Extended Kalman Filter and Particle Filter Lab Report

Ghaith Chamaa & Rafay Aamir

May 13 2025

1 Theoretical part

Kalman Gain

We need to show that in 1D, the Kalman Filter correction step:

$$\mu_{\text{updated}} = \bar{\mu} + K(z - \bar{\mu})$$
$$\sigma^2_{\text{updated}} = (1 - K)\bar{\sigma}^2$$

where $K = \frac{\bar{\sigma}^2}{\bar{\sigma}^2 + \sigma^2_{\text{obs}}}$, is equivalent (up to a scale factor) to multiplying the Gaussians of the predicted state $N(x; \bar{\mu}, \bar{\sigma}^2)$ and the observation $N(x; z, \sigma^2_{\text{obs}})$.

Let:

- $f(x) = N(x; \mu_1, \sigma_1^2)$ be the predicted state, so $\mu_1 = \bar{\mu}$ and $\sigma_1^2 = \bar{\sigma}^2$.
- $g(x) = N(x; \mu_2, \sigma_2^2)$ be the observation, so $\mu_2 = z$ and $\sigma_2^2 = \sigma^2_{\text{obs}}$.

The product $f(x)g(x)$ is proportional to a new Gaussian $N(x; \mu_{\text{prod}}, \sigma^2_{\text{prod}})$. From the provided formula:

$$\mu_{\text{prod}} = \frac{\sigma_2^2\mu_1 + \sigma_1^2\mu_2}{\sigma_1^2 + \sigma_2^2}$$
$$\sigma^2_{\text{prod}} = \frac{1}{\frac{1}{\sigma_1^2} + \frac{1}{\sigma_2^2}}$$

Substituting our specific terms:

1. **Calculate μ_{prod}:**

$$\mu_{\text{prod}} = \frac{\sigma_{\text{obs}}^2 \bar{\mu} + \bar{\sigma}^2 z}{\bar{\sigma}^2 + \sigma_{\text{obs}}^2}$$

2. **Calculate σ_{prod}^2:**

$$\begin{aligned}
\sigma_{\text{prod}}^2 &= \frac{1}{\frac{1}{\bar{\sigma}^2} + \frac{1}{\sigma_{\text{obs}}^2}} \\
&= \frac{1}{\frac{\sigma_{\text{obs}}^2 + \bar{\sigma}^2}{\bar{\sigma}^2 \sigma_{\text{obs}}^2}} \\
&= \frac{\bar{\sigma}^2 \sigma_{\text{obs}}^2}{\bar{\sigma}^2 + \sigma_{\text{obs}}^2}
\end{aligned}$$

Now, let's expand the Kalman Filter update equations:

1. **Expand μ_{updated}:**

$$\begin{aligned}
\mu_{\text{updated}} &= \bar{\mu} + K(z - \bar{\mu}) \\
&= \bar{\mu} + \frac{\bar{\sigma}^2}{\bar{\sigma}^2 + \sigma_{\text{obs}}^2}(z - \bar{\mu}) \\
&= \frac{\bar{\mu}(\bar{\sigma}^2 + \sigma_{\text{obs}}^2) + \bar{\sigma}^2(z - \bar{\mu})}{\bar{\sigma}^2 + \sigma_{\text{obs}}^2} \\
&= \frac{\bar{\mu}\bar{\sigma}^2 + \bar{\mu}\sigma_{\text{obs}}^2 + \bar{\sigma}^2 z - \bar{\sigma}^2\bar{\mu}}{\bar{\sigma}^2 + \sigma_{\text{obs}}^2} \\
&= \frac{\bar{\mu}\sigma_{\text{obs}}^2 + \bar{\sigma}^2 z}{\bar{\sigma}^2 + \sigma_{\text{obs}}^2}
\end{aligned}$$

Comparing μ_{updated} with μ_{prod}, we see they are identical: $\mu_{\text{updated}} = \mu_{\text{prod}}$.

2. **Expand $\sigma^2_{\text{updated}}$:**

$$\begin{aligned}
\sigma^2_{\text{updated}} &= (1-K)\bar{\sigma}^2 \\
&= \left(1 - \frac{\bar{\sigma}^2}{\bar{\sigma}^2 + \sigma^2_{\text{obs}}}\right)\bar{\sigma}^2 \\
&= \left(\frac{\bar{\sigma}^2 + \sigma^2_{\text{obs}} - \bar{\sigma}^2}{\bar{\sigma}^2 + \sigma^2_{\text{obs}}}\right)\bar{\sigma}^2 \\
&= \left(\frac{\sigma^2_{\text{obs}}}{\bar{\sigma}^2 + \sigma^2_{\text{obs}}}\right)\bar{\sigma}^2 \\
&= \frac{\bar{\sigma}^2\sigma^2_{\text{obs}}}{\bar{\sigma}^2 + \sigma^2_{\text{obs}}}
\end{aligned}$$

Comparing $\sigma^2_{\text{updated}}$ with σ^2_{prod}, we see they are identical: $\sigma^2_{\text{updated}} = \sigma^2_{\text{prod}}$.

Since the mean and variance derived from multiplying the two Gaussians are identical to the mean and variance obtained from the Kalman Filter correction step, the equivalence is shown.

Jacobian Motion Model

The robot state is $\mathbf{s} = [x, y, \theta]^T$. The control input is $\mathbf{u} = [\delta_{\text{rot1}}, \delta_{\text{trans}}, \delta_{\text{rot2}}]^T$. The motion model equations $\mathbf{g}(\mathbf{s}_t, \mathbf{u}_t)$ predict the next state $\mathbf{s}_{t+1} = [x', y', \theta']^T$:

$$\begin{aligned}
x' &= x_t + \delta_{\text{trans}}\cos(\theta_t + \delta_{\text{rot1}}) \\
y' &= y_t + \delta_{\text{trans}}\sin(\theta_t + \delta_{\text{rot1}}) \\
\theta' &= \theta_t + \delta_{\text{rot1}} + \delta_{\text{rot2}}
\end{aligned}$$

We need to derive the Jacobians:

- $G = \frac{\partial \mathbf{g}}{\partial \mathbf{s}}$
- $V = \frac{\partial \mathbf{g}}{\partial \mathbf{u}}$

For clarity, we drop the subscript t for x, y, θ when taking derivatives with respect to state components.

1. Jacobian $G = \frac{\partial \mathbf{g}}{\partial \mathbf{s}}$

$\mathbf{s} = [x, y, \theta]^T$ and $\mathbf{g} = [x', y', \theta']^T$. G will be a 3×3 matrix:

$$G = \begin{pmatrix} \frac{\partial x'}{\partial x} & \frac{\partial x'}{\partial y} & \frac{\partial x'}{\partial \theta} \\ \frac{\partial y'}{\partial x} & \frac{\partial y'}{\partial y} & \frac{\partial y'}{\partial \theta} \\ \frac{\partial \theta'}{\partial x} & \frac{\partial \theta'}{\partial y} & \frac{\partial \theta'}{\partial \theta} \end{pmatrix}$$

Partial derivatives:

- $\frac{\partial x'}{\partial x} = \frac{\partial}{\partial x}(x + \delta_{\text{trans}} \cos(\theta + \delta_{\text{rot1}})) = 1$
- $\frac{\partial x'}{\partial y} = \frac{\partial}{\partial y}(x + \delta_{\text{trans}} \cos(\theta + \delta_{\text{rot1}})) = 0$
- $\frac{\partial x'}{\partial \theta} = \frac{\partial}{\partial \theta}(x + \delta_{\text{trans}} \cos(\theta + \delta_{\text{rot1}})) = -\delta_{\text{trans}} \sin(\theta + \delta_{\text{rot1}})$
- $\frac{\partial y'}{\partial x} = \frac{\partial}{\partial x}(y + \delta_{\text{trans}} \sin(\theta + \delta_{\text{rot1}})) = 0$
- $\frac{\partial y'}{\partial y} = \frac{\partial}{\partial y}(y + \delta_{\text{trans}} \sin(\theta + \delta_{\text{rot1}})) = 1$
- $\frac{\partial y'}{\partial \theta} = \frac{\partial}{\partial \theta}(y + \delta_{\text{trans}} \sin(\theta + \delta_{\text{rot1}})) = \delta_{\text{trans}} \cos(\theta + \delta_{\text{rot1}})$
- $\frac{\partial \theta'}{\partial x} = \frac{\partial}{\partial x}(\theta + \delta_{\text{rot1}} + \delta_{\text{rot2}}) = 0$
- $\frac{\partial \theta'}{\partial y} = \frac{\partial}{\partial y}(\theta + \delta_{\text{rot1}} + \delta_{\text{rot2}}) = 0$
- $\frac{\partial \theta'}{\partial \theta} = \frac{\partial}{\partial \theta}(\theta + \delta_{\text{rot1}} + \delta_{\text{rot2}}) = 1$

So, the Jacobian G (or G_t if referring to time t) is:

$$G_t = \begin{pmatrix} 1 & 0 & -\delta_{\text{trans}} \sin(\theta_t + \delta_{\text{rot1}}) \\ 0 & 1 & \delta_{\text{trans}} \cos(\theta_t + \delta_{\text{rot1}}) \\ 0 & 0 & 1 \end{pmatrix}$$

2. Jacobian $V = \frac{\partial \mathbf{g}}{\partial \mathbf{u}}$

$\mathbf{u} = [\delta_{\text{rot1}}, \delta_{\text{trans}}, \delta_{\text{rot2}}]^T$ and $\mathbf{g} = [x', y', \theta']^T$. V will be a 3×3 matrix:

$$V = \begin{pmatrix} \frac{\partial x'}{\partial \delta_{\text{rot1}}} & \frac{\partial x'}{\partial \delta_{\text{trans}}} & \frac{\partial x'}{\partial \delta_{\text{rot2}}} \\ \frac{\partial y'}{\partial \delta_{\text{rot1}}} & \frac{\partial y'}{\partial \delta_{\text{trans}}} & \frac{\partial y'}{\partial \delta_{\text{rot2}}} \\ \frac{\partial \theta'}{\partial \delta_{\text{rot1}}} & \frac{\partial \theta'}{\partial \delta_{\text{trans}}} & \frac{\partial \theta'}{\partial \delta_{\text{rot2}}} \end{pmatrix}$$

Partial derivatives:

- $\frac{\partial x'}{\partial \delta_{\text{rot1}}} = \frac{\partial}{\partial \delta_{\text{rot1}}}(x_t + \delta_{\text{trans}} \cos(\theta_t + \delta_{\text{rot1}})) = -\delta_{\text{trans}} \sin(\theta_t + \delta_{\text{rot1}})$
- $\frac{\partial x'}{\partial \delta_{\text{trans}}} = \frac{\partial}{\partial \delta_{\text{trans}}}(x_t + \delta_{\text{trans}} \cos(\theta_t + \delta_{\text{rot1}})) = \cos(\theta_t + \delta_{\text{rot1}})$
- $\frac{\partial x'}{\partial \delta_{\text{rot2}}} = \frac{\partial}{\partial \delta_{\text{rot2}}}(x_t + \delta_{\text{trans}} \cos(\theta_t + \delta_{\text{rot1}})) = 0$
- $\frac{\partial y'}{\partial \delta_{\text{rot1}}} = \frac{\partial}{\partial \delta_{\text{rot1}}}(y_t + \delta_{\text{trans}} \sin(\theta_t + \delta_{\text{rot1}})) = \delta_{\text{trans}} \cos(\theta_t + \delta_{\text{rot1}})$
- $\frac{\partial y'}{\partial \delta_{\text{trans}}} = \frac{\partial}{\partial \delta_{\text{trans}}}(y_t + \delta_{\text{trans}} \sin(\theta_t + \delta_{\text{rot1}})) = \sin(\theta_t + \delta_{\text{rot1}})$
- $\frac{\partial y'}{\partial \delta_{\text{rot2}}} = \frac{\partial}{\partial \delta_{\text{rot2}}}(y_t + \delta_{\text{trans}} \sin(\theta_t + \delta_{\text{rot1}})) = 0$
- $\frac{\partial \theta'}{\partial \delta_{\text{rot1}}} = \frac{\partial}{\partial \delta_{\text{rot1}}}(\theta_t + \delta_{\text{rot1}} + \delta_{\text{rot2}}) = 1$
- $\frac{\partial \theta'}{\partial \delta_{\text{trans}}} = \frac{\partial}{\partial \delta_{\text{trans}}}(\theta_t + \delta_{\text{rot1}} + \delta_{\text{rot2}}) = 0$
- $\frac{\partial \theta'}{\partial \delta_{\text{rot2}}} = \frac{\partial}{\partial \delta_{\text{rot2}}}(\theta_t + \delta_{\text{rot1}} + \delta_{\text{rot2}}) = 1$

So, the Jacobian V (or V_t if referring to time t) is:

$$V_t = \begin{pmatrix} -\delta_{\text{trans}} \sin(\theta_t + \delta_{\text{rot1}}) & \cos(\theta_t + \delta_{\text{rot1}}) & 0 \\ \delta_{\text{trans}} \cos(\theta_t + \delta_{\text{rot1}}) & \sin(\theta_t + \delta_{\text{rot1}}) & 0 \\ 1 & 0 & 1 \end{pmatrix}$$

2 Experimental Setup

The goal of this report is to implement the Extended Kalman Filter and the Particle Filter (deals with non-parametric), and conduct experiments with respect to the robustness of both filters to noise, by doing simulations and changing the noise factor that models the covariance matrices of noise for movement and observation **filter_factor** for the filters and for the noise of the environment setup called **data_factor**.

3 Analysis

In the following sections, we analyze the robustness of EKF and PF.

3.1 On EKF

Each of the follwoing follows the same structure as the lab handout.

3.2 Task a

This part shows the running of the implementation with default values for **filter_factor** and **data_factor**.

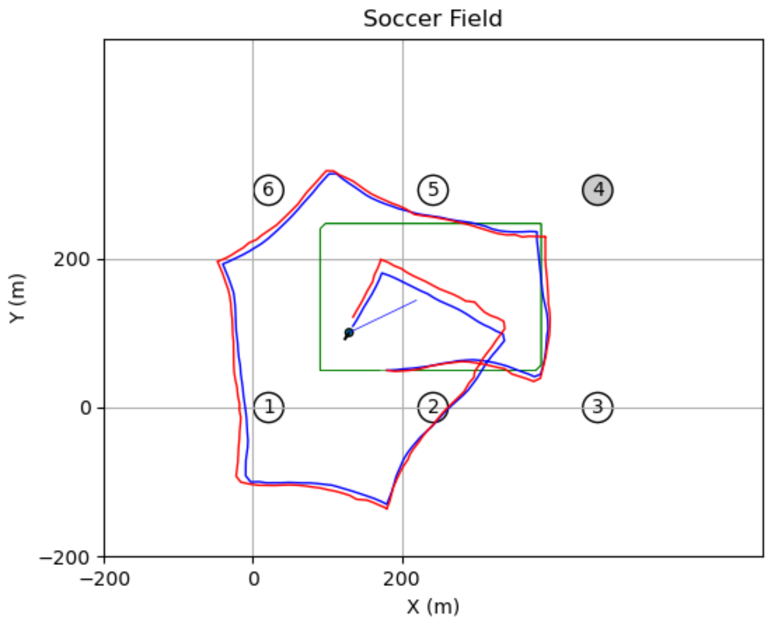


Figure 1: EKF running with default parameters.

The output from the default run is shown below.

```
Data factor: 1
Filter factor: 1
----------------------------------
Mean position error: 8.998367536084693
Mean Mahalanobis error: 4.416418248584294
ANEES: 1.4721394161947645
```

Listing 1: Output from default parameter run

The filter behaves as expected this is noticed by the the figure where there is no big difference between the real noisy trajectory and the trajectory of the filter.

3.3 Task b

Here we plot the mean position error as the factors α & β vary over r = [1/64, 1/16, 1/4, 4, 16, 64] and analyze the results obtained, here **r** is affecting both **filter_factor** & **data_factor** the factors α & β that affect the covariance matrices for the noise in both the filter and the environment, we run 10 trials per value of r, and take the median value for robustness to noise.

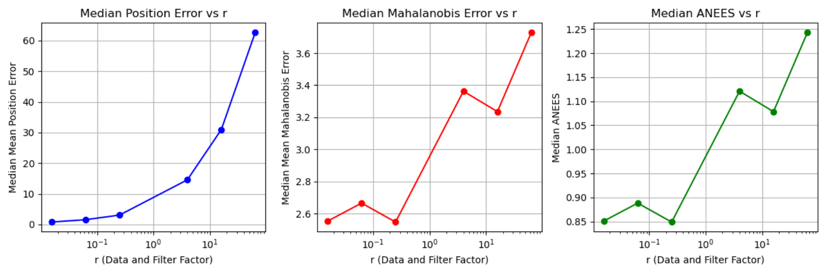


Figure 2: Robustness of EKF to noise while varying both **filter_factor** and **data_factor** for the env and the filter.

The output from the default run is shown below.

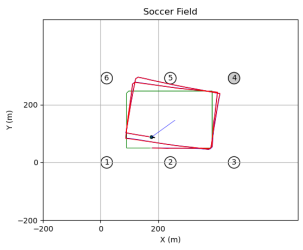
```
Data factor: 0.015625
Filter factor: 0.015625
---------------------------
Mean position error: 0.6722214978172342
Mean Mahalanobis error: 2.1539657777424006
ANEES: 0.7179885925808002

Data factor: 0.0625
Filter factor: 0.0625
---------------------------
Mean position error: 1.2870913752315627
Mean Mahalanobis error: 1.8985519415182606
ANEES: 0.6328506471727535
```

```
Data factor: 0.25
Filter factor: 0.25
---------------------------
Mean position error: 3.106370398932189
Mean Mahalanobis error: 2.8450806185195434
ANEES: 0.9483602061731812

Data factor: 4.0
Filter factor: 4.0
---------------------------
Mean position error: 16.188478055670956
Mean Mahalanobis error: 4.1419046607511145
ANEES: 1.3806348869170382

Data factor: 16.0
Filter factor: 16.0
---------------------------
Mean position error: 41.42404370721852
Mean Mahalanobis error: 2.9725455295927623
ANEES: 0.9908485098642541

Data factor: 16.0
Filter factor: 16.0
---------------------------
Mean position error: 37.18178058383316
Mean Mahalanobis error: 3.939828382426171
ANEES: 1.3132761274753904

Data factor: 64.0
Filter factor: 64.0
---------------------------
Mean position error: 52.01930331368815
Mean Mahalanobis error: 2.835273714254182
ANEES: 0.9450912380847273
```

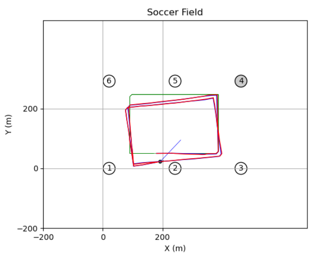
Listing 2: Output from running the experiment

3.4 Discussion on previous part

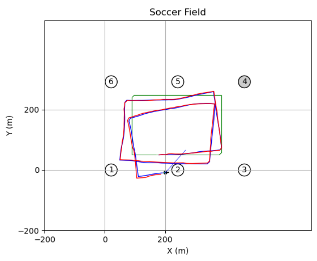
When we have low process noise affecting the covariance matrices, the filter captures the noise of the environment very adequately. While we are increasing the **r** scale factor that affects *filter_factor* and *data_factor*, we notice the



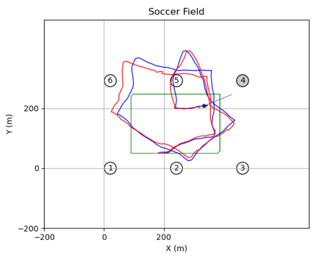
(a) r = 1/64



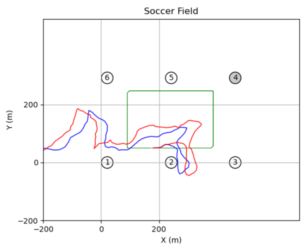
(b) r = 1/16



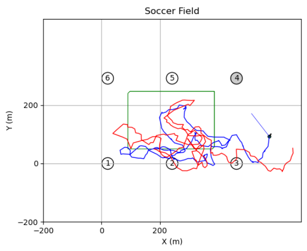
(c) r = 1/4



(d) r = 4



(e) r = 16



(f) r = 64

Figure 3: The trajectory of the filter over all values of r

filter has trouble approximating the trajectory. This is because there are huge shifts and very high variances introduced by the noise that affect the covariance matrices and the filter pipeline.

3.5 Task c

Here we plot the mean position error as the factors α & β vary over r = [1/64, 1/16, 1/4, 4, 16, 64] and analyze the results obtained, here **r** is affecting only **filter_factor** and using the default **data_factor** the factors α & β that affect the covariance matrices for the noise in both the filter and the environment, we run 10 trials per value of r, and take the median value for robustness to noise.

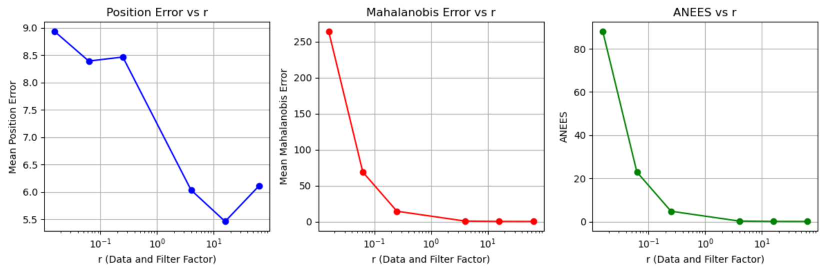


Figure 4: Robustness of EKF to noise while varying the **filter_factor** for the filter.

The output from the default run is shown below.

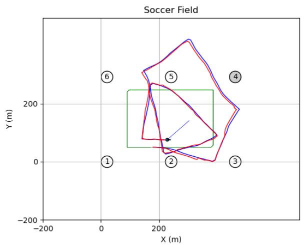
```
Data factor: 1
Filter factor: 0.015625
--------------------------
Mean position error: 5.763912359935691
Mean Mahalanobis error: 129.42791112582486
ANEES: 43.14263704194162

Data factor: 1
Filter factor: 0.0625
```

```
--------------------------
Mean position error: 5.941376280254251
Mean Mahalanobis error: 34.55302291978641
ANEES: 11.517674306595469

Data factor: 1
Filter factor: 0.25
--------------------------
Mean position error: 6.122718450753841
Mean Mahalanobis error: 10.448253052397668
ANEES: 3.482751017465889

Data factor: 1
Filter factor: 4.0
--------------------------
Mean position error: 7.504185067341938
Mean Mahalanobis error: 0.7130769571596332
ANEES: 0.23769231905321106

Data factor: 1
Filter factor: 16.0
--------------------------
Mean position error: 8.32925918612968
Mean Mahalanobis error: 0.26169081220890733
ANEES: 0.08723027073630245

Data factor: 1
Filter factor: 64.0
--------------------------
Mean position error: 6.809370285882931
Mean Mahalanobis error: 0.04742968033153172
ANEES: 0.015809893443843908
```

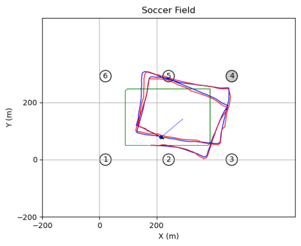
Listing 3: Output from running the experiment

3.6 Discussion on previous part

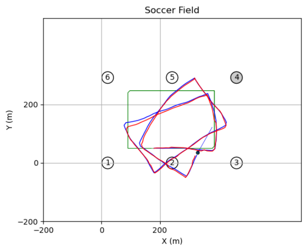
When we have low process noise affecting the covariance matrices of the filter only while the enviroment is introducing higher noise, the filter captures the noise of the environment very poorly, this is because the filter underestimates the process noise in its covariance matrices. While we are increasing the **r** scale factor that affects *filter_factor* only, we notice the filter improves in



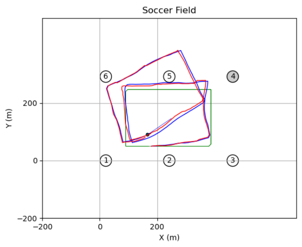
(a) r = 1/64



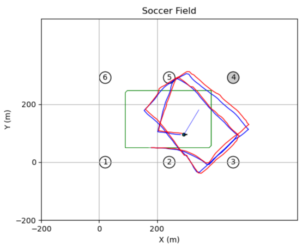
(b) r = 1/16



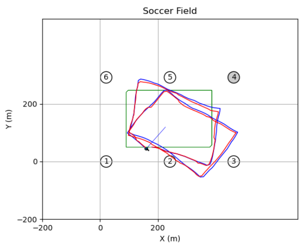
(c) r = 1/4



(d) r = 4



(e) r = 16



(f) r = 64

Figure 5: The trajectory of the filter over all values of r

approximating the trajectory. This is because the filter is estimating and for higher r it overestimates the noise that affect the covariance matrices in the filter pipeline, making more robust in its trajectory.

3.7 On PF

Each of the follwoing follows the same structure as the lab handout.

3.8 Task a

This part shows the running of the implementation with default values for **filter_factor** and **data_factor**.

The output from the default run is shown below.

```
Data factor: 1
Filter factor: 1
------------------------------
Mean position error: 8.567264372950898
Mean Mahalanobis error: 14.742252771106537
ANEES: 4.914084257035513
```

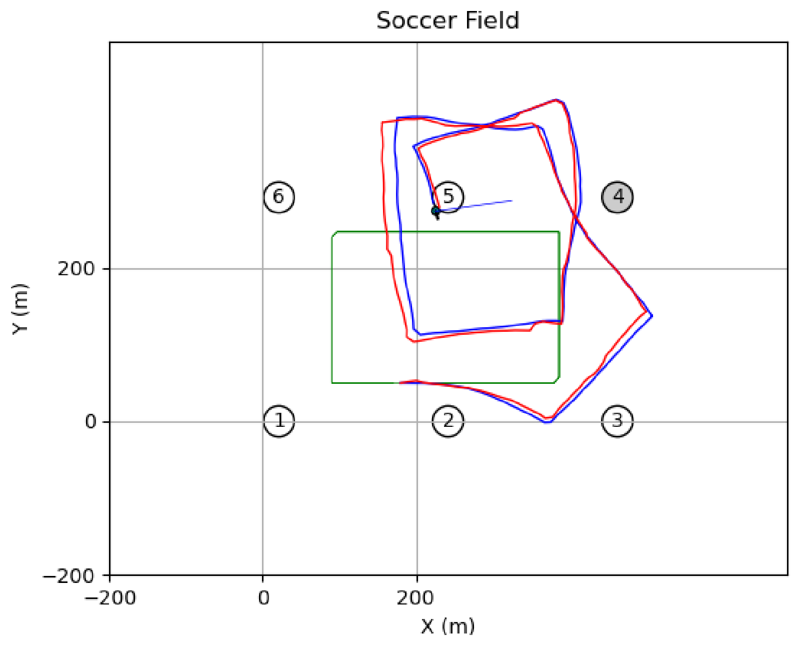


Figure 6: PF running with default parameters.

```

```

Listing 4: Output from default parameter run

The filter behaves as expected this is noticed by the the figure where there is no big difference between the real noisy trajectory and the trajectory of the filter.

3.9 Task b

Here we plot the mean position error as the factors α & β vary over r = [1/64, 1/16, 1/4, 4, 16, 64] and analyze the results obtained, here **r** is affecting both **filter_factor** & **data_factor** the factors α & β that affect the noise of the process from which we are drawing samples from and also the ammount of noise in movement and observation in filter steps in both the filter and the environment, we run 10 trials per value of r, and take the median value for robustness to noise.

The output from the default run is shown below.

```
            Data factor: 0.015625
```

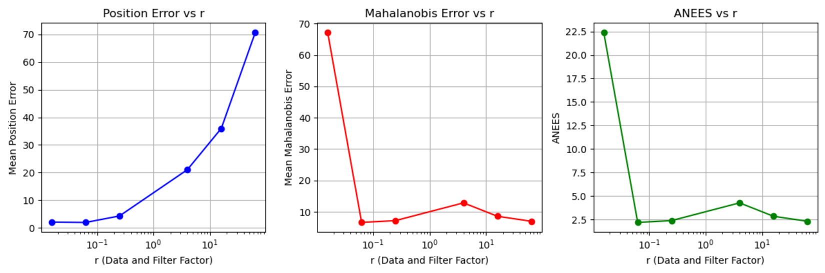
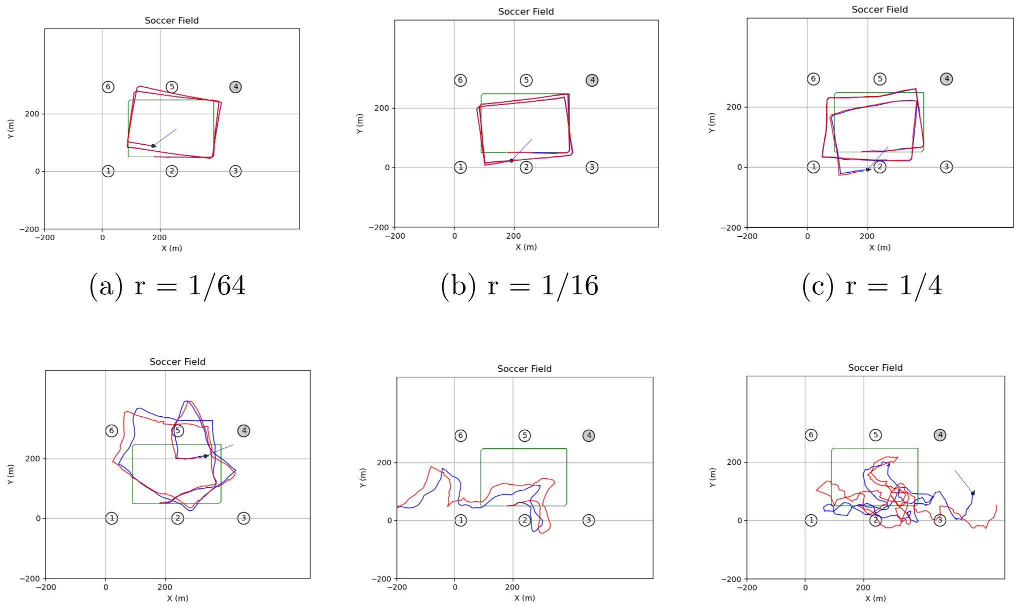


Figure 7: Robustness of PF to noise while varying both **filter_factor** and **data_factor** for the env and the filter.

```
Filter factor: 0.015625
---------------------------------
Mean position error: 3.2448189123320232
Mean Mahalanobis error: 245.73415910891737
ANEES: 81.91138636963912

Data factor: 0.0625
Filter factor: 0.0625
---------------------------------
Mean position error: 3.894702179813308
Mean Mahalanobis error: 79.62590756757595
ANEES: 26.541969189191985

Data factor: 0.25
Filter factor: 0.25
---------------------------------
Mean position error: 4.095557902912394
Mean Mahalanobis error: 7.015695577044925
ANEES: 2.3385651923483084

Data factor: 4.0
Filter factor: 4.0
---------------------------------
Mean position error: 17.83923895240144
Mean Mahalanobis error: 22.35668607733333
ANEES: 7.452228692444443

Data factor: 16.0
Filter factor: 16.0
---------------------------------
Mean position error: 27.097754221310527
Mean Mahalanobis error: 4.174475160873427
```

```
ANEES: 1.3914917202911423

Data factor: 64.0
Filter factor: 64.0
--------------------------------
Mean position error: 95.40173776502938
Mean Mahalanobis error: 8.966351578180108
ANEES: 2.988783859393369
```

Listing 5: Output from runing the experiment



(a) r = 1/64 (b) r = 1/16 (c) r = 1/4

(d) r = 4 (e) r = 16 (f) r = 64

Figure 8: The trajectory of the filter over all values of r

3.10 Discussion on previous part

Similar to EKF, when we have low process noise affecting the covariance matrices, the filter captures the noise of the environment very adequately. While we are increasing the **r** scale factor that affects *filter_factor* and *data_factor*, we notice the filter has trouble approximating the trajectory. This is because there are huge shifts and very high variances introduced by the noise that affect the update of the particles and consequently this affects the filter pipeline in approximating the localization trajectory.

3.11 Task c

Here we plot the mean position error as the factors α & β vary over r = [1/64, 1/16, 1/4, 4, 16, 64] and analyze the results obtained, here **r** is affecting only **filter_factor** and keeping the default **data_factor** these factors α & β that affect the noise of the process from which we are drawing samples from and also the amount of noise in movement and observation in filter steps in both the filter, we run 10 trials per value of r, and take the median value for robustness to noise.

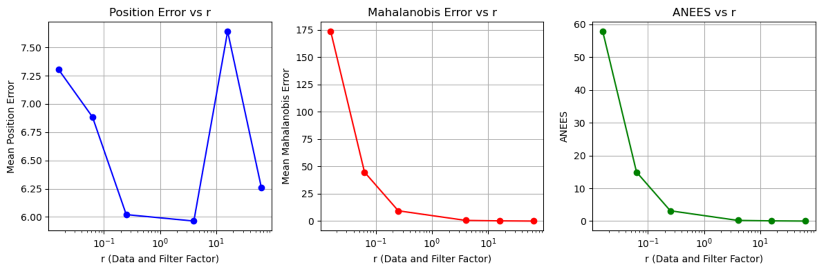


Figure 9: Robustness of PF to noise while varying **filter_factor** for the filter.

The output from the default run is shown below.

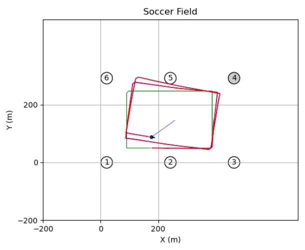
```
Data factor: 1
Filter factor: 0.015625
----------------------------
Mean position error: 73.38403331501544
Mean Mahalanobis error: 91644826.87401302
ANEES: 30548275.62467101

Data factor: 1
Filter factor: 0.0625
```

```
----------------------------
Mean position error: 27.403134443489677
Mean Mahalanobis error: 3696582434.083284
ANEES: 1232194144.694428

Data factor: 1
Filter factor: 0.25
---------------------------
Mean position error: 15.8678385404053
Mean Mahalanobis error: 23355178800.89689
ANEES: 7785059600.298963

Data factor: 1
Filter factor: 4.0
---------------------------
Mean position error: 7.445470421089603
Mean Mahalanobis error: 1.8712226751343946
ANEES: 0.6237408917114649

Data factor: 1
Filter factor: 16.0
---------------------------
Mean position error: 9.757650782461878
Mean Mahalanobis error: 1.1366935396868925
ANEES: 0.37889784656229747

Data factor: 1
Filter factor: 64.0
---------------------------
Mean position error: 22.44437395979284
Mean Mahalanobis error: 1.2184515831689777
ANEES: 0.4061505277229926
```

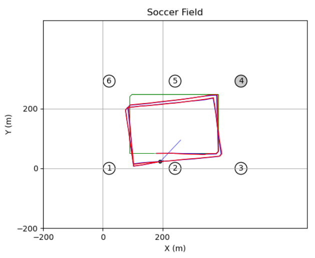
Listing 6: Output from runing the experiment

3.12 Discussion on previous part

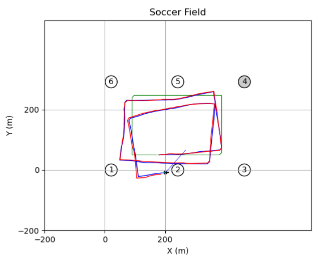
Similar to EKF, when we have low process noise affecting the covariance matrices of the filter only while the enviroment is introducing higher noise, the filter captures the noise of the environment very poorly, this is because the filter underestimates the process noise in its covariance matrices. While we are increasing the **r** scale factor that affects *filter_factor* only, we notice



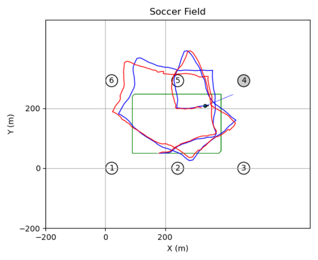
(a) r = 1/64



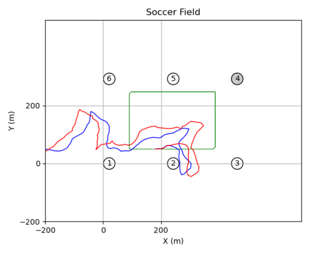
(b) r = 1/16



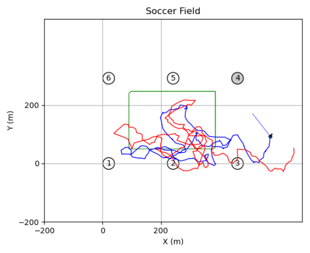
(c) r = 1/4



(d) r = 4



(e) r = 16



(f) r = 64

Figure 10: The trajectory of the filter over all values of r

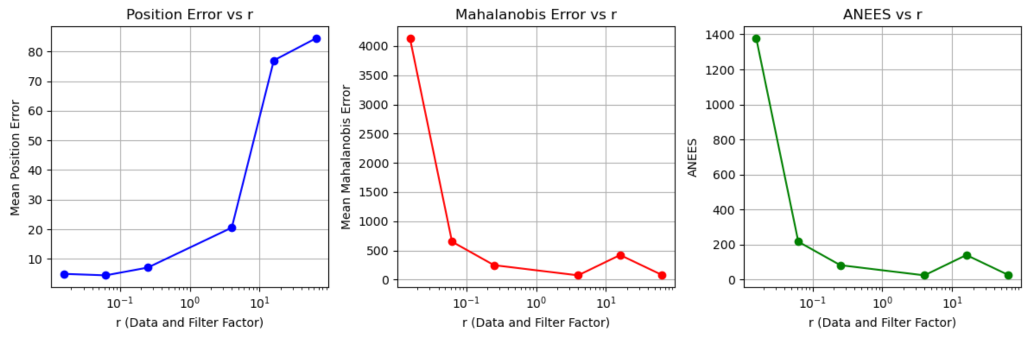
the filter improves in approximating the trajectory. This is because the filter is estimating and for higher r it overestimates the noise that affect the robustness of particles in the filter pipeline, making more robust in its trajectory.

3.13 Task d

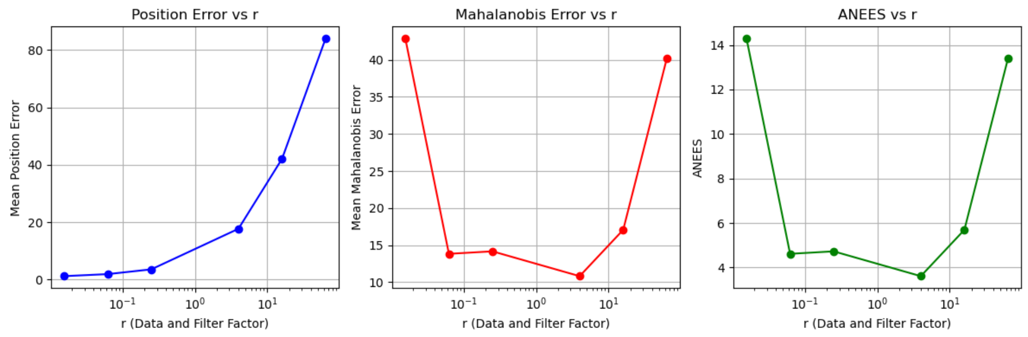
Here we plot the mean position error as the factors α & β vary over r = [1/64, 1/16, 1/4, 4, 16, 64] and analyze the results obtained, here **r** is affecting both **filter_factor** & **data_factor** the factors α & β that affect the noise of the process from which we are drawing samples from and also the ammount of noise in movement and observation in filter steps in both the filter and the environment, we run 10 trials per value of r, and take the median value for robustness to noise, this is done while the number of particles varies over [20, 50, 500].

The output from the default run is shown below.

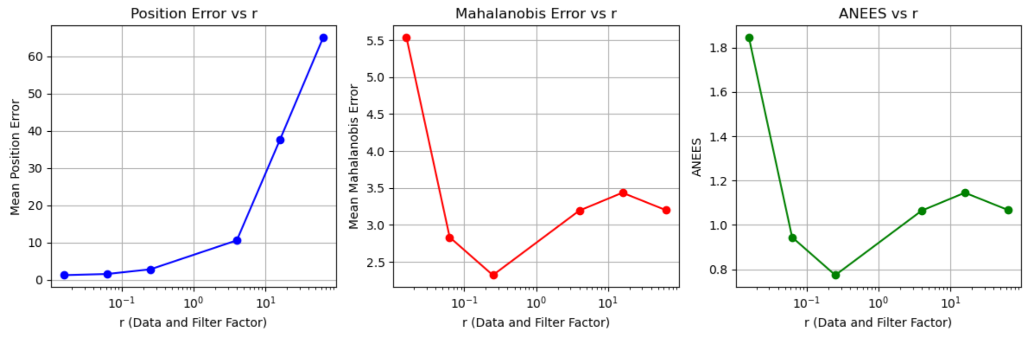
```
              for 20 particles
```



(a) Robustness of PF to noise while varying **filter_factor** for the filter for 20 particles



(b) Robustness of PF to noise while varying **filter_factor** for the filter for 50 particles

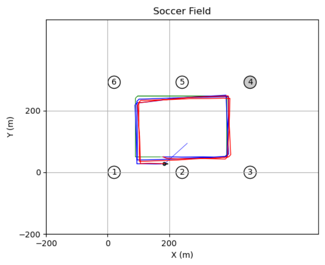


(c) Robustness of PF to noise while varying **filter_factor** for the filter for 500 particles

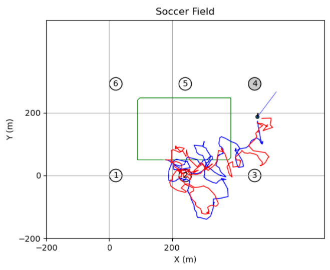
Figure 11: The trajectory of the filter over all values of r

```
Data factor: 0.015625
Filter factor: 0.015625
--------------------------
Mean position error: 4.176061466647679
Mean Mahalanobis error: 4965.961070999495
ANEES: 1655.3203569998316

Data factor: 64.0
Filter factor: 64.0
--------------------------
Mean position error: 45.787545778328415
Mean Mahalanobis error: 14.866025683835591
ANEES: 4.9553418946118635

for 50 particles

Data factor: 0.015625
Filter factor: 0.015625
--------------------------
Mean position error: 2.652811851749612
Mean Mahalanobis error: 127.05290542804673
ANEES: 42.350968476015574

Data factor: 64.0
Filter factor: 64.0
--------------------------
Mean position error: 160.06593213186068
Mean Mahalanobis error: 99.00519851809406
ANEES: 33.001732839364685

for 500 particles

Data factor: 0.015625
Filter factor: 0.015625
--------------------------
Mean position error: 1.4572814218932124
Mean Mahalanobis error: 17.204468971165895

Data factor: 64.0
Filter factor: 64.0
--------------------------
Mean position error: 74.68954744222013
Mean Mahalanobis error: 4.653777099060265
ANEES: 1.5512590330200882
```

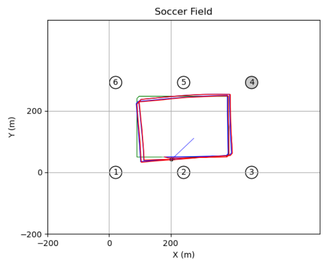
Listing 7: Output from runing the experiment



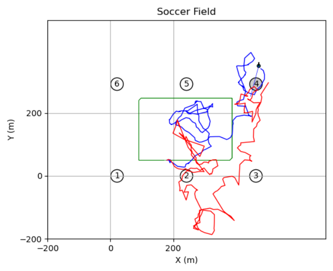
(a) r = 1/16, 20 particles



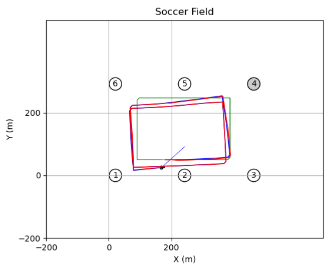
(b) r = 64, 20 particles



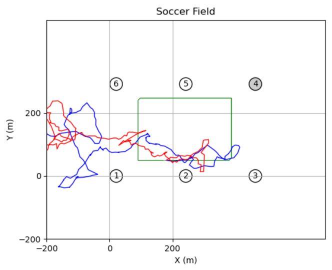
(c) r = 1/16, 50 particles



(d) r = 64, 50 particles



(e) r = 1/16, 500 particles



(f) r = 64, 500 particles

Figure 12: The trajectory of the filter over all values of r

3.14 Discussion on previous part

Similar to Task b, when we have low process noise affecting the covariance matrices, the filter captures the noise of the environment very adequately. While we are increasing the **r** scale factor that affects *filter_factor* and *data_factor*, we notice the filter has trouble approximating the trajectory. This is because there are huge shifts and very high variances introduced by the noise that affect the update of the particles and consequently this affects the filter pipeline in approximating the localization trajectory, this is by viewing the mean position error.

But note that for the Mahlanobis metric:

$$D_M(\mathbf{x}) = \sqrt{(\mathbf{x} - \boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu})}$$

we have an interesting behavior, where we have low r value we get high Mahlanobis error for 50 particles this is because of the inverse of the covariance, for low r we have low covariance then the inverse is very big, so this augments the Mahlanobis metric, In addition to that, for high r we get high Mahlanobis error for 50 particles this is because of the inverse of the covariance and high state vector error, for high r we have high covariance then the inverse is very small, but the high state vector error is high which augments the Mahlanobis metric.