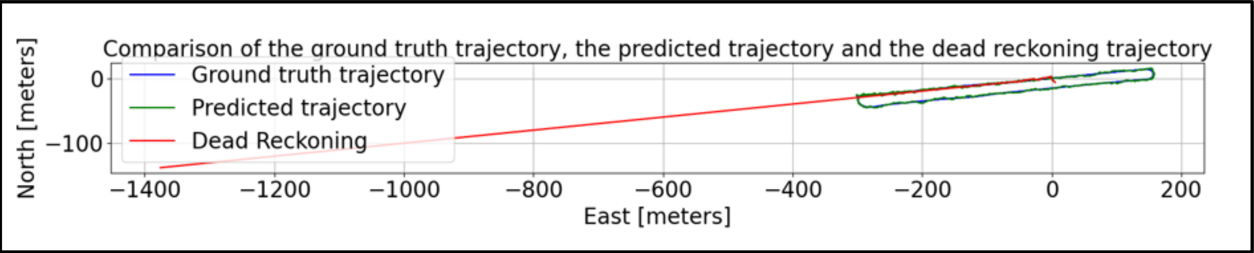


Figure 7 – Comparison of the ground truth trajectory and the dead reckoning

After approximately 5 seconds, we zeroed out the Kalman gain and therefore we have no correction. The car goes in a straight line and the measurements have no effect. The fine details are challenging to discern due to the image and video resolution limitations, despite zooming in. Nevertheless, the filter's smoothing capability with the ground truth path is evident, demonstrating its effectiveness in mitigating noise and maintaining alignment with the actual trajectory.

g. After analyzing the estimation results, it's essential to verify if we meet the uncertainty conditions. Specifically, we need to determine if the errors fall within the range of +/- sigma.

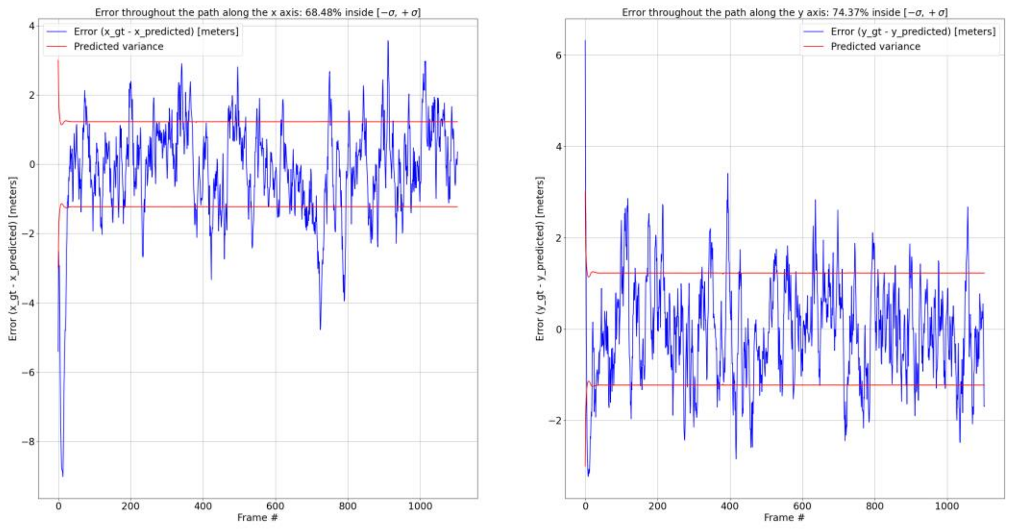


Figure 8 – Plot trajectory error

These percentages indicate that a significant portion of the errors fall within expected bounds. Specifically, approximately 68.48% of predictions in the X direction and 74.37% in the Y direction are within the typical range of ~68%, demonstrating reasonably accurate estimation performance without exceeding expected levels of uncertainty.

Part B: Extended Kalman Filter

a. The same data as in part A.
b. The data is extracted using the given function.
c. Adding Gaussian noise to the ground-truth GPS data generates noisy observations intended for input into the Extended Kalman filter.
d. Adding Gaussian noise to forward velocity and angular rate:

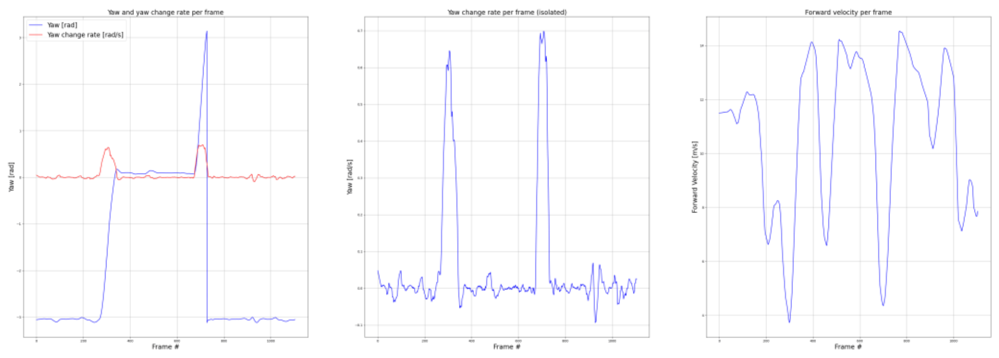


Figure 9 – Plot of Yaw and forward velocity per frame

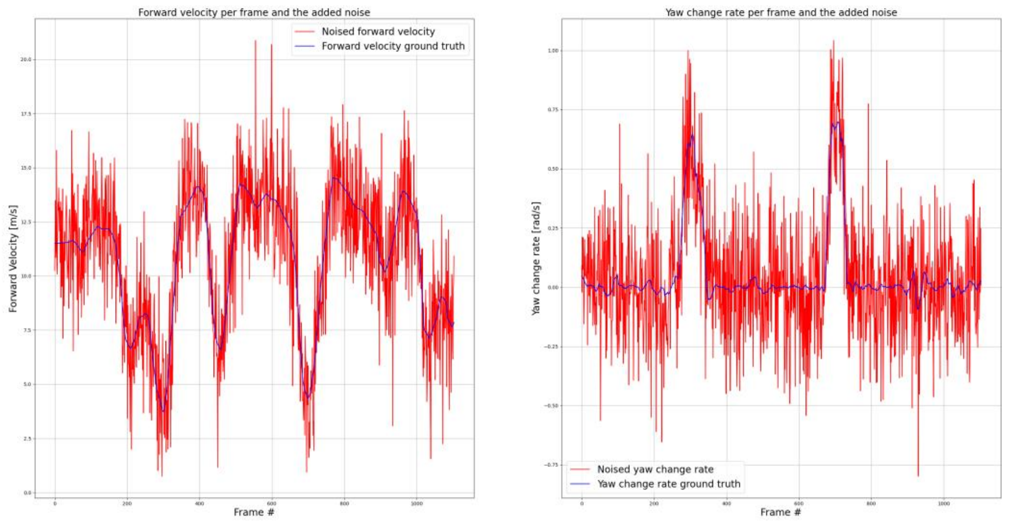


Figure 10 – Plot of noised yaw and forward velocity per frame

I added Gaussian noise to the ground-truth GPS/IMU data, with a standard deviation of 3 meters for both the x and y coordinates, consistent with the approach used in part A. Additionally, angular rates were perturbed with Gaussian noise characterized by a standard deviation of 0.2 rad/s, and forward velocity rates were similarly noised with a standard deviation of 2 m/s. These noisy observations will be utilized as inputs to the Extended Kalman Filter in subsequent analyses.

e. The Extended Kalman Filter operates on a nonlinear system where the state evolves over time according to a known model, and nonlinear observations are made at each step.

A non-linear process model: $X_t = g(u_t, X_{t-1}) + \epsilon_t \qquad \epsilon_t \sim N(0, R)$

$$\begin{bmatrix} x_t \\ y_t \\ \theta_t \end{bmatrix} = \begin{bmatrix} x_{t-1} \\ y_{t-1} \\ \theta_{t-1} \end{bmatrix} + \begin{bmatrix} -\frac{v_t}{w_t}\sin(\theta_{t-1}) + \frac{v_t}{w_t}\sin(\theta_{t-1} + w_t\Delta t) \\ \frac{v_t}{w_t}\cos(\theta_{t-1}) - \frac{v_t}{w_t}\cos(\theta_{t-1} + w_t\Delta t) \\ w_t\Delta t \end{bmatrix} + \epsilon_t$$

Observations: $\widehat{Z_t} = h(X_t) + \delta_t \qquad \delta_t \sim N(0, Q)$

$$\begin{bmatrix} Z_{x_t} \\ Z_{y_t} \end{bmatrix} = h\left(\begin{bmatrix} x_t \\ y_t \\ \theta_t \end{bmatrix}\right) + \delta_t = H_t x_t + \delta_t$$

Kalman Filter Flow:

1. **Initialize Parameters**: initial state, covariance, measurement covariance Q, and noise covariance R.

Initial state $x_0 = \begin{bmatrix} x_0 \\ y_0 \\ \theta_0 \end{bmatrix}$ Initial Covariance $P_0 = \begin{bmatrix} k\sigma_{x_0}^2 & 0 & 0 \\ 0 & k\sigma_{y_0}^2 & 0 \\ 0 & 0 & k\sigma_{\theta_0}^2 \end{bmatrix}$

Noise covariance $R = \begin{bmatrix} \sigma_v^2 & 0 \\ 0 & \sigma_\omega^2 \end{bmatrix}$ Measurement covariance $Q = \begin{bmatrix} \sigma_x^2 & 0 \\ 0 & \sigma_y^2 \end{bmatrix}$

2. **Build Matrices**: Construct matrices G, V, and H. In our case, these matrices define the relationships and transformations between the state variables, the process noise, and the observations.

$$V_t = \frac{\partial g(\mu_t, \mu_{t-1})}{\partial \mu_t}$$
$$= \begin{bmatrix} -\frac{1}{w_t}\sin(\theta_{t-1}) + \frac{1}{w_t}\sin(\theta_{t-1} + w_t\Delta t) & \frac{v_t}{w_t^2}\sin(\theta_{t-1}) - \frac{v_t}{w_t^2}\sin(\theta_{t-1} + w_t\Delta t) + \frac{v_t}{w_t}\cos(\theta_{t-1} + w_t\Delta t)\Delta t \\ \frac{1}{w_t}\cos(\theta_{t-1}) - \frac{1}{w_t}\cos(\theta_{t-1} + w_t\Delta t) & -\frac{v_t}{w_t^2}\cos(\theta_{t-1}) + \frac{v_t}{w_t^2}\cos(\theta_{t-1} + w_t\Delta t) + \frac{v_t}{w_t}\sin(\theta_{t-1} + w_t\Delta t)\Delta t \\ 0 & \Delta t \end{bmatrix}$$

$$G_t = \frac{\partial g(\mu_t, \mu_{t-1})}{\partial \mu_{t-1}} = \begin{bmatrix} 1 & 0 & -\frac{v_t}{w_t}\cos(\theta_{t-1}) + \frac{v_t}{w_t}\cos(\theta_{t-1} + w_t\Delta t) \\ 0 & 1 & -\frac{v_t}{w_t}\sin(\theta_{t-1}) + \frac{v_t}{w_t}\sin(\theta_{t-1} + w_t\Delta t) \\ 0 & 0 & 1 \end{bmatrix} \qquad H = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

We initialize the first position Q using the noisy GPS data and derive the yaw from the IMU data. This time, we estimate both the 2D position and the heading angle of the object. The noisy velocity and angular rate are also given, along with the known covariance matrices R and Q. To match the GPS dimension data, we use the matrix H to transform our state into the 2D position, similar to the role of matrix C in the linear Kalman Filter.

3. Iterate Over Frames (Data Measurements):

<u>- Predict:</u> Using the motion model to forecast the vehicle's state and compute the filter's covariance matrix Σ.

$\bar{\mu}_t = g(u_t, \mu_{t-1})$

$\bar{\Sigma}_t = G_t \Sigma_{t-1} G_t^T + R_t$

Where μ represents the state vector x, Σ denotes the covariance matrix P, and R is the process noise covariance matrix Q.

<u>- Kalman Gain:</u> Calculation of the Kalman Gain, which determines how much weight to assign to the current measurement versus the predicted state estimate during the update.

$K_t = \bar{\Sigma}_t H_t^T (H_t \bar{\Sigma}_t H_t^T + Q_t)^{-1}$

<u>- Correct:</u> Adjust the state estimate and covariance based on the Kalman Gain to refine the estimation.

$\mu_t = \bar{\mu}_t + K_t(z_t - h(\bar{\mu}_t))$

$\Sigma_t = (I - K_t H_t)\bar{\Sigma}_t$

Using the same noisy GPS data, we set the initial standard deviations to $\sigma_x = \sigma_y$=3. For the angular rate, we use the known Gaussian noise standard deviation, σ_w= 0.2, which has yielded satisfactory results. And σ_{vf}= 2 Next, we determine the values of k and any additional noise to enhance performance. We test k values from 1 to 5. After testing various options and iterating through several choices, I settled on σ_θ= 0.18, which produced excellent results.

f. Result analysis ground-truth and estimated results:

The values achieved in our test were as follows:
maxE: 3.5285106239445625, **RMSE:** 1.3169028684098618

The maximum error represents the largest single discrepancy between the true position and the estimated position by the Extended Kalman Filter. Achieving a maxE of 3.5285106239445625 indicates that the largest deviation in our estimates was about 3.53 from the true path.

The Root Mean Square Error is a measure of the average magnitude of the errors between the estimated positions and the true positions, providing an overall sense of how well the Extended Kalman Filter is performing. An RMSE of 1.3169028684098618 indicates that, on average, our position estimates were about 1.32 away from the true positions.

These results demonstrate that the chosen parameters (with σ_θ= 0.18) effectively minimized both the maximum error and the average error, indicating a well-calibrated Extended Kalman Filter for our application.

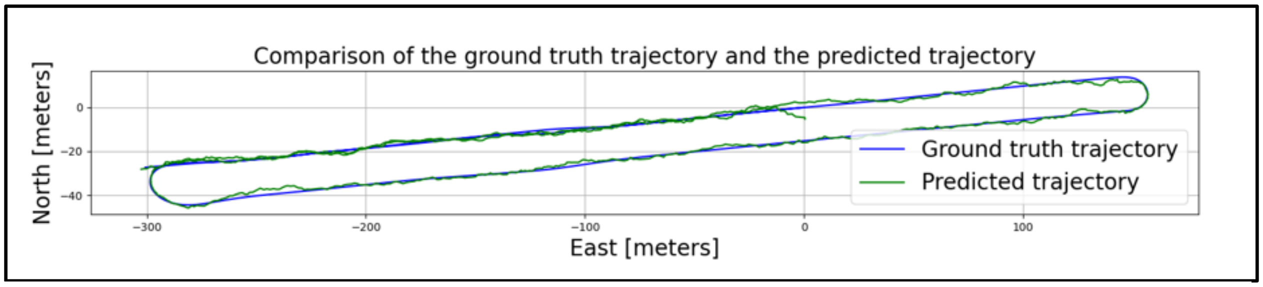


Figure 11 – Comparison of the ground truth trajectory and the filtered trajectory (EKF)

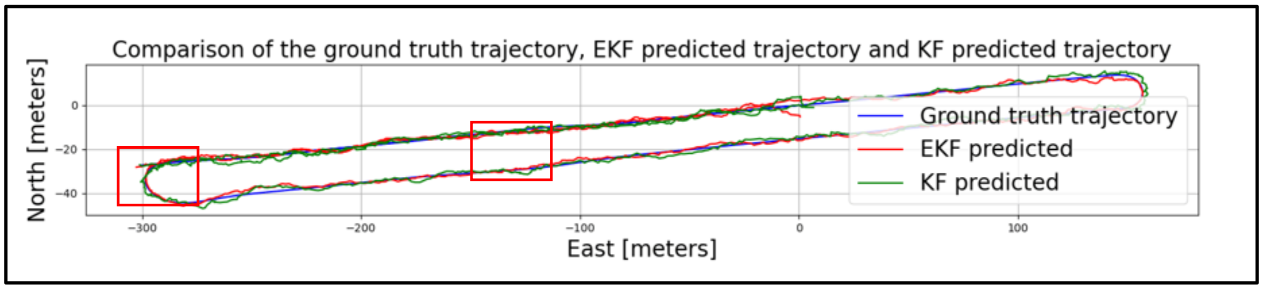
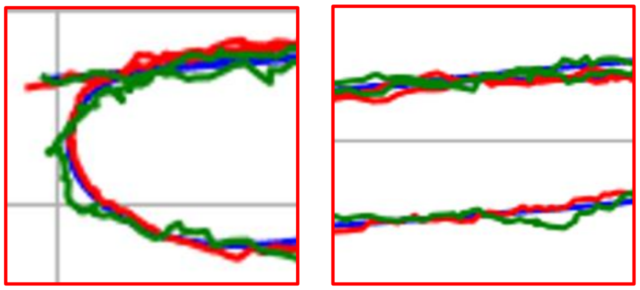


Figure 12 – Comparison of the ground truth trajectory, EKF predicted trajectory and KF predicted trajectory

Let's focus on the interesting parts.
As anticipated, the EKF demonstrates a more accurate and smoother path compared to the KF. In the turn, the EKF shows a clean and precise trajectory, closely following the ground truth. Similarly, in the middle of the track, the EKF maintains a smooth and accurate path. This illustrates the EKF's superior performance in maintaining accuracy and reducing noise throughout the entire trajectory.



We obtained good results for the predicted path, similar to the outcome of the Kalman Filter. Compared to the KF, we can see that the Extended Kalman Filter is closer to the ground truth during turns, and the EKF's trajectory is smoother than that of the KF. However, towards the end of the path, it appears that the nonlinear method manages to stay closer to the ground truth. Overall, considering the maxE and RMSE values, the EKF is the better choice.

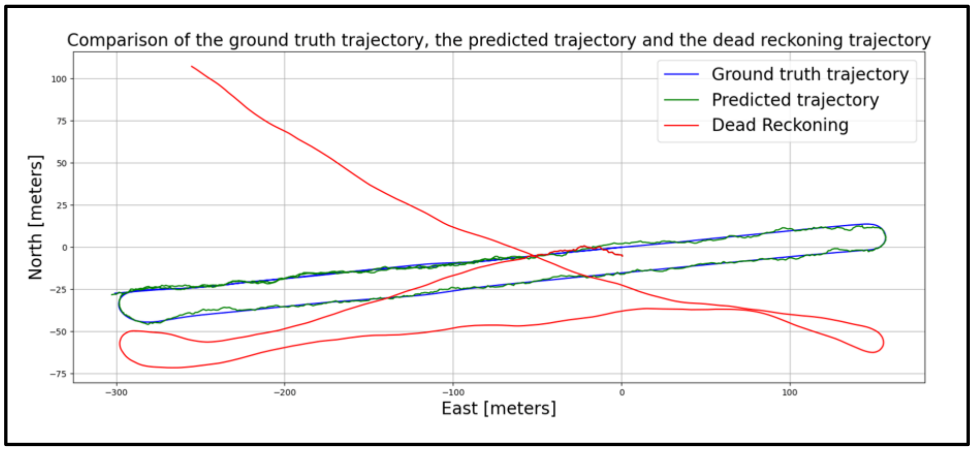


Figure 13 – Comparison of the ground truth trajectory and the dead reckoning

As the model here is non-linear and operates based on the v_f and w measurements, the resulting path closely resembles the ground truth trajectory. However, since the filter does not receive any direct information about the 2D position, it deviates from the correct path over time.

g. After analyzing the estimation results, it's essential to verify if we meet the uncertainty conditions. Specifically, we need to determine if the errors fall within the range of +/- sigma.

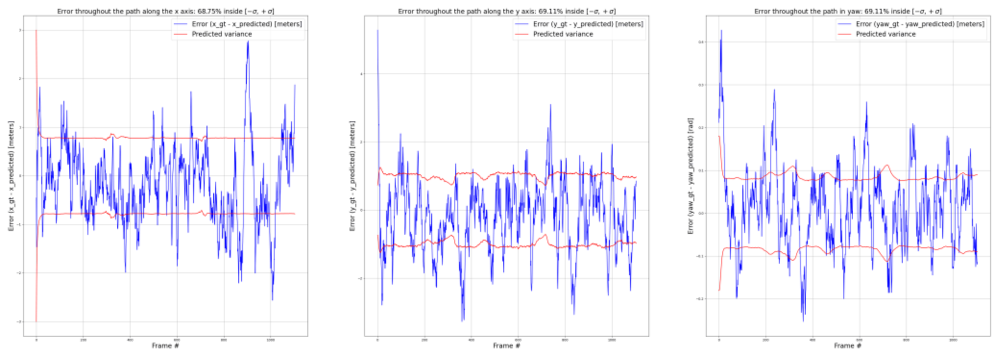


Figure 14 – Plot trajectory error

These percentages indicate that a significant portion of the errors fall within expected bounds. Specifically, approximately 68.75% of predictions in the X direction, 69.11% in the Y direction, and 69.11% in the yaw direction are within the typical range of ~68%, demonstrating reasonably accurate estimation performance without exceeding expected levels of uncertainty.

In conclusion, our report focused on implementing and evaluating regular and extended Kalman filters on the KITTI dataset, comparing their performance against ground truth data. We found that incorporating augmented state components notably improved directional accuracy, highlighting the benefits of using a non-linear filter.

Appendix

Attached is a directory named "Figures" containing all the images added to the report.

The corresponding animation for part A is labeled as " kf_predict_with_dead_reckoning".

The corresponding animation for part B is labeled as " EKF_predict_with_dead_reckoning".