

**Project 2**

Mapping and perception for an autonomous robot/ 0510-7591

By

Nimrod Curtis

Tel Aviv University  
April 2023

**Contents**

[1. Introduction 1](#_Toc135585942)

[2. Solution 2](#_Toc135585943)

[1. Kalman Filter 2](#_Toc135585944)

[2. Extended Kalman Filter 9](#_Toc135585945)

[3. EKF-SLAM 17](#_Toc135585946)

[3. Summary 27](#_Toc135585947)

**Figures**

[Figure 1 - Global trajectory from google maps 3](#_Toc135586019)

[Figure 2 - tarajectory in LLA coordinate system 4](#_Toc135586020)

[Figure 3 - trajectory in ENU coordinate system 4](#_Toc135586021)

[Figure 4 - GT vs. Noisy trajectory in local coordinate system 4](#_Toc135586022)

[Figure 5 - KF trajectory compared to GT and Noise 7](#_Toc135586023)

[Figure 6 - calibration process of k and sigma\_n w.r.t rmse 8](#_Toc135586024)

[Figure 7 - x & y Error (GT - predicted) 9](#_Toc135586025)

[Figure 8 - Yaw & Yaw change rate 10](#_Toc135586026)

[Figure 9 - Forward valocity per frame 11](#_Toc135586027)

[Figure 10 - Gaussian noised velocities (linear and angular) 12](#_Toc135586028)

[Figure 11 - EKF algorithm 14](#_Toc135586029)

[Figure 12 - trajectory of EKF estimation compared to the GT and noisy 15](#_Toc135586030)

[Figure 13 - EKF hyperparametrers calibration heatmap 15](#_Toc135586031)

[Figure 14 - x-y-theta errors 17](#_Toc135586032)

[Figure 15 - odometry model 18](#_Toc135586033)

[Figure 16 - GT trajectory 18](#_Toc135586034)

[Figure 17 - Noisy trajectory 19](#_Toc135586035)

[Figure 18 - GT vs predicted EKF-SLAM 22](#_Toc135586036)

[Figure 19 - x error 24](#_Toc135586037)

[Figure 20 - y error 24](#_Toc135586038)

[Figure 21 - theta error 25](#_Toc135586039)

[Figure 22 - top->bottom: landmark 1 x&y errors, landmark2 x&y errors 27](#_Toc135586040)

# Introduction

This is the second project for a mapping and sensing course on autonomous systems, which covers three primary subjects:

* Kalman Filter
* The Extended Kalman Filter (EKF)
* SLAM using EKF

During the project we used the recorded data from KITTI dataset.

This is the second project for a mapping and sensing course on autonomous systems, which covers three primary subjects: the Kalman Filter, the Extended Kalman Filter (EKF), and SLAM using EKF. In this project, we utilized recorded data from the KITTI dataset as our experimental data. Our initial task was to apply the Kalman filter to mitigate the noise in GPS data by employing the constant velocity model. Through this process, we aimed to estimate the vehicle's 2D pose accurately. In the second part of the project, we explored the application of the Extended Kalman Filter, which is capable of handling nonlinear models. By incorporating additional sensor measurements such as yaw angle, yaw rate, and forward velocity, we aimed to improve the accuracy of our pose estimation. Lastly, we delved into the field of SLAM (Simultaneous Localization and Mapping) using the Extended Kalman Filter, which allowed us to simultaneously estimate the vehicle's pose while mapping the environment.

The record number: 2011\_09\_30\_drive\_0020

The recorded footage captures a drive through a suburban neighborhood in Karlsruhe Germany. Throughout the recording, the vehicle navigates the streets, including a turn onto an inner pedestrian street within the neighborhood. At each turn, the vehicle decelerates slightly before accelerating again, eventually exiting onto a main road.

**Important Notes:**

##### To execute the code, please follow these instructions in the 'run' function: uncomment the relevant question and comment out the other two questions by adding the comment symbol. This is important to ensure that any values from previous questions do not carry over to the next one.

##### I added a calibration file – calibration.py which execute the Kalman filter params grid search calibration using an optimization library called 'hyperopt'.

Please '**pip install hyperopt**' before.

##### Because of the calibration process is done by hyperopt function it can get different values then I got on mu runs. So it outputs can be better or worth, but still will be good enough for the requirements.

# Solution

## Kalman Filter

#### Install the dataset

As well described the record is taking place at the city of Karlsruhe inside a small neighborhood.

תמונה שמכילה טקסט, צילום מסך, קו, מפה

התיאור נוצר באופן אוטומטי

Figure - Global trajectory from google maps

The vehicle begins its journey at the blue flag depicted in the figure above. It navigates through an inner pedestrian street within the neighborhood, surrounded by private houses and parking lots, until it reaches the red flag marking the end point. To get a visual representation of the drive, you can watch the video located in the "Results/animations/kitti\_0020\_video" folder. This video displays the left camera frames and the lidar data, providing additional insights into the driving experience.

#### GPS trajectory extraction

The OXTS sensor data was extracted and timestamps were converted to seconds difference between frames. This data will be used later for ground truth reference.

#### Transform trajectory to local coordinate system

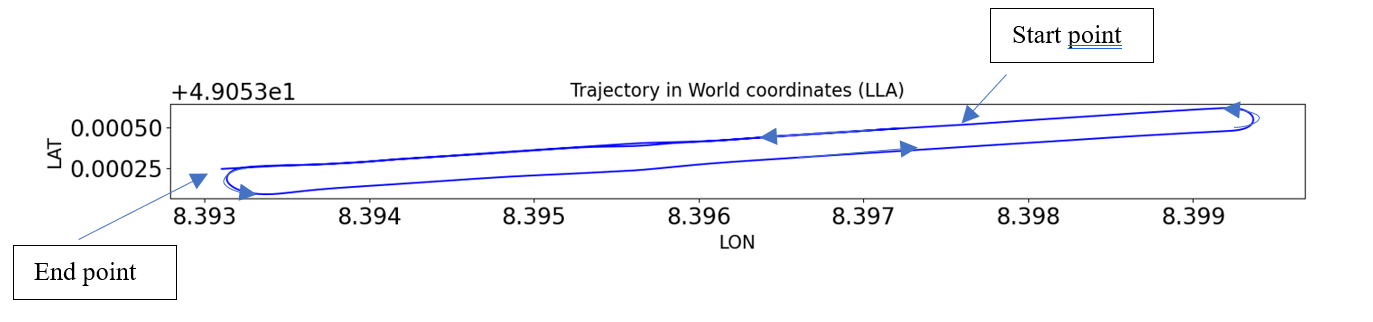


Figure - tarajectory in LLA coordinate system

In the plot above the arrows indicating the driving direction.

The GPS trajectory was transformed from LLA global coordinate system to ENU local coordinate system.

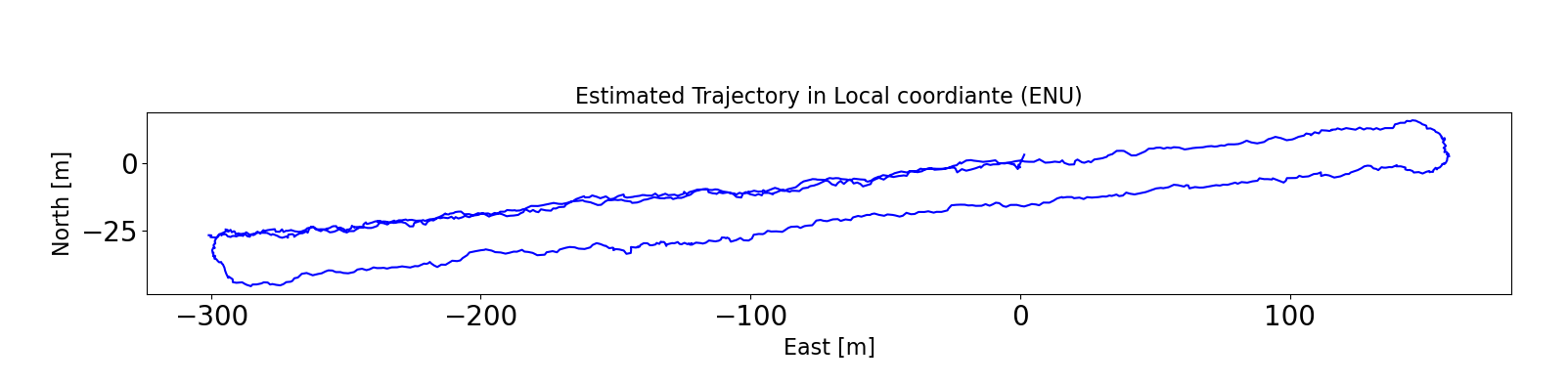


Figure - trajectory in ENU coordinate system

#### Adding noise to data

Random errors, in the form of Gaussian noise, were added to the data. The standard deviation of the noise was 3 meters. The resulting noisy positions was then used as input observations to the Kalman filter.

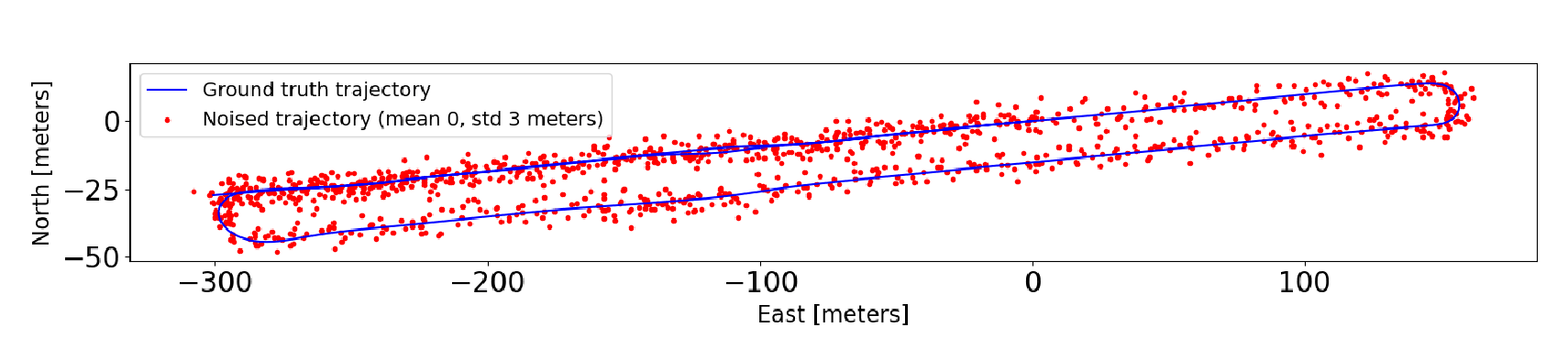


Figure - GT vs. Noisy trajectory in local coordinate system

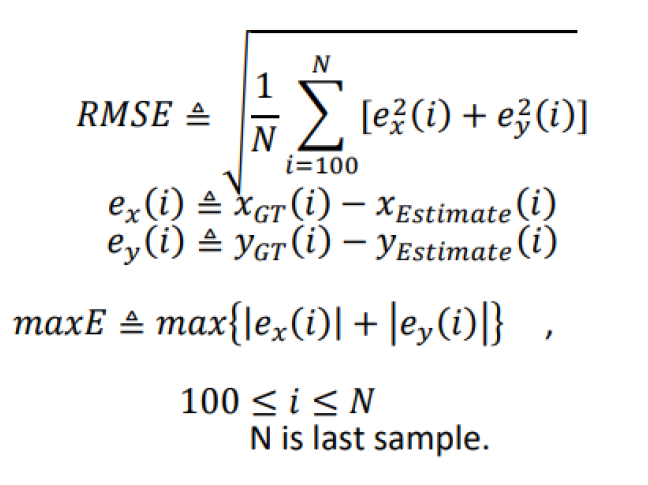
The above plot displays the dispersion of the red dots (representing noisy data) around the blue trajectory (representing the true/actual trajectory). Due to the considerable distances covered by the vehicle compared to the disturbance magnitude (3 meters), there are instances where the red dots can be seen as overlap each other and exhibit relatively low resolution.

#### Apply Kalman Filter and Calibration

In this section Kalman Filtered was applied to estimate the vehicle 2D (local coordinate [x,y] positions) pose based on constant velocity model.

The Kalman filter predictions were based on the **constant velocity model –** no sensor control inputs.

Next, the Kalman filter's hyperparameters were adjusted in a calibration process that relied on two criteria: the root-mean-square error (RMSE) and maximum error (maxE) as follow:



The goal was to minimize the the rmse while maxE<7 [meter].

##### Initial conditions

The initial estimated state was defined as:

Supposed the estimation starts with the first noisy GPS observation.

The initial covariance matrix was defined as:

Where the variances are the measurements noise (known) and the values were set to a large number to increase the observations impact at the beginning of the process instead of the estimation.

Hyperparameter k was set initially to **3** and **later was tuned to optimal value for the filter (calib process will be described)**.

##### A / B / C Matrixes

Matrix B is the control coefficients. Because we have no control inputs on this section, it was set to 0.

Matrix A is the state vector translation matrix. Therefore it was defined according to the constant velocity model:

Where is the time difference between the current and the previous frames.

Matrix C is the observation matrix.

Suppose the observation is of the form:

And the observation equation is:

Then C can be defined as:

##### Observation Covariance

The covariance matrix Q was set based on the known measurements GPS noise:

##### Process Covariance

The covariance matrix R was set based on the constant velocity method as follow:

Where is the estimated velocity noise which initially was set to **1** and **later was calibrated to optimal value for the filter**.

The main routine of the Kalman filter involves initializing the state estimate and covariance matrix, predicting the current state based on the previous estimate and system dynamics, incorporating measurements to update the estimate, and repeating these steps iteratively. It aims to accurately estimate the true state of a system by combining predictions and measurements while considering uncertainties.

* Now the main routine of the Kalman filter can be presented:

The filter is based on two main steps:

* + - 1. Prediction step
      2. Correction step
* The prediction steps equations:
* Then the Kalman gain matrix can be calculated and present the relation between the process covariances and the impact of the observation noise to correct better the estimations values.
* Finally the estimations can be corrected using the Kalman gain and the correction step equations:

**Calibration process:**

The calibration process involves iterating through a loop where each iteration involves modifying the and values and calculating the RMSE and MAXE. If the obtained RMSE is lower than the previously lowest RMSE, it is saved, and further adjustments are made to the hyperparameters based on the values that resulted in the lowest RMSE. Finally, a test is conducted to ensure that the MAXE condition is met. This process, resembling a grid search, consists of approximately 30 iterations, during which the Kalman Filter is reassessed. This iterative procedure closely resembles the calibration process typically conducted for a realistic filter calibration scenario.

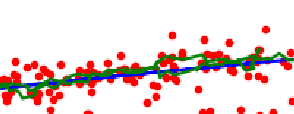
Search space:

**Best parameters:**

**Results & analysis section is based on this params.**

#### Results analysis:

##### Ground-truth and estimated results

תמונה שמכילה צבעוני, אדום

התיאור נוצר באופן אוטומטיThe graphs of the ground-truth and the tuned Kalman filtered estimation trajectory can be seen below:

start

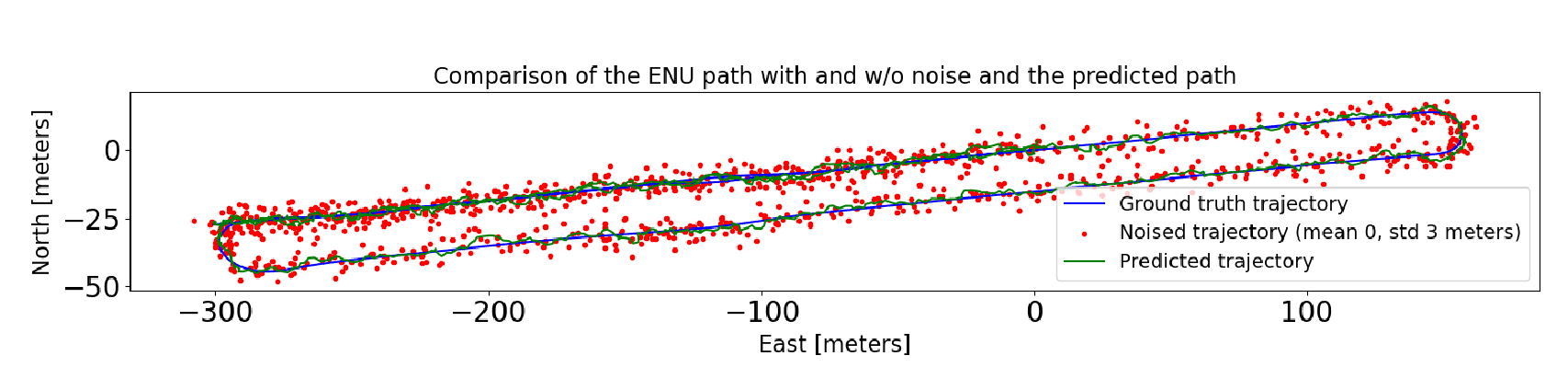
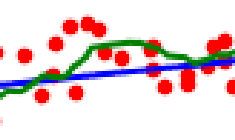


Figure - KF trajectory compared to GT and Noise

The graph above illustrates the estimated path of the Kalman Filter (KF) based on the best obtained values. Notably, the most significant errors appear to occur on the start point, during turns and when consecutive observations exhibit noisy deviations in the same direction.

Supposed the initial predict is the first noisy observation it makes sense that the error at first would be large, but after a few measurements it seems that the estimator manages to converge and deal with the noisy information.

The classic Kalman Filter based on a constant velocity model assumes linear and constant motion, **which may not accurately capture the complex and nonlinear dynamics** of real-world scenarios. This can lead to suboptimal performance and significant errors in estimating the system's state during tracking or prediction.

In turns, where the vehicle typically slows down to change direction, the classic Kalman Filter may encounter difficulties. Specifically, after driving in a straight line for a prolonged period, the estimator may heavily rely on its prediction and, assuming a constant speed, deviate during the slower turn.

In overall the KF managed quite well to estimate the position of the vehicle given the noise in the measurements.

##### Minimum values achieved

The minimum values achived for RMSE and maxE are as follow:

Calibration analysis:

תמונה שמכילה צילום מסך, תרשים, צבעוני, טקסט

התיאור נוצר באופן אוטומטי

Figure - calibration process of k and sigma\_n w.r.t rmse

For better analysis of the calibration process a heat-map was plotted. The axes - and the color represent the rmse error achieved – the more brighter the less rmse value (better). Each dot represent a single trial.

From this plot we can see the area of a good values of the hyperparameters.

##### Kalman filter performance:

I believe the performance was better explained in the last and next sections.

##### Final animations

An animation was generated and saved in Results/animations/ q1\_KF\_traj\_ani to show the trajectory of the vehicle using the tuned Kalman filter.

A second animation was created - Results/animations/ q1\_KF\_cov\_ani to display the trajectory along with the error ellipse. The size and rotation of the ellipse was calculated from the covariance of each frame. It can be seen that the ellipse is the biggest on the first frame, then while the KF trajectory is pretty coverged the ellipse become smaller. Due to the resolution of the plot it is quite difficult to recognize the ellipse after the first frames (but it is there!).

Additionally, another animation was created - Results/animations/ q1\_KF\_cov\_drck\_ani to demonstrate the dead-reckoning mode, which involves a Kalman gain of 0 and relies solely on prediction.

##### The animation's ellipse represents the uncertainty area where the vehicle is located. Both the estimated and real routes can be seen within this ellipse. **It is worth noting** that the ellipse is actually a circle. This is because the assumption is that there is no correlation between the X-axis and Y-axis frequencies. As a result, the ellipse lacks any directional bias and its size is determined solely by the variance, which is equal in both the X and Y axes. Thus, the radius of the ellipse is the same in both width and length, making it a circle.

In the dead-reckoning it can be seen that the uncertainty-ellipse is growing continuously right after the K-gain is set to zero. When the filter is based only on the predictions it means that actually there is no filter and the uncertainty is increasing from frame to frame.

##### Errors

The Kalman filter performance is reflected in the location errors its achieved, this can be seen in the errors graph in x and y axis.

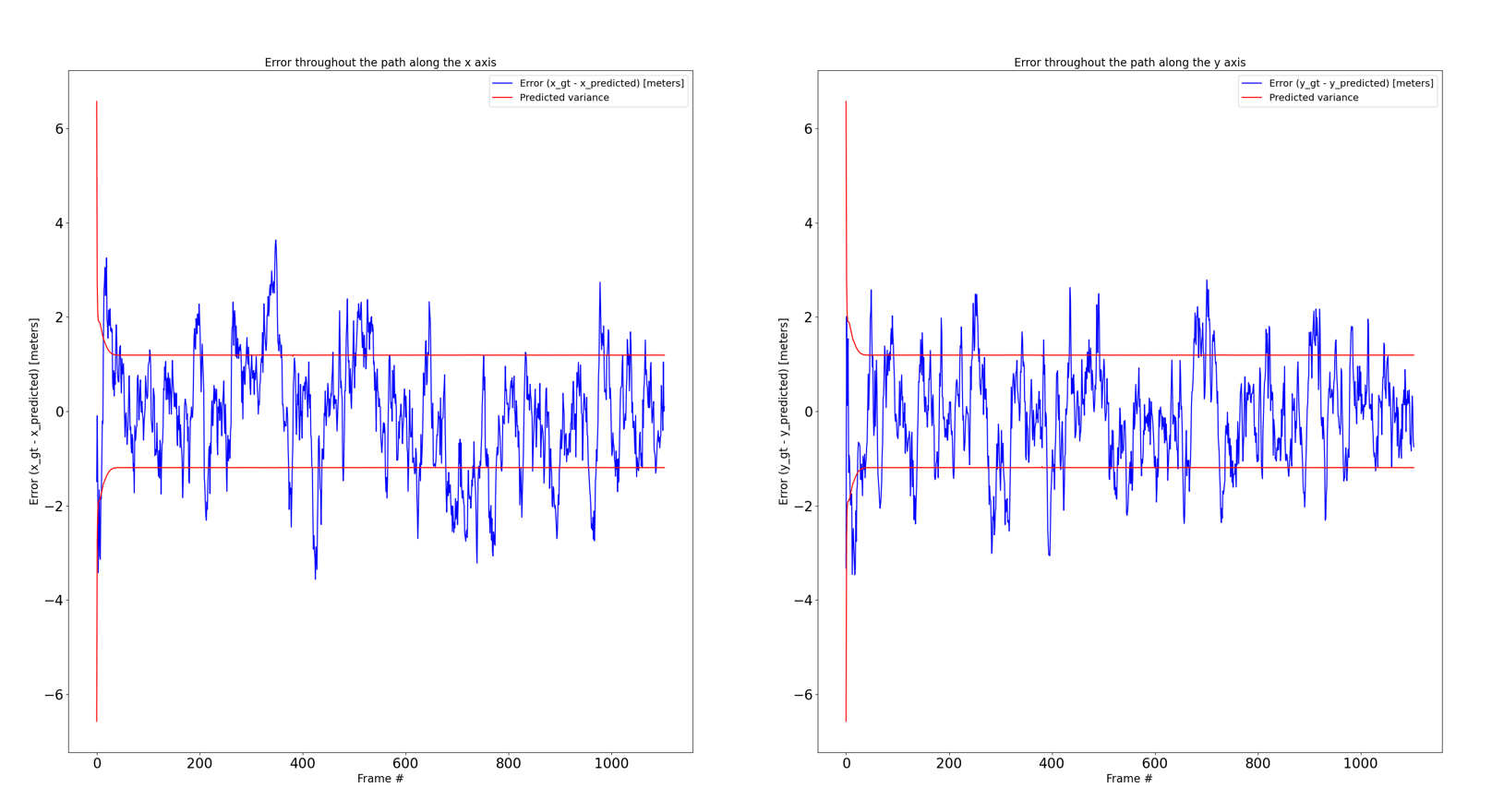


Figure - x & y Error (GT - predicted)

From the plots above the initial variances is converging to a constant value of approximately 1.5 which bounds ~68% of the error.

On this plots we can see again that the most significant errors occurred on the beginning of the trajectory (which is typical behavior) and on turns.

## Extended Kalman Filter

#### Install dataset

#### Same sequence from last section was use for the EKF.

#### Data extraction

#### In this section the EKF was based on the **non-linear velocity model**, so data of – heading, forward velocity, and angular rate were extracted in addition to the GPS and timestamps readings.

#### Trajectory and velocities

For the GPS (LLA) and local trajectory (ENU) see Figure 1,Figure 2.

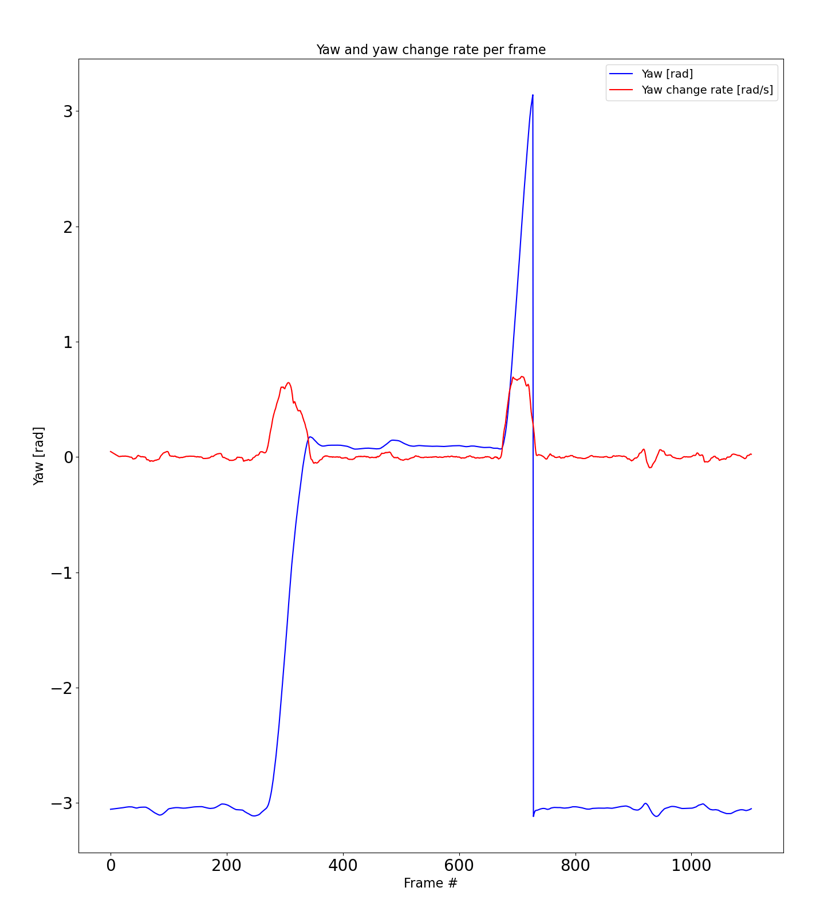


Figure - Yaw & Yaw change rate

By observing the graph above, we can gather information about the car's heading values along its trajectory. Notably, we can observe the variation of the yaw angle during turns, where the yaw change rate initially increases and then decreases as the car returns to driving straight.

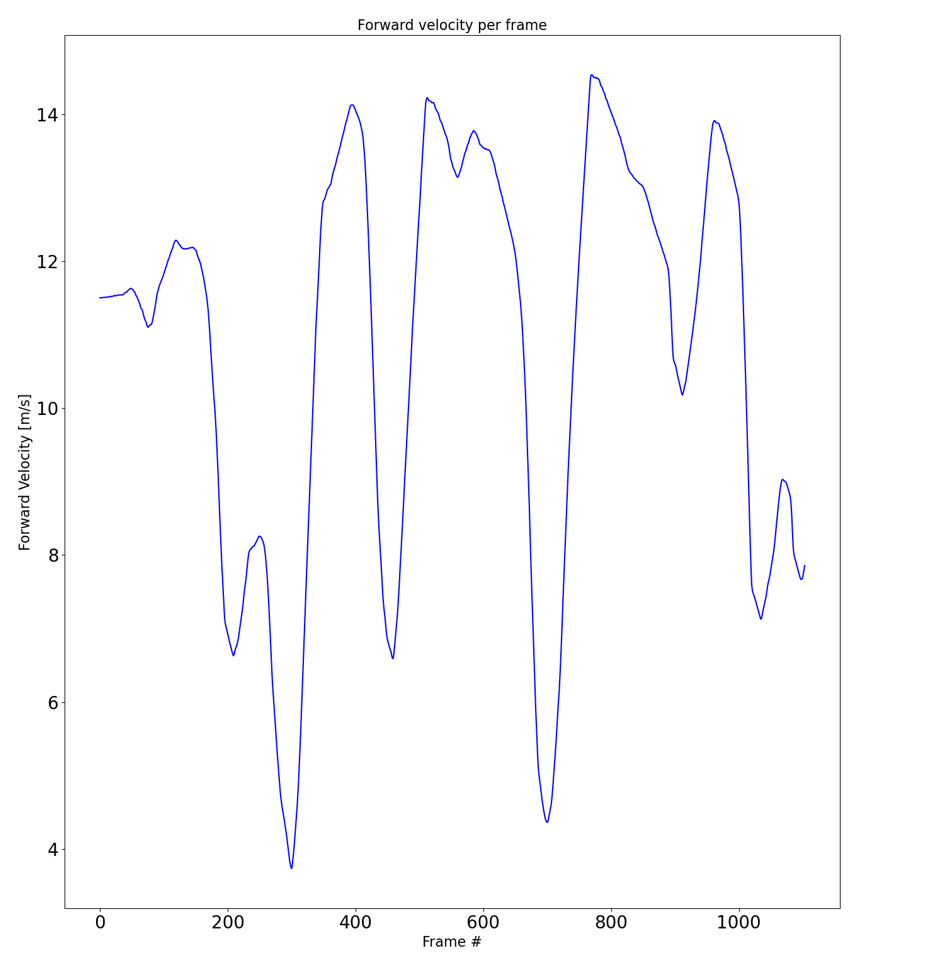


Figure - Forward valocity per frame

From the plot above, we can gather insights into the linear velocity of the vehicle along its trajectory. It is evident that the linear speed decreases during turns and then increases again when the vehicle is driving straight.

#### Adding position noise

Same noise was added to the ground truth positions. See section ‎d of KF section.

#### Apply EKF

In this section Extended Kalman Filter was applied to estimate the vehicle 2D (local coordinate [x,y] positions) pose based on non-linear velocity model.

The Kalman filter predictions were based on the **non-linear velocity model –** non linear state translation matrix with noisy velocity control inputs.

Next, the Extended Kalman filter's hyperparameters were adjusted in a calibration process that relied on the same criteria as the last section.

#### Add noise to velocity

A gaussian noise was added both to the linear velocity and the angular velocity (.

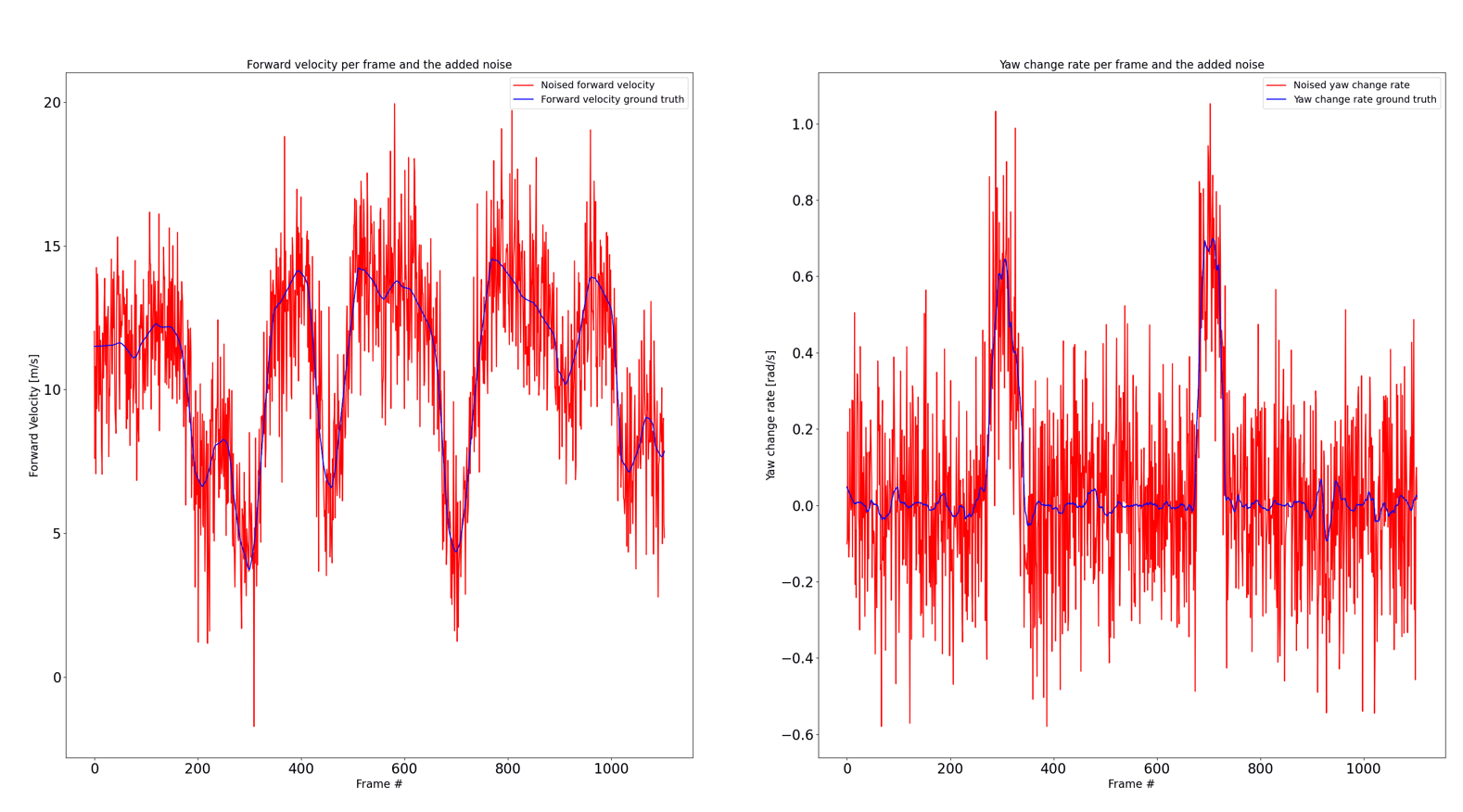


Figure - Gaussian noised velocities (linear and angular)

#### EKF calibration

The goal was to minimize the the rmse while maxE<5 [meter]. For the rmse and maxE equation see section e of KF section.

##### Initial conditions

The initial estimated state was defined as:

Supposed the estimation starts with the first noisy GPS and yaw observation.

The initial covariance matrix was defined as:

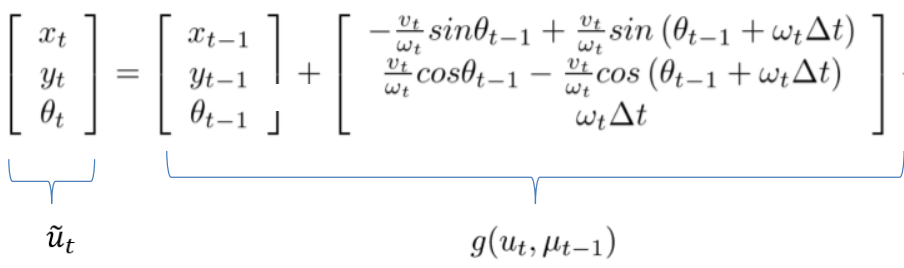
Where the variances are the measurements noise (known).

Hyperparameter k was set initially to **3** and **later was tuned to optimal value for the filter (calib process will be described)**.

Hyperparameter was set initially to **0.5** and **later was tuned to optimal value for the filter (calib process will be described)**.

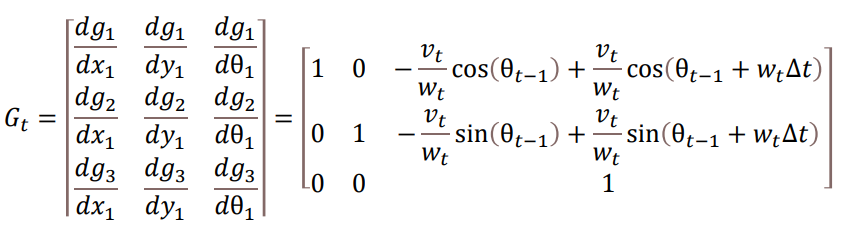
##### Jacobians G / V / C Matrixes

The non-linear function of the velocity model prediction is as follows:



Matrix G is the jacobian of the non-linear velocity model – function g - w.r.t the state vector.

So the matrix was set to:



Matrix V is the jacobian matrix of g w.r.t the velocities, so it was set to:

תמונה שמכילה טקסט, גופן, קו

התיאור נוצר באופן אוטומטי

Matrix C = H is the observation matrix.

Suppose the observation is of the form:

And the observation equation is:

Then H can be defined as:

##### Process and Observation Covariances

The covariance matrix Q was set based on the known measurements GPS noise.

The covariance matrix R was set based on the known noise of the velocities.

Another noise matrix was set to tune the filter, and prevent it to be extremely optimistic. This matrix was calibrated to add tiny noise values to the covariance matrix .

The values of = and **were calibrated** for better performance of the filter.

The main routine of the Extended Kalman Filter (EKF) involves initializing the state estimate and covariance matrix, predicting the current state based on the nonlinear system dynamics, updating the estimate using measurements by linearizing the system and measurement functions, and repeating these steps iteratively. The EKF extends the Kalman Filter to handle nonlinear systems by linearizing functions using Jacobian matrices, enabling better state estimation in such scenarios.

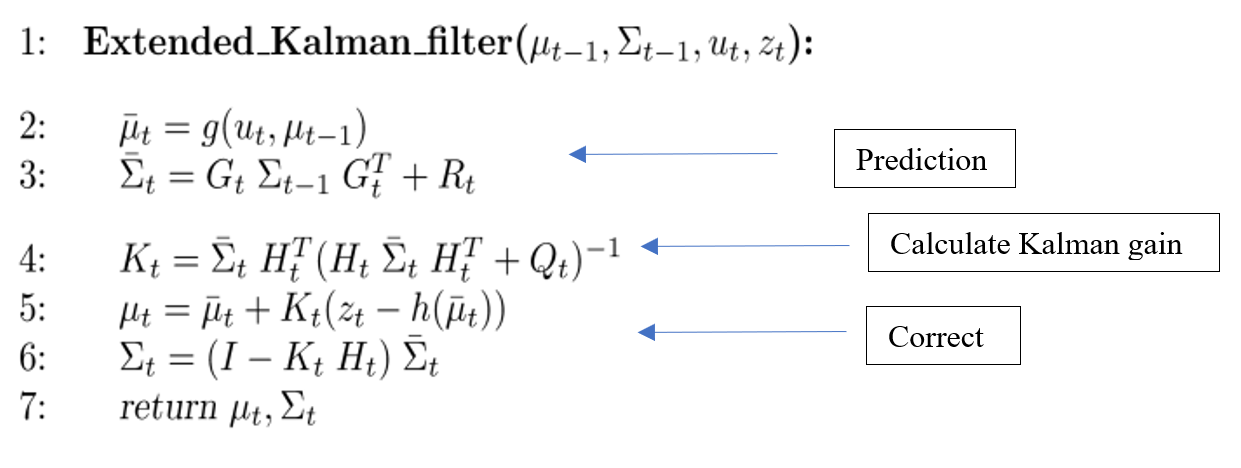
* Now the main routine can be explained mathematically using the pseudo algorithm :

Figure - EKF algorithm

**Calibration process:**

The calibration process was done in the same manner as the same section. The hyperparameters tuned were for achieving the goal.

Now the number of trails was 100 due to increasing in the number of hyperparams and combinations.

Search space:

**Best parameters:**

From first impressions it can be seen that the additional noise values that have been added to the covariance and (in the matrix) were converge to the bottom boundary of the search space, indicating that maybe the additional noise doesn't has significant effect.

**Results & analysis section is based on this params.**

#### Results and analysis

##### Ground truth and estimated results

תמונה שמכילה טקסט, צילום מסך, קו, גופן

התיאור נוצר באופן אוטומטי

Figure - trajectory of EKF estimation compared to the GT and noisy

The graph above depicts the behavior of the filter. It is evident that, on the whole, the filter closely tracks the actual vehicle route. Initially, due to high noise levels, there is a significant deviation, but the filter manages to stabilize over time. The filter accounting for the nonlinearity of the motion performs better in tracking the vehicle's turns, as indicated by the blue arrow in the graph.

**Calibration analysis:**

תמונה שמכילה תרשים, טקסט, צילום מסך, קו

התיאור נוצר באופן אוטומטי

Figure - EKF hyperparametrers calibration heatmap

Similar to the previous part, a heat map was extracted that shows the optimal area of the hyperparams. The variable is not considered because it has the smallest effect. And the graph extraction was done based on the three main params.

##### EKF performance

I believe that the performance was well explained last sections and on the next ones.

##### Minimum values

The minimum values achieved are as follows:

The EKF demonstrates superior performance compared to the KF, successfully meeting the requirement of maxE<5 while achieving a lower RMSE value.

The main advantage of the Extended Kalman Filter over the classic Kalman Filter is its ability to handle nonlinear system dynamics by linearizing them using Jacobian matrices, leading to more accurate state estimation in nonlinear scenarios such as in our scenario of car maneuvering. So the results achieved are make sense.

And yet it should be taken with limited warranty due to the number of calibration attempts (100 compared to 30) .

##### Final animations

An animation was generated and saved in Results/animations/ q2\_KF\_traj\_ani to show the trajectory of the vehicle using the tuned Extended Kalman filter. In this animation we can clearly see the improvement in tracking after curve as a result of the way the filter works.

A second animation was created - Results/animations/ q2\_KF\_cov\_ani to display the trajectory along with the error ellipse. The size and rotation of the ellipse was calculated from the covariance of each frame. It can be seen that the ellipse is the biggest on the first frame, then while the KF trajectory is pretty coverged the ellipse become smaller. Due to the resolution of the plot it is quite difficult to recognize the ellipse after the first frames (but it is there!).

Additionally, another animation was created - Results/animations/ q2\_KF\_cov\_drck\_ani to demonstrate the dead-reckoning mode, which involves a Kalman gain of 0 and relies solely on prediction.

In the dead-reckoning it can be seen that the uncertainty-ellipse is growing continuously right after the K-gain is set to zero. When the filter is based only on the predictions it means that actually there is no filter and the uncertainty is increasing from frame to frame.

Unlike in the previous part, this time there is a correlation between the movements in the axes and therefore the different directions and sizes of the radii of the ellipse. This can be seen clearly in the track of the dead-reckoning and less clearly on the regular animation because of the resolution issue.

##### Errors

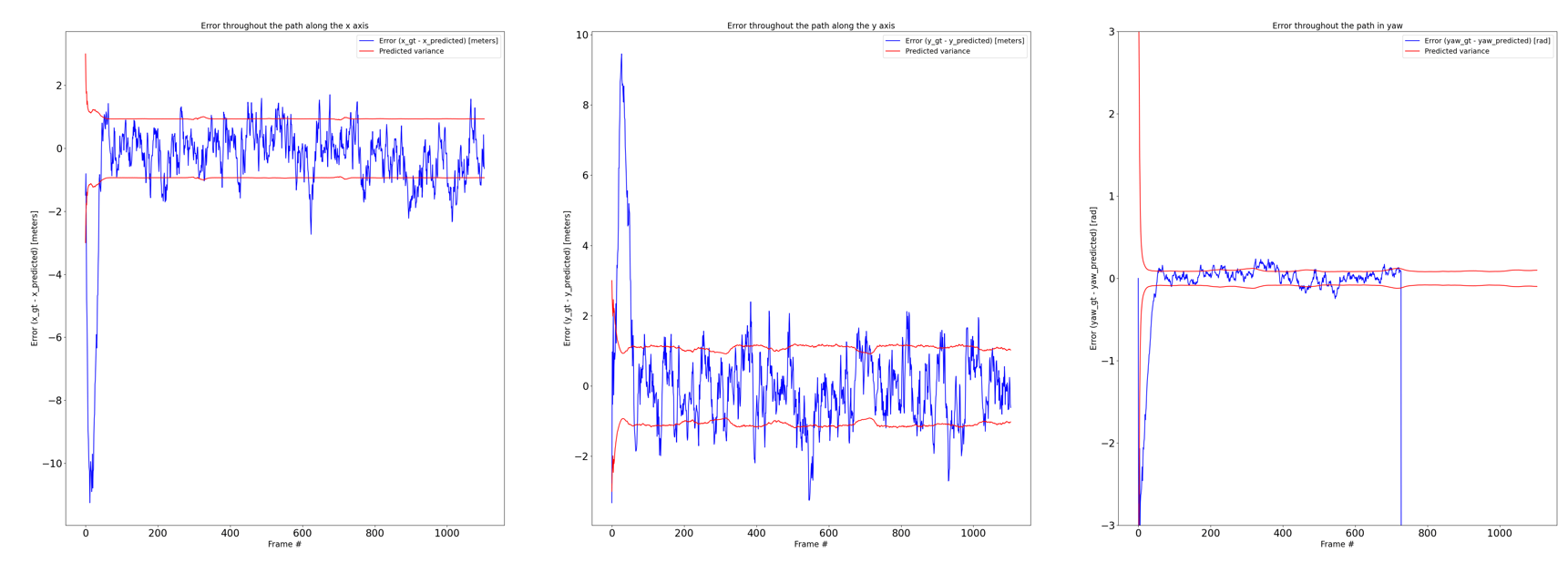


Figure - x-y-theta errors

From the figure above we can clearly see the big initial error of all the three state variables.

But immediately after that there is a drastic decrease in the error and stabilization within the sleeve of the standard deviation which 'should' bound ~68% of the error.

On the right plot of the yaw's error it can be see that from frame ~730 there is increasing of the absolute value of the error. This phenomenon can be explained by noting that during the last turn, the vehicle rapidly changes direction from an yaw angle of approximately 3 radians to a negative angle of around -3 radians (see Figure 7). During this abrupt transition, the filter, which initially had an angle value of approximately 2.7 radians, fails to update accurately due to the sudden change in the angular rate. Because right after that the angular rate stabilizes, as a result, the filter's predicted value remains in positive territory. However, it is important to highlight that the actual error is relatively small. To present the error more accurately, a minor adjustment of normalization (which was not performed in this case) in the error calculation would be necessary.

## EKF-SLAM

SLAM stands for Simultaneous Localization and Mapping. It's a problem in robotics where a robot or a vehicle needs to simultaneously build a map of an unknown environment and determine its own position within that map.

EKF-SLAM, is an algorithm used to solve the SLAM problem. It combines sensor measurements and motion models to estimate the robot's pose and the map of the environment. The algorithm uses the Extended Kalman Filter to iteratively update and refine these estimates based on incoming sensor data.

In this section an EKF-SLAM algorithm was implemented for localization and mapping of a vehicle in an area with 9 landmarks. The vehicle control inputs are based on the odometry data, and the observation inputs are based on the relative range and bearing from the landmarks.

#### Install data

Odometry data and sensor data were loaded.

Odometry data – [rotation1, translation, rotation2] per frame.

sensor data – [idx(which landmark),range,bearing] per frame.

#### Odometry model and GT trajectory

Based on the odometry data we can calculate the ground truth trajectory using the odometry model as below:

תמונה שמכילה טקסט, גופן, כתב יד, קו

התיאור נוצר באופן אוטומטי

Figure - odometry model

תמונה שמכילה טקסט, גופן, כתב יד, קו

התיאור נוצר באופן אוטומטי

תמונה שמכילה טקסט, תרשים, קו, עלילה

התיאור נוצר באופן אוטומטי

Figure - GT trajectory

#### Adding gaussian noise to the motion model

Gaussian noise of was added to the model.

תמונה שמכילה טקסט, תרשים, קו, עלילה

התיאור נוצר באופן אוטומטי

Figure - Noisy trajectory

We can see that indeed the noisy trajectory has translation and rotation relate to the GT.

#### Apply EKF-SLAM & Initial conditions

In this section Extended Kalman Filter - SLAM was applied to estimate the vehicle and landmarks 2D position based on the sensor and odometry data.

The Kalman filter predictions were based on the **non-linear odometry model –** non linear state translation matrix with noisy translation and rotations control inputs.

Next, the Extended Kalman filter's hyperparameters were adjusted in a calibration process that relied on the same criteria as the last section.

Therefore, the goal was to minimize the RMSE but now the maxE required to be less than 1.5 meters and the minimum index was 20.

Initial conditions:

In the SLAM algorithm, a key modification of the Kalman filter is incorporating the 2D positions of landmarks into the state vector, along with their associated uncertainty in the covariance matrix. As a result, the sizes of the state vector and covariance matrix vary depending on the number of landmarks in the environment.

So in our case the initial state vector and covariance matrix are:

Where are the pose variances and were tuned to get a good results.

And is the variance of the landmark i and initially need to be set as high value such as 5 meters in our case scales.

#### Prediction step

As said before the prediction step is based on the odometry motion model.

תמונה שמכילה טקסט, גופן, כתב יד, קו

התיאור נוצר באופן אוטומטי

תמונה שמכילה טקסט, גופן, כתב יד, לבן

התיאור נוצר באופן אוטומטי

Where matrix F is an adaptation matrix for sure that only pose is updating while handling the dimensions.

#### Jacobian

The jacobian matrix is the jacobian matrix w.r.t the pose vector so it can be write as:

תמונה שמכילה טקסט, גופן, כתב יד, לבן

התיאור נוצר באופן אוטומטי

Then we can construct the full matrix as:

תמונה שמכילה טקסט, גופן, תרשים, עיצוב

התיאור נוצר באופן אוטומטי

#### Jacobian

V is the jacobian matrix w.r.t the odometry so it can be write as:

תמונה שמכילה טקסט, גופן, לבן, קו

התיאור נוצר באופן אוטומטי

So now we can construct the full matrix as :

#### Update step

The correction step is based on the algorithm below:

תמונה שמכילה טקסט, גופן, צילום מסך, כתב יד

התיאור נוצר באופן אוטומטי

Where is the observation covariance matrix composed from the known variances of the range and bearing noises.

תמונה שמכילה טקסט, כתב יד, תרשים, גופן

התיאור נוצר באופן אוטומטי

Where is an adaptation matrix which describe which landmark is observed.

And matrix is the transformation matrix from the specific landmark observation to transform the r and phi to x,y positions.

In our case matrix of every landmark observed was stack into general matrix . The Kalman gain was computed so we could correct the state and its covariance depend on it.

#### Analyze results

##### Trajectory of EKF-SLAM

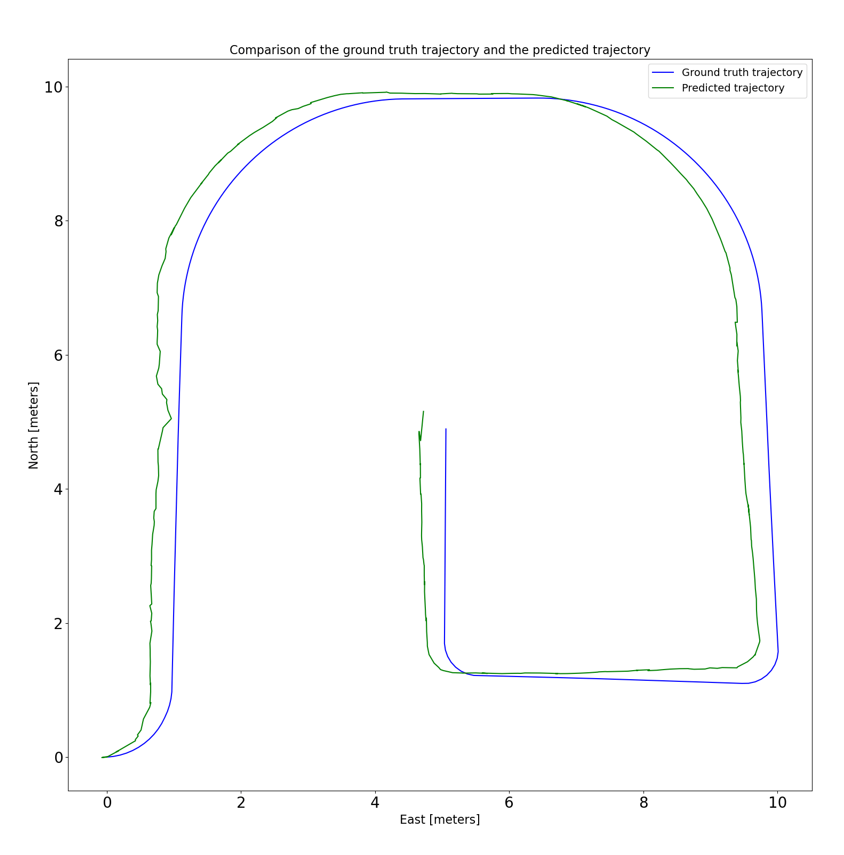


Figure - GT vs predicted EKF-SLAM

An animation depicting the SLAM scenario was generated and can be found at "Results/animations/q3\_EKF\_SLAM\_traj-ani". The animation shows that, overall, the filter is able to track the ground truth path successfully. However, it is evident that the filter's predictions are consistently ahead of the actual ground truth path. This behavior may be attributed to an offset in the landmark measurements and an underestimation of the uncertainties associated with these measurements. Consequently, the filter exhibits overconfidence in the observation sensor.

One potential improvement to enhance the tracking performance could be to slightly increase the initial covariance of observations and the noise associated with the observations.

Throughout the algorithm's execution, you can observe an ellipse surrounding both the vehicle and landmarks. This ellipse represents the area of uncertainty, which is influenced by the covariance matrix. Particularly in the initial frames, when a new observation is received and the covariance is updated, the uncertainty and subsequently the size of the ellipse are small.

When the filter stabilizes, the updates get smaller and smaller, so the difference in the size of the error ellipse is no longer visible.

In EKF-SLAM, when there are more observations of more landmarks, it generally leads to better corrections. The additional observations provide more information to update and refine the estimates of the robot's pose and the map. With more observations, the algorithm can better constrain the uncertainties and improve the accuracy of the estimated states. In our example, the phenomenon is not easily discernible due to the inherent measurement deviations in the landmarks. However, it is noticeable that when there are observations from numerous landmarks, their position points exhibit greater stability within the same frames.

RMSE and maxE found :

For

After experimenting with various parameter values, we identified certain values that resulted in good results for RMSE and maxE. The presence of offset in the landmarks justifies that the maxE primarily arises from the differences in the turns.

##### Pose errors

תמונה שמכילה טקסט, קו, תרשים, עלילה

התיאור נוצר באופן אוטומטי

Figure - x error

From the depicted graph, it is evident that the average error on the X-axis does not align around 0 as initially expected; rather, it tends to hover around 0.4. This deviation primarily stems from the estimated trajectory's main discrepancy in the X-axis. Notably, the error remains bounded within the decreasing uncertainty as the algorithm progresses with updates.

תמונה שמכילה טקסט, תרשים, קו, עלילה

התיאור נוצר באופן אוטומטי

Figure - y error

Similar to the previous graph, in this graph as well, the diminishing uncertainty can be observed with each frame. However, as anticipated, the error is centered around 0, particularly in sections where the track follows a straight path and the filter seamlessly integrates with it. In these regions, the error appears to exhibit a continuous pattern around the 0 mark.

תמונה שמכילה טקסט, קו, תרשים, מלבן

התיאור נוצר באופן אוטומטי

Figure - theta error

Here, too, the dispersion of the error around the zero and the standard deviation is seen in decay until it stabilizes in the sleeve approximately bounding the error.

##### Landmarks errors

Furthermore, two landmarks' estimated positions were captured during the execution, and their errors were calculated in both axes.

Lanmarks indexes: landmark 1 = ind 3, landmark 2 = ind 8

תמונה שמכילה טקסט, קו, תרשים, עלילה

התיאור נוצר באופן אוטומטי

תמונה שמכילה טקסט, קו, תרשים, עלילה

התיאור נוצר באופן אוטומטי

תמונה שמכילה טקסט, קו, תרשים, עלילה

התיאור נוצר באופן אוטומטי

תמונה שמכילה טקסט, קו, תרשים, צילום מסך

התיאור נוצר באופן אוטומטי

Figure - top->bottom: landmark 1 x&y errors, landmark2 x&y errors

The deviation in the x-axis is evident for both landmarks. Each landmark undergoes a process of updating the standard deviation after every observation and correction. It can be observed that when there is no observation of the Kian landmark, there is an update in the standard deviation. As a result, the graph of the standard deviation exhibits a staircase-like pattern, with a descending step occurring with each correction. Consequently, in the case of the second landmark, the decrease in standard deviation is noticeable only from the 30th frame, which corresponds to when the landmark was first acquired by the sensor.

# Summary

In this project three main topics are covered: Kalman Filter, Extended Kalman Filter, and EKF-SLAM. For the Kalman Filter, we were required to download a dataset from the KITTI dataset, extract GPS trajectory, transform it to local coordinates, add Gaussian noise, and apply the filter to estimate the vehicle's 2D pose. The goal was to calibrate the filter and analyze its performance. Similar steps are followed for the EKF, where additional sensor data such as yaw angle and forward velocity are used, and noise is added to the measurements. During the calibration process, we observed the impact of various parameters and identified the optimal values that yielded the desired criteria. We observed a distinction in the functionality of the filters, particularly in the extended Kalman filter, where significant enhancement was noticed primarily during the vehicle's non-linear behaviors. The EKF-SLAM topic focuses on applying the Extended Kalman SLAM filter to estimate the vehicle's trajectory and landmarks using odometry and sensor data, with the aim of achieving maximum accuracy. Results are analyzed and presented through various plots, animations, and evaluation metrics.

**THANKS!**