Applied Estimation (EL2320)

Lab 1

Extended Kalman Filter (EKF)

Introduction

This lab consists of two parts:

1. A series of preparatory questions that we will go over in class
2. Two separate MATLAB exercises
   - A warm-up case study with a standard Kalman filter where you learn more about the behavior of the Kalman filter. Very little extra code is needed.
   - The main problem in which you need to complete an implementation of an Extended Kalman filter-based robot localization.

It is a good time to review chapter 1 and 2 to gain more background about what the course is about including Kalman filters, chapter 3 (up to the end of 3.3) to do a recap about the (Extended) Kalman filter and chapter 7 (up to the end of 7.5) to further familiarize yourself with the EKF localization problem. If you want to dig more, you can also read the section 1 and 2 of An Introduction to the Kalman Filter by Greg Welch and Gary Bishop. You need to keep in mind that you will encounter the same equations with different notations in different literature. Therefore, the best way to not confuse different notations is to make a short list of variables/notations in the literature you are reading and stick with one notation that you feel more comfortable with. We will try to use the book, Probabilistic Robotics, notation as much as possible.

In order to pass this lab you need to upload to the course web a pdf with:

1. Written answers to the preparatory questions, Part I
2. Written answers to the questions in Part II

3. The required plots for each dataset. In total, that is four plots, one for each of the first two data sets and two (batch and sequential update) for the last data set. The plots should be analysed in the report with a short text for each plot that explains why it looks the way it does compared to the other plots. More specific instructions are given in Section 2.7.

You must also upload your (**working**) code.

Uploading and passing the lab on time gives a bonus point on the exam. Do as much as you can and upload by the deadline. You may then have to complete sections that don't pass. You need to do the readings and answer the questions (as many as you can) before you start to write the code in order to benefit from this lab.

PART I - Preparatory Questions

These questions are intended to encourage you to think about what we have covered so far on Bayesian filtering, the Kalman filter, the Extended Kalman filter and localization.

You should write out your answers to the questions neatly and legibly and bring them to class on the assigned day. Be sure to include your name. These answers will be collected and redistributed randomly in class. You will be grading each other's papers while we discuss them in class. The act of grading is part of the process of re-enforcing what you have learned. These grades will be recorded but the value of the grade does not effect your grade for the lab. If you get more than 30% correct you will get a bonus point on the exam. You will then be able to correct the answers before uploading the lab report to the assignment on the course web. In other words, getting all the answers correct for class is not necessary, but participation is of great benefit to you. The proper way to do this is come to class with a pdf that you then modify and upload, but as long as you get there in the end, any path is ok.

The 30% applies for the bonus point but is not good enough for passing the lab. Upload correct answers for all questions.

Linear Kalman Filter:

1. What is the difference between a "control" $\mathbf{u}_t$, a "measurement" $\mathbf{z}_t$ and the state $\mathbf{x}_t$? Give examples of each!
2. Can the uncertainty in the belief increase during an update? Why (or not)?
3. During update what is it that decides the weighing between measurements and belief?
4. What would be the result of using too large a covariance (Q matrix) for the measurement model?

5. What would give the measurements an increased effect on the updated state estimate?

6. What happens to the belief uncertainty during prediction? How can you show that?

7. How can we say that the Kalman filter is the optimal and minimum least squared error estimator in the case of independent Gaussian noise and Gaussian priori distribution? (Just describe the reasoning not a formal proof.)

8. In the case of Gaussian white noise and Gaussian priori distribution, is the Kalman Filter a MLE and/or MAP estimator?

Extended Kalman Filter:

9. How does the Extended Kalman filter relate to the Kalman filter?

10. Is the EKF guaranteed to converge to a consistent solution?

11. If our filter seems to diverge often, can we change any parameter to try and reduce this?

Localization:

12. If a robot is completely unsure of its location and measures the range r to a know landmark with Gaussian noise, what does the posterior belief of its location $p(x, y, \theta|r)$ look like? A formula is not needed but describe it at least.

13. If the above measurement also included a bearing, how would the posterior look?

14. If the robot moves with relatively good motion estimation (prediction error is small) but a large initial uncertainty in heading θ, how will the posterior look after traveling a long distance without seeing any features?

15. If the above robot then sees a point feature and measures range and bearing to it, how might the EKF update go wrong?

PART II - MATLAB Exercises

1 Warm-up problem: Linear Kalman filter

Consider a car example. The acceleration of the car is controlled by the throttle/break. The system is described by:

$$\begin{aligned}
x_k &= \begin{bmatrix} p_k \\ v_k \end{bmatrix} \\
u_k &= a_0 \\
A &= \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} \\
B &= \begin{bmatrix} 0 \\ \Delta t \end{bmatrix} \\
x_{k+1} &= Ax_k + Bu_k + \varepsilon_k \\
C &= \begin{bmatrix} 1 & 0 \end{bmatrix} \\
z_k &= Cx_k + \delta_k
\end{aligned} \tag{1}$$

where p_k and v_k are the position and velocity of the car time step k, a_0 is the (constant) acceleration of the car and ε_k and δ_k are white Gaussians.

- **Question 1:** What are the dimensions of ε_k and δ_k? What parameters do you need to define in order to uniquely characterize a white Gaussian?

Open `Warm_Up/kf_car.m`. Read the comments and the code and make sure you understand the system model and the roles of the different variables.

- **Question 2:** Make a table showing the roles/usages of the variables `(x, xhat, P, G, D, Q, R, wStdP, wStdV, vStd, u, PP)`. To do this one must go beyond simply reading the comments in the code. **Hint:** Some of these variables are part of our estimation model and some are for simulating the car motion.

- **Question 3:** Please answer this question with one paragraph of text that summarizes broadly what you learn/deduce from changing the parameters in the code as described below. Choose two illustrative sets of plots to include as demonstration. What do you expect if you increase/decrease the covariance matrix of the modeled (not the actually simulated) process noise/measurement noise by a factor of 100 (only one change per run)? Characterize your expectations. Confirm your expectations using the code (save the corresponding figures so you can analyze them in your report). Do the same analysis for the case of increasing/decreasing both parameters by the same factor at the same time. **Hint:** It is the mean and covariance behavior over time that we are asking about.

- **Question 4:** How do the initial values for P and $\hat{x}$ affect the rate of convergence and the error of the estimates (try both much bigger and much smaller)?

2 Main problem: EKF Localization

Consider the robot localization (tracking) problem. Given a map of the environment (M^1), some measurements and some control signals, you are to estimate where the robot is located at each time step. Using a first order Markov assumption and Bayesian update, this can be formalized in recursive form as:[2]

$$\begin{cases} p(\mathbf{x_t}|\mathbf{u_{1:t}}, \mathbf{z_{1:t}}, \mathbf{\bar{x}_0}, M) & = \eta p(\mathbf{z_t}|\mathbf{x_t}, M) \int p(\mathbf{x_t}|\mathbf{u_t}, \mathbf{x_{t-1}}) p(\mathbf{x_{t-1}}|\mathbf{z_{1:t-1}}, \mathbf{u_{1:t-1}}, \mathbf{\bar{x}_0}, M) \mathbf{dx_{t-1}} \\ p(\mathbf{x_0}|\mathbf{\bar{x}_0}) & = \delta(\mathbf{x_0} - \mathbf{\bar{x}_0}) \end{cases} \tag{2}$$

or equivalently

$$\begin{cases} bel(\mathbf{x_t}) & = p(\mathbf{x_t}|\mathbf{u_{1:t}}, \mathbf{z_{1:t}}, \mathbf{\bar{x}_0}, M) = \eta p(\mathbf{z_t}|\mathbf{x_t}, M) \overline{bel}(\mathbf{x_t}) \\ \overline{bel}(\mathbf{x_t}) & = p(\mathbf{x_t}|\mathbf{u_{1:t}}, \mathbf{z_{1:t-1}}, \mathbf{\bar{x}_0}, M) = \int p(\mathbf{x_t}|\mathbf{u_t}, \mathbf{x_{t-1}}) bel(\mathbf{x_{t-1}}) \mathbf{dx_{t-1}} \\ bel(\mathbf{x_0}) & = p(\mathbf{x_0}|\mathbf{\bar{x}_0}) = \delta(\mathbf{x_0} - \mathbf{\bar{x}_0}) \end{cases} \tag{3}$$

- **Question 5:** Which parts of (2) and (3) are responsible for prediction and update steps?

Now, assuming the observation noise and process noise are white Gaussians, it might seem that a standard Kalman filter can be utilized to come up with the optimal estimate. However, as it will be pointed out in Section 2.2, the measurement model in this case is not linear and thus, the standard Kalman filter will be useless for this problem. However, given the initial conditions (initial position: $\boldsymbol{\mu}_0 = \mathbf{\bar{x}_0}$ and initial uncertainty: $\Sigma_0 = \bar{\Sigma}_0$), we can utilize an EKF to perform the prediction and update steps (Algorithm 1). Note that Algorithm 1 assumes time-invariance for measurement noise and process noise.

Algorithm 1 The Extended Kalman Filter algorithm

```
$\left.\begin{array}{ll} \boldsymbol{\bar{\mu}_t} & = \mathbf{g}(\mathbf{u_t}, \boldsymbol{\mu_{t-1}}) \\ \bar{\Sigma}_t & = G_t \Sigma_{t-1} G_t^T + R \end{array}\right\}$    {Prediction}
$\left.\begin{array}{ll} K_t & = \bar{\Sigma}_t H_t^T (H_t \bar{\Sigma}_t H_t^T + Q)^{-1} \\ \boldsymbol{\mu}_t & = \boldsymbol{\bar{\mu}_t} + K_t(\mathbf{z_t} - \mathbf{h}(\boldsymbol{\bar{\mu}_t})) \\ \Sigma_t & = (I - K_t H_t) \bar{\Sigma}_t \end{array}\right\}$    {Update}
```

[1]The units in the EKF Localization problem are meter and radians.

[2]Note that the definitions for Q and R are reversed in the EKF Localization problem. In the lectures, Q and R correspond to the modeled process and measurement noise covariance matrices respectively, while in the notation in the lab instructions (and also the course literature), Q and R refer to the modeled measurement and process noise covariance matrices.

2.1 Prediction

The prediction can be performed using the odometry information or control signals to the motors for example. Odometry information is usually computed using sensors like wheel encoders. In a simple case, the odometry can be computed using (4):

$$\begin{aligned}
\omega_t^R &= \frac{2*\pi*E_t^R}{E_T \Delta t} \\
\omega_t^L &= \frac{2*\pi*E_t^L}{E_T \Delta t} \\
\omega_t &= \frac{\omega_t^R R_R - \omega_t^L R_L}{B} \\
v_t &= \frac{\omega_t^R R_R + \omega_t^L R_L}{2} \\
u_t &\approx \begin{bmatrix} v_t \Delta t \cos \mu_{t-1,\theta} \\ v_t \Delta t \sin \mu_{t-1,\theta} \\ \omega_t \Delta t \end{bmatrix}
\end{aligned} \tag{4}$$

where E_T is the number of encoder ticks per wheel revolution, E_t^R and E_t^L are the encoder ticks for the right and the left wheel in the t^{th} time step, B is the wheel base (the distance between the contact points of the wheels), R_R and R_L are the radius of the right and the left wheels and ω_t and v_t are the angular and translational velocities of the robot in the t^{th} time step. Having computed the odometry information, the prediction process is performed using (5):

$$\begin{aligned}
\mathbf{g}(\mathbf{u_t}, \boldsymbol{\mu}_{\mathbf{t-1}}) &= \boldsymbol{\mu}_{t-1} + \mathbf{u_t} \\
G_t &= \begin{bmatrix} 1 & 0 & -\sin \mu_{t-1,\theta} v_t \Delta t \\ 0 & 1 & \cos \mu_{t-1,\theta} v_t \Delta_t \\ 0 & 0 & 1 \end{bmatrix}
\end{aligned} \tag{5}$$

where G_t is the Jacobian of $\mathbf{g}$ and $\mu_{t-1,\theta}$ is the orientation of the robot in the last time step.

2.2 Observation Model

Assuming a range-bearing measurement setting, the observation model for the sensor is given by (6):

$$\begin{aligned}
\mathbf{h}(\mathbf{x}_t, M, j) &= \begin{bmatrix} \sqrt{(M_{j,x} - x_{t,x})^2 + (M_{j,y} - x_{t,y})^2} \\ atan2(M_{j,y} - x_{t,y}, M_{j,x} - x_{t,x}) - x_{t,\theta} \end{bmatrix} \\
\mathbf{z}_{t,j} &= \mathbf{h}(\mathbf{x}_t, M, j) + \delta_t \\
\hat{\mathbf{z}}_{t,j} &= \mathbf{h}(\bar{\boldsymbol{\mu}}_t, M, j) \\
H(\bar{\mu}_t, M, j) &= \begin{bmatrix} \frac{\bar{\mu}_{t,x} - M_{j,x}}{\hat{z}_{1,j}} & \frac{\bar{\mu}_{t,y} - M_{j,y}}{\hat{z}_{1,j}} & 0 \\ -\frac{\bar{\mu}_{t,y} - M_{j,y}}{(\hat{z}_{1,j})^2} & \frac{\bar{\mu}_{t,x} - M_{j,x}}{(\hat{z}_{1,j})^2} & -1 \end{bmatrix}
\end{aligned} \tag{6}$$

where j is the index of the landmark being matched to the measurement observation, $\mathbf{h}$ is the observation function, the first component of $\hat{\mathbf{z}}_{t,j}$ is $\hat{z}_{1,j}$, which is also the distance between the feature j and the state $\bar{\boldsymbol{\mu}}_{\mathbf{t}}$; H is the Jacobian of h

corresponding to any observation evaluated at $\bar{\boldsymbol{\mu}}_t$; $\bar{\mu}_{t,x}$ refers to the x component of $\bar{\boldsymbol{\mu}}_t$.

Please keep in mind that indices i and j have no great fixed meaning and in your implementation what seemed natural might not be the same as here or in other solutions. Don't get fixated on the letters but rather on the function.

2.3 Data Association

One of the most challenging problems in estimation such as EKF localization is the data association. If we know which landmark is being observed in each observation, then we have a known associations problem. In most of the situations, we usually do not know this information and thus, need to somehow associate each observation with a landmark (or classify it as an outlier). One of the most popular and simple approaches to the data association problem is the *Maximum Likelihood* data association.

Maximum Likelihood Data Association

In the maximum likelihood data association approach, observations are associated with landmarks separately. For each observation one computes the likelihood of making the observation while observing each landmark from the best estimate of the state before the update process ($\bar{\mu}_t$). Then, each observation is associated with the landmark which leads to the maximum likelihood. Algorithm 2 shows the detailed steps of associating landmark $\hat{c}_t^i$ with the i^{th} observation[3]:

Algorithm 2 Maximum Likelihood Data Association Algorithm for the i^{th} observation

```
for all landmarks $j$ in $M$ do
    $\hat{\mathbf{z}}_{\mathbf{t,j}} = \mathbf{h}(\bar{\boldsymbol{\mu}}_{\mathbf{t}}, M, j)$    {Predict Measurement}
    $H_{t,j} = H(\bar{\boldsymbol{\mu}}_{\mathbf{t}}, M, j, \hat{\mathbf{z}}_{\mathbf{t,j}})$
    $S_{t,j} = H_{t,j}\bar{\Sigma}_t(H_{t,j})^T + Q$
    $\boldsymbol{\nu}_t^{i,j} = \mathbf{z}_{\mathbf{t,i}} - \hat{\mathbf{z}}_{\mathbf{t,j}}$    {Calculate Innovation}
    $\psi_t^{i,j} = \det(2\pi S_{t,j})^{-\frac{1}{2}} exp[-\frac{1}{2}(\boldsymbol{\nu}_t^{i,j})^T(S_{t,j})^{-1}\boldsymbol{\nu}_t^{i,j}]$    {Calculate Likelihood}
end for
$\hat{c}_t^i = \arg\max_j \ \psi_t^{i,j}$
$\bar{\boldsymbol{\nu}}_t^i = \boldsymbol{\nu}_t^{i,\hat{c}_t^i}$
$\bar{S}_{t,i} = S_{t,\hat{c}_t^i}$
$\bar{H}_{t,i} = H_{t,\hat{c}_t^i}$
```

[3]You need to keep in mind that there may be more than one measurement at time t (here we use i to denote them). $z_{t,j}$ refers to the j^{th} feature being associated with any of the measurements at the time t. When the inovation is computed the actual i^{th} measurement needs to be used. Also $\nu_t^{i,\hat{c}_t^i}$ represents the innovation of the i^{th} observation/measurement associated with $\hat{c}_t^i$ at time t.

Notice that ML data association requires that you normalize the Gaussian pdf.

- **Question 6:** In the Maximum Likelihood data association, we assumed that the measurements are independent of each other. Is this a valid assumption? Explain why.

2.4 Outlier Detection

Sometimes, due to various reasons like unforeseen dynamics in the system (e.g. an un-modeled moving object), sensor failure or unreasonable noise, we get measurements which are not reliable. We refer to these measurements as outliers. To detect outliers, one can define a threshold on the likelihood of the measurement (ψ_t^i) and label the measurement as an outlier if the maximum likelihood falls below the threshold. Alternatively, it is possible to define a threshold on the Mahalanobis distance between the measurement (z_t^i) and the most likely association which is given by:

$$D_M = (\bar{\boldsymbol{\nu}}_t^i)^T (\bar{S}_{t,i})^{-1} (\bar{\boldsymbol{\nu}}_t^i) \tag{7}$$

The upside of this method is the fact that D_M follows the chi square (X_n^2) cumulative distribution with n degrees of freedom and therefore, one can define a threshold for this method based on a probability. We encourage you to read more about the intuition behind this in `Outlier_Detection.pdf`. As (in our case with two measurement components) ν has two degrees of freedom, the threshold (λ_M) is given by:

$$\lambda_M = X_2^{-2}(\delta_M) \tag{8}$$

- **Question 7:** What are the bounds for δ_M in (8)? How does the choice of δ_M affect the outlier rejection process? What value do you suggest for λ_M when we have reliable measurements all arising from features in our map, that is all our measurements come from features on our map? What about a scenario with unreliable measurements with many arising from so called clutter or spurious measurements?

2.5 Update

Having computed the associations and rejected the outliers, the update process can be done easily. The most simple type of update, namely sequential update, performs one update for each observation z_t^i in the t^{th} time step (Algorithm 3).

- **Question 8:** Can you think of some down-sides of the sequential update approach (Algorithm 3)? **Hint:** How does the first (noisy) measurement affect the intermediate results?

Algorithm 3 Sequential Update Algorithm for the i^{th} observation

```
for all Observations $i$ in $z_t$ do
    ...            {Compute the $i^{th}$ association using $\bar{\boldsymbol{\mu}}_t$}
    $K_{t,i} = \bar{\Sigma}_t(\bar{H}_{t,i})^T(\bar{S}_{t,i})^{-1}$
    $\bar{\boldsymbol{\mu}}_t = \bar{\boldsymbol{\mu}}_t + K_{t,i}\bar{\boldsymbol{\nu}}_t^i$
    $\bar{\Sigma}_t = (I - K_{t,i}\bar{H}_{t,i})\bar{\Sigma}_t$
end for
$\boldsymbol{\mu}_t = \bar{\boldsymbol{\mu}}_t$
$\Sigma_t = \bar{\Sigma}_t$
```

Batch Update

An alternative to the sequential update is the batch update algorithm. The batch update algorithm associates all observations with landmarks simultaneously and performs one update for all the observations in each time step. The sequential update can cause problems if you get outliers into the system and you perform data association for every new measurement processed. After an update with an outlier the estimate might be thrown off and the uncertainty incorrectly reduced (this is called inconsistency, i.e. that the uncertainty does not capture the true situation) which means that the other measurements may be incorrectly discarded as outliers. The order in which the measurements are processed is therefore important in the sequential update as in the sequential update, the association of the i^{th} measurement is computed using the estimates that are updated using the $i-1^{th}$ measurement. The batch update is less sensitive to outliers as more measurements processed at the same time adds some robustness. The downside of the batch update is instead the computational cost. Algorithm 4 shows the complete EKF localization problem with the batch update.

- **Question 9:** How can you modify Algorithm 4 to avoid redundant recomputations?
- **Question 10:** What are the dimensions of $\bar{\boldsymbol{\nu}}_t$ and $\bar{H}_t$ in Algorithm 4? What were the corresponding dimensions in the sequential update algorithm? What does this tell you?

2.6 Instructions

In the unzipped lab file, go to the folder `EKF_Localization`. This directory contains all MATLAB files required to complete the main problem. You are provided with code skeletons for all functions you are asked to implement. Make sure to add this directory along with all subdirectories to your current MATLAB path. For the simulation we make use of a GUI which allows you to change parameters during the simulation and observe the effect on the filter's behavior.

In the remainder of this section, we give a more explicit description of the functions you are asked to implement along with some common implementation

Algorithm 4 EKF Localization with Batch update for the i^{th} time step

```
for all Observations $i$ in $z_t$ do
  for all Landmarks $j$ in $M$ do
    $\hat{\mathbf{z}}_{t,j} = \mathbf{h}(\bar{\boldsymbol{\mu}}_t, M, j)$
    $H_{t,j} = H(\bar{\boldsymbol{\mu}}_t, M, j, \hat{\mathbf{z}}_{t,j})$
    $S_{t,j} = H_{t,j}\bar{\Sigma}_t(H_{t,j})^T + Q$
    $\boldsymbol{\nu}_t^{i,j} = \mathbf{z_{t,i}} - \hat{\mathbf{z}}_{\mathbf{t,j}}$
    $D_t^{i,j} = (\boldsymbol{\nu}_t^{i,j})^T(S_{t,j})^{-1}\boldsymbol{\nu}_t^{i,j}$
    $\psi_t^{i,j} = \det(2\pi S_{t,j})^{-\frac{1}{2}} exp[-\frac{1}{2}D_t^{i,j}]$
  end for
  $\hat{c}_t^i = \arg\max_j \ \psi_t^{i,j}$
  $\hat{o}_t^i = D^{i,c_t^i} \geq \lambda_M$
  $\bar{\boldsymbol{\nu}}_t^i = \boldsymbol{\nu}_t^{i,\hat{c}_t^i}$
  $\bar{H}_{t,i} = H_{t,\hat{c}_t^i}$
end for
{For inlier indices $1, ..., n$}
$\bar{\boldsymbol{\nu}}_t = \begin{bmatrix} (\bar{\boldsymbol{\nu}}_t^1)^T & (\bar{\boldsymbol{\nu}}_t^2)^T & ... & (\bar{\boldsymbol{\nu}}_t^n)^T \end{bmatrix}^T$
$\bar{H}_t = \begin{bmatrix} (\bar{H}_{t,1})^T & (\bar{H}_{t,2})^T & ... & (\bar{H}_{t,n})^T \end{bmatrix}^T$
$\bar{Q}_t = \begin{bmatrix} Q & 0 & 0 & 0 \\ 0 & Q & 0 & 0 \\ 0 & 0 & ... & 0 \\ 0 & 0 & 0 & Q \end{bmatrix}_{(2n\times 2n)}$
$K_t = \bar{\Sigma}_t(\bar{H}_t)^T(\bar{H}_t\bar{\Sigma}_t(\bar{H}_t)^T + \bar{Q}_t)^{-1}$
$\boldsymbol{\mu}_t = \bar{\boldsymbol{\mu}}_t + K_t\bar{\boldsymbol{\nu}}_t$
$\Sigma_t = (I - K_t\bar{H}_t)\bar{\Sigma}_t$
```

errors. Furthermore, we provide you with a quick tutorial on how to use the GUI once you've finished your implementation.

Implementation Details

In order to run the simulation on the given datasets, you first have to complete the implementation of the following functions:

1. **calculate_odometry.m**: Use (4)
2. **predict.m**: Use Algorithm 1 and (5)
3. **observation_model.m**: Use (6)
4. **jacobian_observation_model.m**: Use (**??**)
5. **associate.m**: Use Algorithm 2
6. **update_.m**: Use Algorithm 3
7. **batch_associate.m**: Use Algorithm 4
8. **batch_update.m**: Use Algorithm 4

We provide you with empty skeletons for each function in `EKF_Localization/Skeletons`. Please don't change any of the written code, especially the global variables shouldn't be modified as they affect the performance of the GUI. Use the specified shapes of input and output data as a guideline for your implementation.

The total number of lines of code you need to write to complete all mentioned functions is approximately 80 lines. You can also use MATLAB's help command to see the comments: e.g. `help predict`.

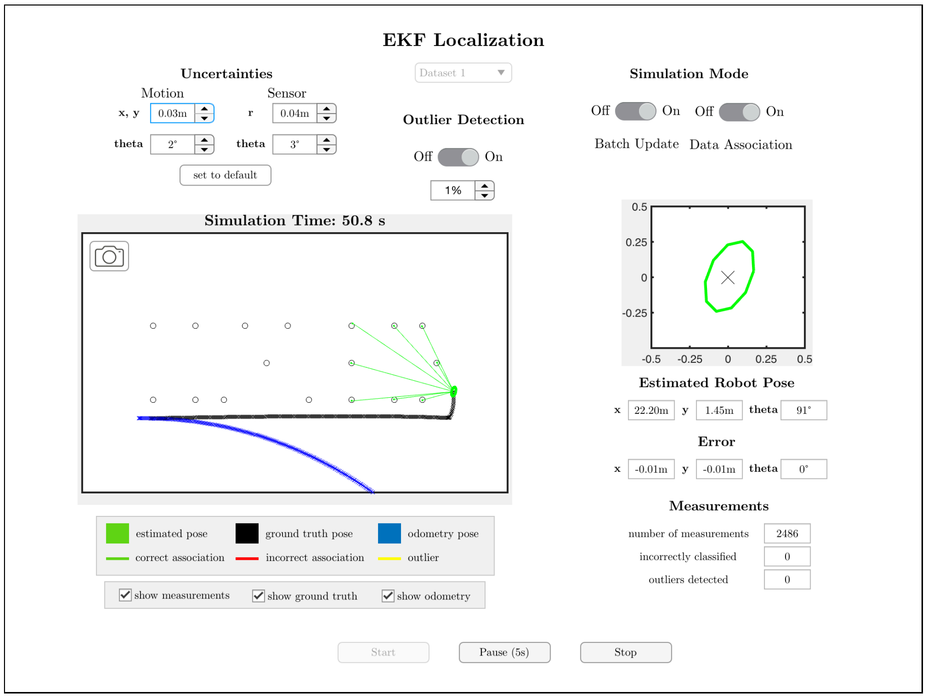
Implementation Errors

The most common error while implementing the functions is to not realize that an angle error (innovation) of 2π is an error of 0. One should explicitly make sure that all angles (even intermediate results) are between $-\pi$ and π. This is done by adding or subtracting 2π. We suggest using this line of code to put an angle ϕ in the range:

$$\phi = \quad \mathrm{mod}\ (\phi + \pi, 2\pi) - \pi; \tag{9}$$

Another common error is the matrix indexing. For example `H(:,k)` instead of `H(:,:,k)` or similar. Make sure to check the given input and output shapes. Also, remember that indexing in MATLAB starts at 1. Furthermore, whenever possible, try to vectorize your matrix operations instead of using loops to improve performance of your code.

To check if your implementation is correct, we provide you with a test function for each file you are supposed to modify. These functions are located in


EKF Localization

Dataset 1

Uncertainties

Motion: x, y 0.03m; theta 2°

Sensor: r 0.04m; theta 3°

Outlier Detection: Off On; 1%

Simulation Mode: Batch Update Off On; Data Association Off On

Simulation Time: 50.8 s

Estimated Robot Pose: x 22.20m, y 1.45m, theta 91°

Error: x -0.01m, y -0.01m, theta 0°

Measurements

number of measurements	2486
incorrectly classified	0
outliers detected	0

🟩 estimated pose ■ ground truth pose 🟦 odometry pose

correct association, incorrect association, outlier

✓ show measurements ✓ show ground truth ✓ show odometry

Figure 1: Illustration of the GUI. Parameters and filter modes can be changed during the simulation and the effect on the filter performance can be observed immediately.

`EKF_Localization/Test_Cases`. Make sure to run the corresponding test case for each function you implement. If the test function returns 1, you most likely have a bug-free implementation of the respective function. Finally, to make sure all functions work together as well, there is a final test function called `test_cases_lab1.m`. This functions runs all previous test cases and returns 1 if all functions appear correct. Please don't proceed to the next step before your implementation passes the final test case.

Simulation GUI

For the final simulation of the EKF localization algorithm we use the GUI displayed in Figure 1. The GUI was developed and tested on MATLAB 2019a. Start the GUI by running `ekf_gui` from your command window.

In the following, we explain in detail how to interact with the GUI during the experiments:

- **Start Simulation** To start a simulation, select a dataset from the drop-down menu and press the start button. This will start the simulation using the current parameter setting.

- **Modify Parameters** All active buttons, switches and spinners can be used to change the respective parameters during the simulation. The changes will take immediate effect on the filter performance.
  - **Uncertainties** Using the spinners, the modeled noise on motion model and observation model can be adjusted. The values shown are the respective standard deviations.
  - **Outlier Detection** Using this switch, the outlier detection can be enabled/disabled. The threshold corresponds to 1-δ_M. Consequently, a higher threshold results in more detected outliers.
  - **Batch Update** Using this switch, the update mode can be changed from batch update to sequential update.
  - **Data Association** By disabling data association, the maximum likelihood estimation is skipped and the ground truth associations are used.
  - **Set Default** When selecting a dataset, reasonable values are provided for all parameters. The "Default-Button" undoes all changes made to any parameters during the active simulation.
- **Capture Simulation** For the documentation of your conducted simulations, you are required to include plots of various scenarios in your report. By clicking on the camera icon in the main simulation window, a screen shot is created and opened in a separate figure which can be saved and included in your report.
- **Analyse Filter Performance** In order to numerically analyze the filter performance, it is convenient to plot the evolution of the estimation error over time. If you wait for the simulation to finish, additional figures containing plotted errors along with the evolution of the covariance matrices will be displayed. However, this will only work if you don't terminate the simulation manually.

2.7 Datasets

In this last section, we introduce the three datasets used in the lab assignment and describe the final instructions for the experiments you should perform for your report. For each dataset you are asked to save one or more plots. You should include the plots in your pdf file and describe the situation displayed in each plot.

1. **Dataset 1**: In this dataset, the laser scanner has an accuracy of (1 cm, 1 degree), the odometry information has an un-modeled noise of approximately 1 cm and 1 degree per time step. Decide about reasonable noise models for process and observation noises and set your parameters in the

GUI. If you have done everything correctly, the resulting mean absolute error of your estimations should be less than 0.01 (m, rad) on all dimensions.

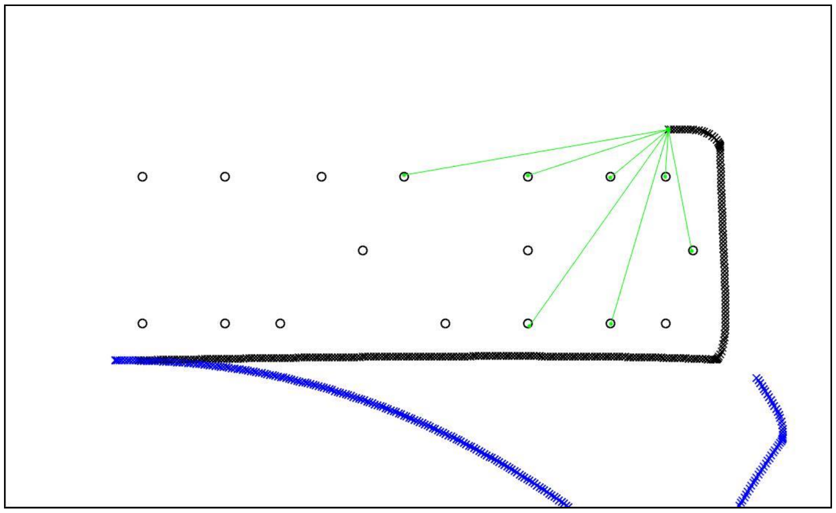


Figure 2: Dataset 1. Accurate sensor readings are available and low uncertainties on both measurement and motion model can be used to achieve accurate tracking.

2. **Dataset 2**: In this dataset, there are certain outliers that need to be detected (and rejected). The measurements have undergone a white Gaussian with the standard deviation of 0.2 (m, rad). Try to find a reasonable value for the process noise covariance matrix. If your outlier detection works correctly, the resulting mean absolute error of your estimations should be less than 0.06 (m, rad) on all dimensions.

3. **Dataset 3**: This data set does not include any odometry information and therefore, we should compensate for it by considering a big standard deviation for the process noise. Model the measurement noise with a standard deviation of 0.1 (m, rad), the process noise with a standard deviation of 1 (m, rad) and disable the outlier detection. Run your algorithm on the data set using 1) the sequential update algorithm, 2) the batch update algorithm. If you have done everything right, you should be able to see a sensible difference between the results of the two algorithms with mean absolute error of the batch update algorithm being less than 0.1 (m, rad) on all dimensions.

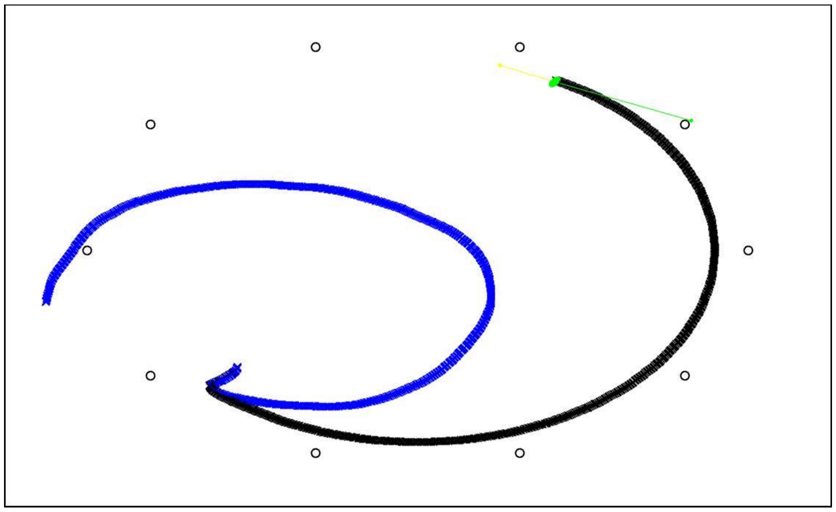


Figure 3: Dataset 2. The available measurements contain a large number of outliers that should be detected and discarded. Using bad measurements (yellow) in the update can result in an inconsistent estimate. The uncertainty of the measurement model and the threshold for outlier detection are most important for this dataset.

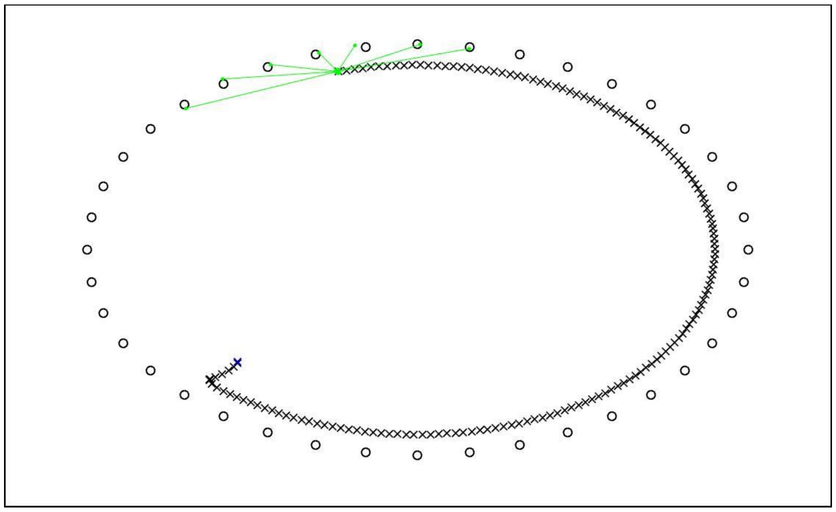


Figure 4: Dataset 3. No odometry information is available for this dataset. The map layout is prone to association errors due to the large number of landmarks. Robust updates are important.