CSE 571

Homework 1

Grant Tannert

Saturday 6th May, 2023

1.1 Motion Model Jacobian

$$\frac{\partial x_{t+1}}{\partial x_t} = 1 \quad \frac{\partial x_{t+1}}{\partial y_t} = 0 \quad \frac{\partial x_{t+1}}{\partial \theta_t} = -\delta_{trans} * \sin(\theta_t + \delta_{rot1})$$
$$\frac{\partial y_{t+1}}{\partial x_t} = 0 \quad \frac{\partial y_{t+1}}{\partial y_t} = 1 \quad \frac{\partial y_{t+1}}{\partial \theta_t} = \delta_{trans} * \cos(\theta_t + \delta_{rot1})$$
$$\frac{\partial \theta_{t+1}}{\partial x_t} = 0 \quad \frac{\partial \theta_{t+1}}{\partial y_t} = 0 \quad \frac{\partial \theta_{t+1}}{\partial \theta_t} = 1$$

$$G = \begin{bmatrix} 1 & 0 & -\delta_{trans} * \sin(\theta_t + \delta_{rot1}) \\ 0 & 1 & \delta_{trans} * \cos(\theta_t + \delta_{rot1}) \\ 0 & 0 & 1 \end{bmatrix}$$

$$\frac{\partial x_{t+1}}{\partial \delta_{rot1}} = -\delta_{trans} * \sin(\theta_t + \delta_{rot1}) \quad \frac{\partial x_{t+1}}{\partial \delta_{trans}} = \cos(\theta_t + \delta_{rot1}) \quad \frac{\partial x_{t+1}}{\partial \delta_{rot2}} = 0$$
$$\frac{\partial y_{t+1}}{\partial \delta_{rot1}} = \delta_{trans} * \cos(\theta_t + \delta_{rot1}) \quad \frac{\partial y_{t+1}}{\partial \delta_{trans}} = \sin(\theta_t + \delta_{rot1}) \quad \frac{\partial y_{t+1}}{\partial \delta_{rot2}} = 0$$
$$\frac{\partial \theta_{t+1}}{\partial \delta_{rot1}} = 1 \quad \frac{\partial \theta_{t+1}}{\partial \delta_{trans}} = 0 \quad \frac{\partial \theta_{t+1}}{\partial \delta_{rot2}} = 1$$

$$V = \begin{bmatrix} -\delta_{trans} * \sin(\theta_t + \delta_{rot1}) & \cos(\theta_t + \delta_{rot1}) & 0 \\ \delta_{trans} * \cos(\theta_t + \delta_{rot1}) & \sin(\theta_t + \delta_{rot1}) & 0 \\ 1 & 0 & 1 \end{bmatrix}$$

1.2 Observation Model Jacobian

$$\frac{\partial}{\partial x_t}\frac{l_y - y_t}{l_x - x_t} = \frac{l_y - y_t}{(l_x - x_t)^2} \text{ by the Quotient rule.}$$

$$\frac{\partial \phi}{\partial x_t} = \frac{1}{1 + (\frac{l_y - y_t}{l_x - x_t})^2} * \frac{l_y - y_t}{(l_x - x_t)^2} = \frac{l_y - y_t}{(l_x - x_t)^2 + (l_y - y_t)^2}$$

$$\frac{\partial}{\partial y_t}\frac{l_y - y_t}{l_x - x_t} = -\frac{1}{l_x - x_t} \text{ by the Quotient rule.}$$

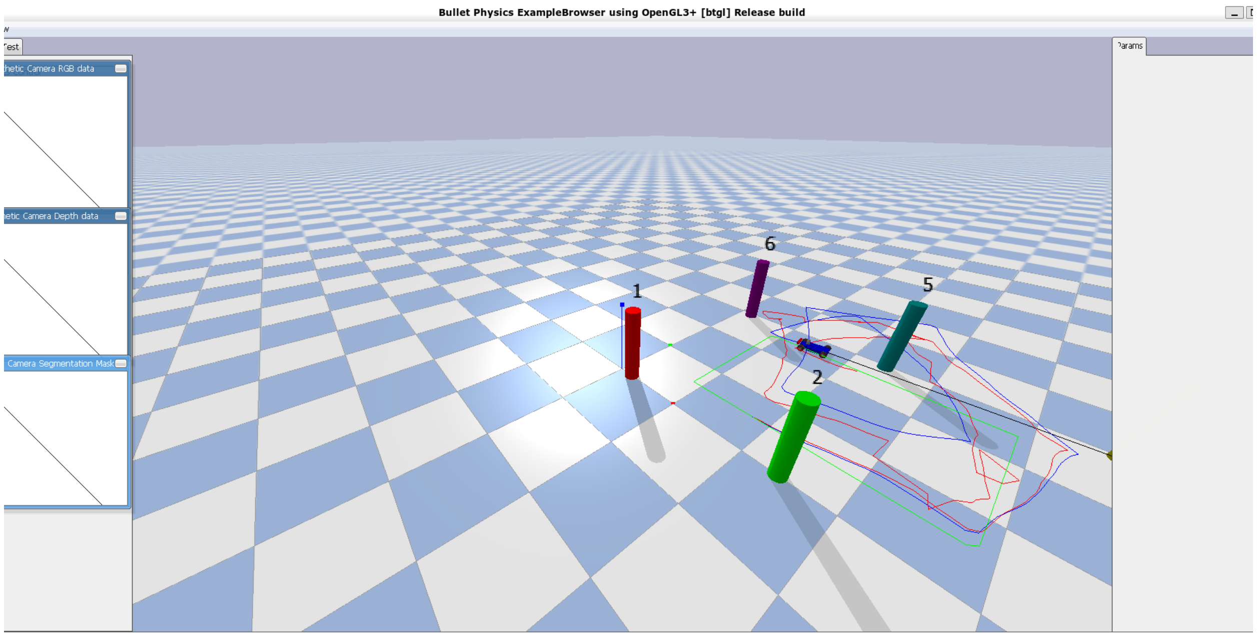
$$\frac{\partial \phi}{\partial y_t} = \frac{1}{1 + (\frac{l_y - y_t}{l_x - x_t})^2} * -\frac{1}{l_x - x_t} = -\frac{l_x - x_t}{(l_x - x_t)^2 + (l_y - y_t)^2}$$

$$\frac{\partial \phi}{\partial \theta_t} = -1$$

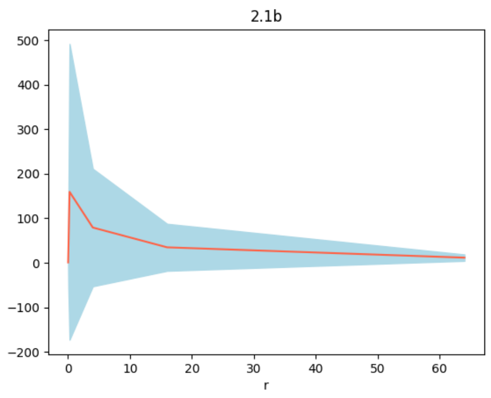
$H = \begin{bmatrix} \frac{l_y - y_t}{(l_x - x_t)^2 + (l_y - y_t)^2} & -\frac{l_x - x_t}{(l_x - x_t)^2 + (l_y - y_t)^2} & -1 \end{bmatrix}$

2.1 EKF Implementation

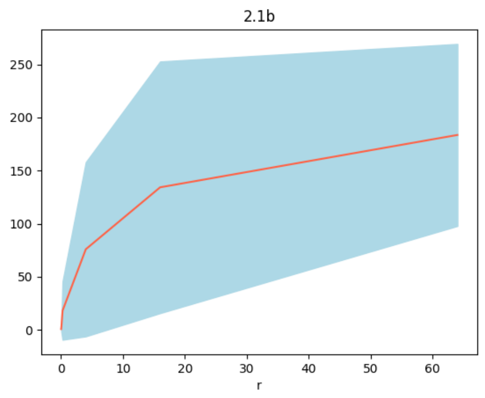
(a)



(b)



(a) ANEES



(b) position error

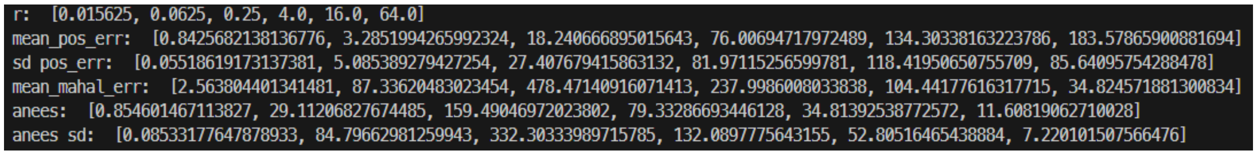
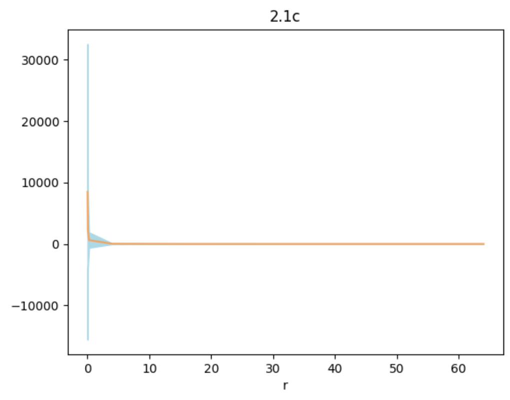

```
r:  [0.015625, 0.0625, 0.25, 4.0, 16.0, 64.0]
mean_pos_err:  [0.8425682138136776, 3.2851994265992324, 18.240666895015643, 76.00694717972489, 134.30338163223786, 183.57865900881694]
sd pos_err:  [0.05518619173137381, 5.085389279427254, 27.407679415863132, 81.97115256599781, 118.41950650755709, 85.64095754288478]
mean_mahal_err:  [2.563804401341481, 87.33620483023454, 478.47140916071413, 237.9986008033838, 104.44177616317715, 34.824571881300834]
anees:  [0.854601467113827, 29.11206827674485, 159.49046972023802, 79.33286693446128, 34.81392538772572, 11.60819062710028]
anees sd:  [0.08533177647878933, 84.79662981259943, 332.30333989715785, 132.0897775643155, 52.80516465438884, 7.220101507566476]
```

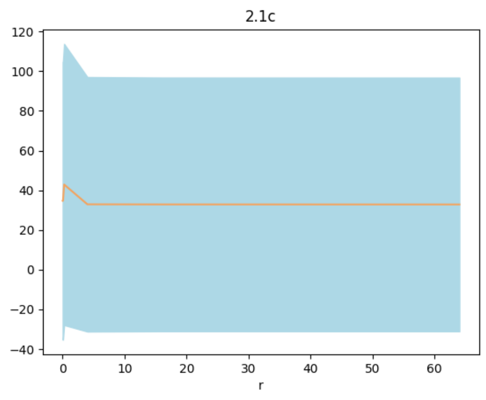
Figure 2: Data

Observations
Note that because of the outliers, the data looks very skewed and we can see this from the standard deviation being really high. Looking at the data file I produced, it seems that the mean position error and ANEES both increase r increases for our filter and data/env. It is noted that for values of $r \leq 1$, the mean position error and ANEES were really low mostly roughly around 0.9.

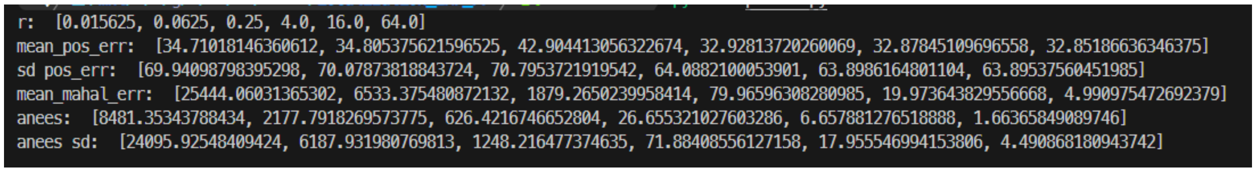
(c)



(a) ANEES



(b) position error



```
r:  [0.015625, 0.0625, 0.25, 4.0, 16.0, 64.0]
mean_pos_err:  [34.71018146360612, 34.805375621596525, 42.904413056322674, 32.92813720260069, 32.87845109696558, 32.85186636346375]
sd pos_err:  [69.94098798395298, 70.07873818843724, 70.7953721919542, 64.0882100053901, 63.8986164801104, 63.89537560451985]
mean_mahal_err:  [25444.06031365302, 6533.375480872132, 1879.2650239958414, 79.96596308280985, 19.973643829556668, 4.990975472692379]
anees:  [8481.35343788434, 2177.7918269573775, 626.4216746652804, 26.655321027603286, 6.657881276518888, 1.66365849089746]
anees sd:  [24095.92548409424, 6187.931980769813, 1248.216477374635, 71.88408556127158, 17.955546994153806, 4.490868180943742]
```

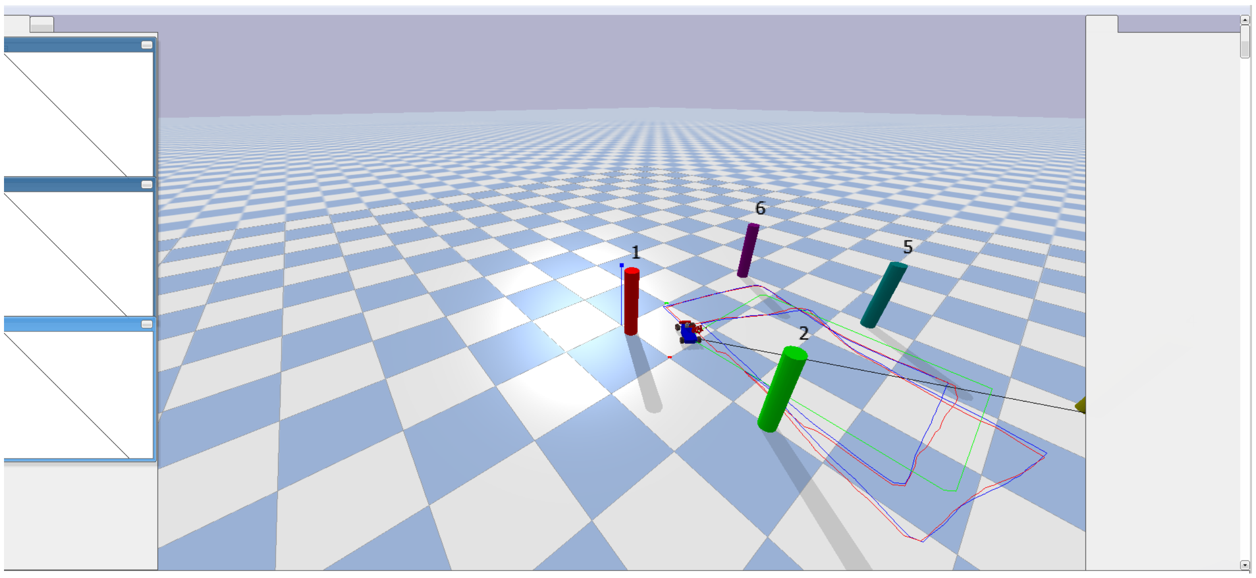
Figure 4: Data

Observations
r refers to the scale factor for the filter, while the data filter stays the same. The mean position error doesn't really change relative to r and looks like it stays relatively high around 30. While the ANEES decreases as we increase r.

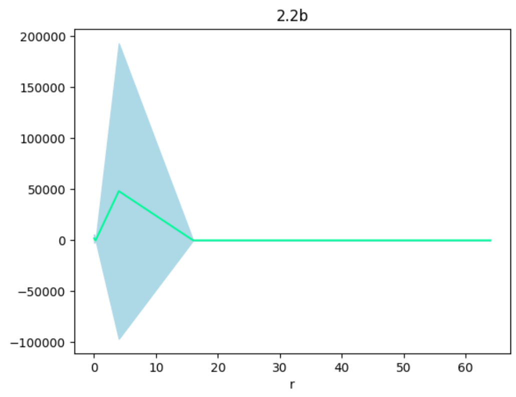
(d) I looked into the lecture slides for the PF implementation and I found it difficult to understand which methods correspond to some steps and what some variables represented. I printed a lot of their shapes and to figure out what each variable represented and after some time I was able to implement it. Another trouble I had was when matrix multiplying, I ran into a lot of mismatch shape bugs so I had to print them out and write their dimensions out and did a lot of reshaping to get it to work.

2.2

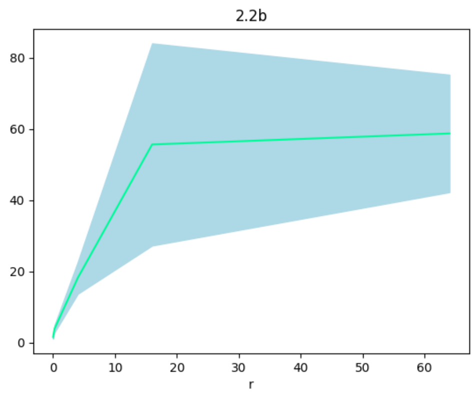
(a)



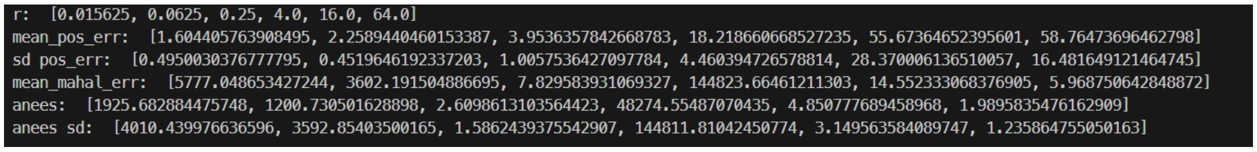
(b)



(a) ANEES



(b) position error

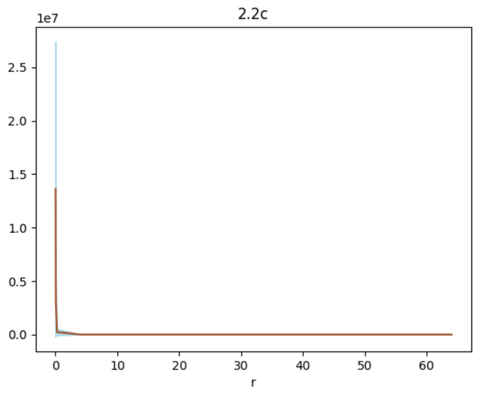


```
r:  [0.015625, 0.0625, 0.25, 4.0, 16.0, 64.0]
mean_pos_err:  [1.604405763908495, 2.2589440460153387, 3.9536357842668783, 18.218660668527235, 55.67364652395601, 58.76473696462798]
sd pos_err:  [0.4950030376777795, 0.4519646192337203, 1.0057536427097784, 4.460394726578814, 28.370006136510057, 16.481649121464745]
mean_mahal_err:  [5777.048653427244, 3602.191504886695, 7.829583931069327, 144823.66461211303, 14.552333068376905, 5.968750642848872]
anees:  [1925.682884475748, 1200.730501628898, 2.6098613103564423, 48274.55487070435, 4.850777689458968, 1.9895835476162909]
anees sd:  [4010.439976636596, 3592.85403500165, 1.5862439375542907, 144811.81042450774, 3.149563584089747, 1.235864755050163]
```

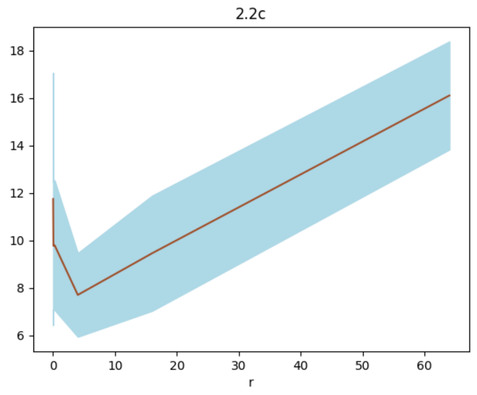
Figure 6: Data

Observations
r is the scale factor for the filter and data/env noise. As r increases, the mean position error increases. While the ANEES, slightly decreases, but is almost the same.

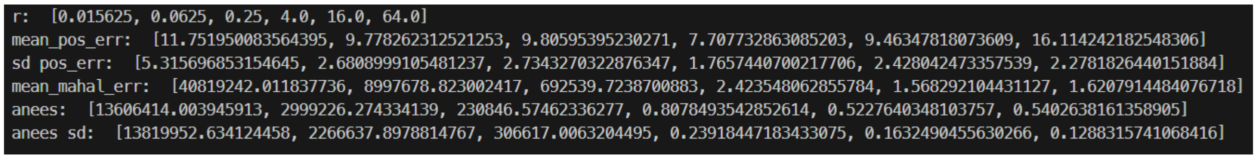
(c)



(a) ANEES



(b) position error

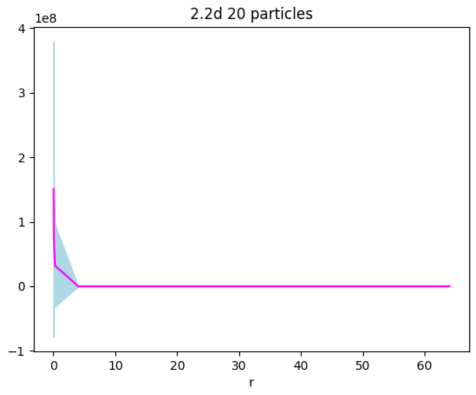


```
r:  [0.015625, 0.0625, 0.25, 4.0, 16.0, 64.0]
mean_pos_err:  [11.751950083564395, 9.778262312521253, 9.80595395230271, 7.707732863085203, 9.46347818073609, 16.114242182548306]
sd pos_err:  [5.315696853154645, 2.6808999105481237, 2.7343270322876347, 1.7657440700217706, 2.428042473357539, 2.2781826440151884]
mean_mahal_err:  [40819242.011837736, 8997678.823002417, 692539.7238700883, 2.423548062855784, 1.568292104431127, 1.6207914484076718]
anees:  [13606414.003945913, 2999226.274334139, 230846.57462336277, 0.8078493542852614, 0.5227640348103757, 0.5402638161358905]
anees sd:  [13819952.634124458, 2266637.8978814767, 306617.0063204495, 0.23918447183433075, 0.1632490455630266, 0.1288315741068416]
```

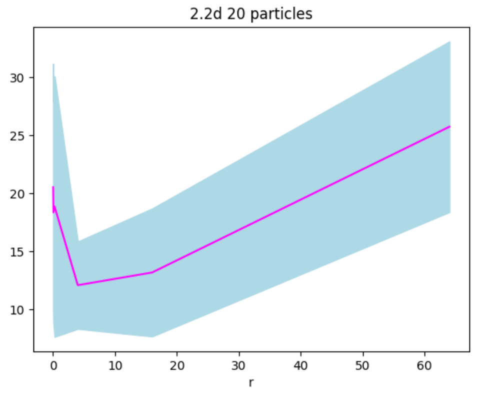
Figure 8: Data

Observations
r is the scale factor for the filter noise while the data/env noise stays the same at 1. The ANEES is very high for $r \leq 1$, but is very low $r = 4, 16, 64$. While the mean position error seems to be at its lowest when the filter noise is getting closer to the data/env noise.

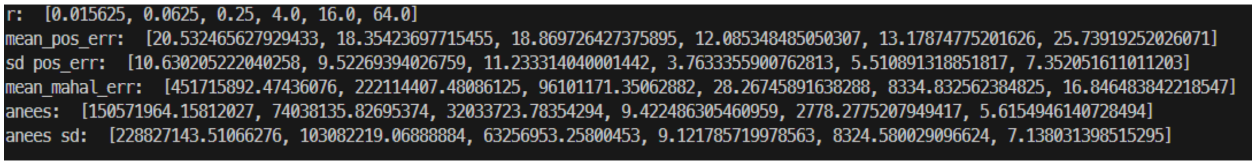
(d)



(a) ANEES

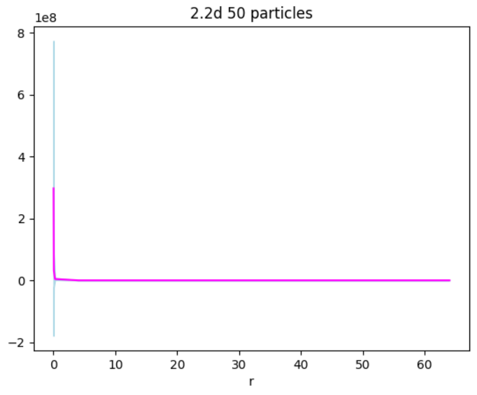


(b) position error

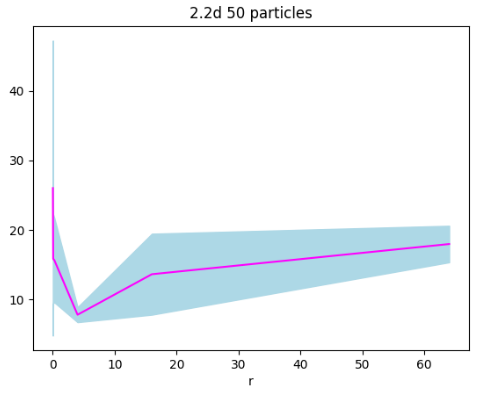


```
r:  [0.015625, 0.0625, 0.25, 4.0, 16.0, 64.0]
mean_pos_err:  [20.532465627929433, 18.35423697715455, 18.869726427375895, 12.085348485050307, 13.17874775201626, 25.73919252026071]
sd pos_err:  [10.630205222040258, 9.52269394026759, 11.233314040001442, 3.7633355900762813, 5.510891318851817, 7.352051611011203]
mean_mahal_err:  [451715892.47436076, 222114407.48086125, 96101171.35062882, 28.26745891638288, 8334.832562384825, 16.846483842218547]
anees:  [150571964.15812027, 74038135.82695374, 32033723.78354294, 9.422486305460959, 2778.2775207949417, 5.6154946140728494]
anees sd:  [228827143.51066276, 103082219.06888884, 63256953.25800453, 9.121785719978563, 8324.580029096624, 7.138031398515295]
```

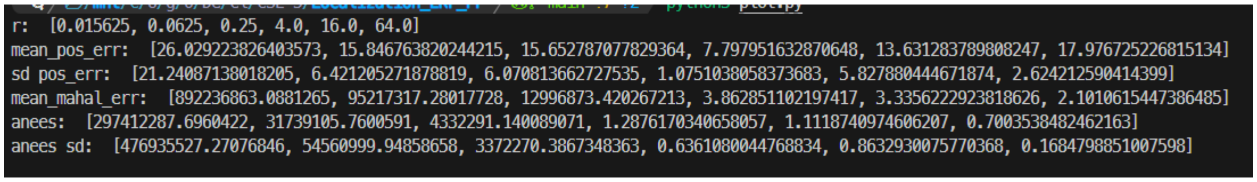
Figure 10: Data



(a) ANEES

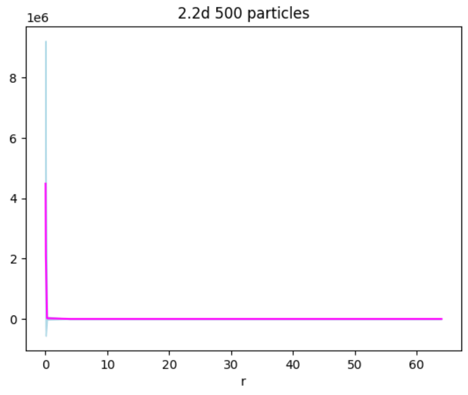


(b) position error

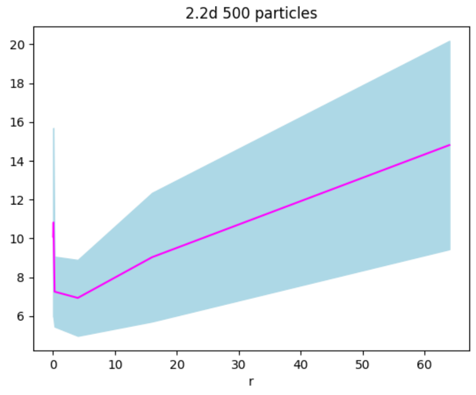


```
r:  [0.015625, 0.0625, 0.25, 4.0, 16.0, 64.0]
mean_pos_err:  [26.029223826403573, 15.846763820244215, 15.652787077829364, 7.797951632870648, 13.631283789808247, 17.976725226815134]
sd pos_err:  [21.24087138018205, 6.421205271878819, 6.070813662727535, 1.0751038058373683, 5.827880444671874, 2.624212590414399]
mean_mahal_err:  [892236863.0881265, 95217317.28017728, 12996873.420267213, 3.862851102197417, 3.3356222923818626, 2.1010615447386485]
anees:  [297412287.6960422, 31739105.7600591, 4332291.140089071, 1.2876170340658057, 1.1118740974606207, 0.7003538482462163]
anees sd:  [476935527.27076846, 54560999.94858658, 3372270.3867348363, 0.6361080044768834, 0.8632930075770368, 0.1684798851007598]
```

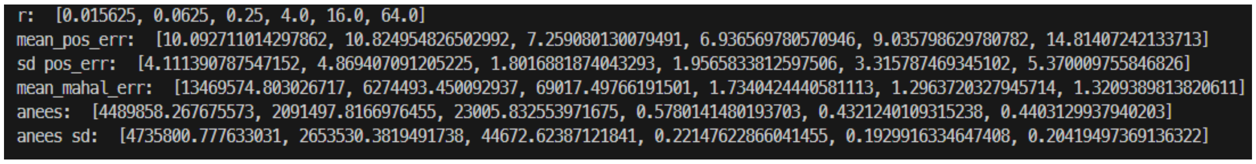
Figure 12: Data



(a) ANEES



(b) position error



```
r:  [0.015625, 0.0625, 0.25, 4.0, 16.0, 64.0]
mean_pos_err:  [10.092711014297862, 10.824954826502992, 7.259080130079491, 6.936569780570946, 9.035798629780782, 14.81407242133713]
sd pos_err:  [4.111390787547152, 4.869407091205225, 1.8016881874043293, 1.9565833812597506, 3.315787469345102, 5.370009755846826]
mean_mahal_err:  [13469574.803026717, 6274493.450092937, 69017.49766191501, 1.7340424440581113, 1.2963720327945714, 1.3209389813820611]
anees:  [4489858.267675573, 2091497.8166976455, 23005.832553971675, 0.5780141480193703, 0.4321240109315238, 0.4403129937940203]
anees sd:  [4735800.777633031, 2653530.3819491738, 44672.62387121841, 0.22147622866041455, 0.1929916334647408, 0.20419497369136322]
```

Figure 14: Data

Observations
As we add the number of particles, we can observe a decrease in the mean position error and ANEES for all r values. However, it does take slower to run. Thus, there is a trade-off between accuracy and runtime here.

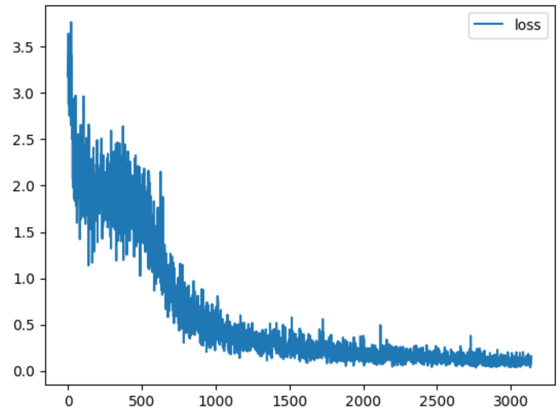
2.3 Learned Observation Model

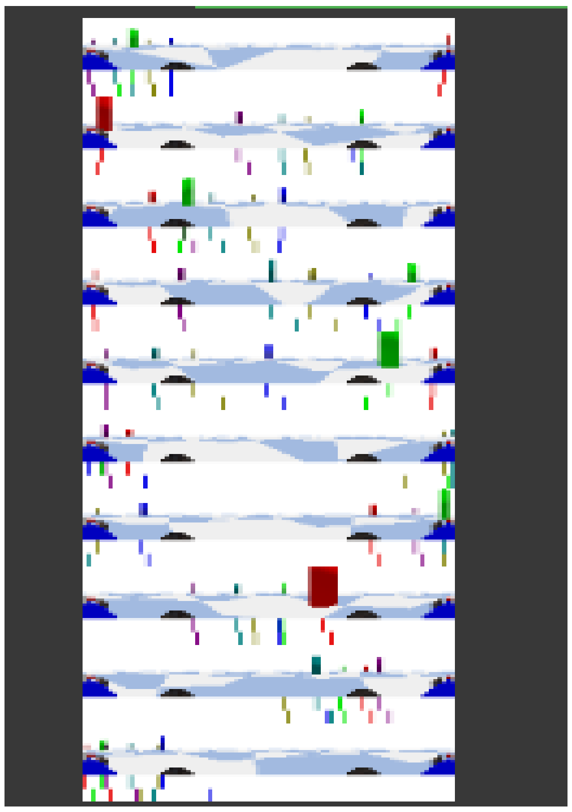
(a)

(b) I made no changes to the neural network model because I was planning to do so for part f.

(c)
(i):

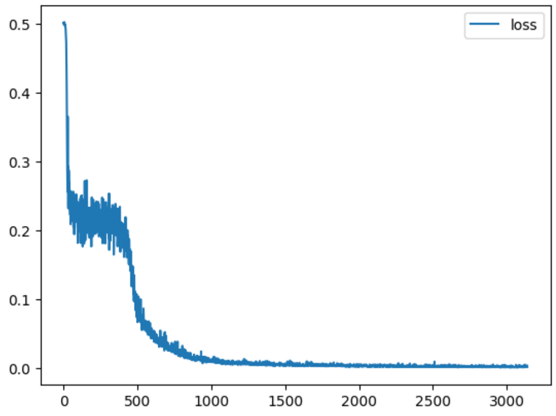
- Error Mean: 0.199187
- Error Std: 0.311930

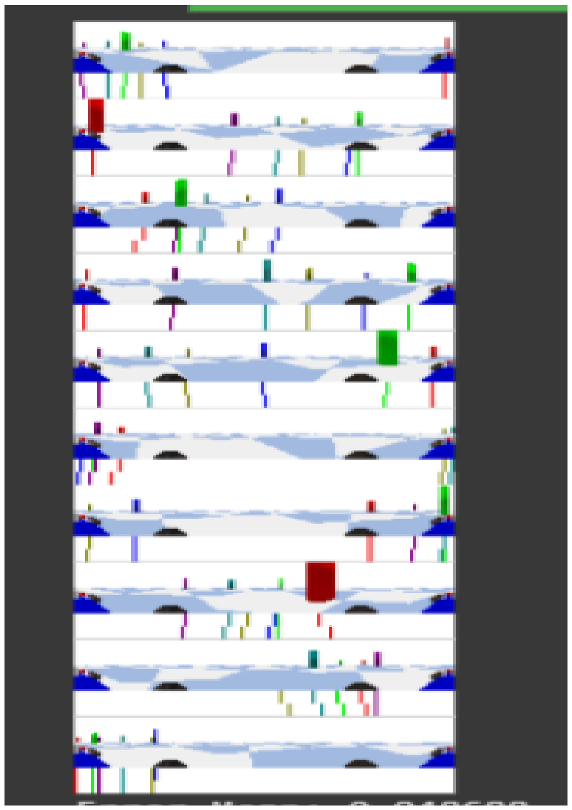




(ii):

- Error Mean: 0.048688
- Error Std: 0.069844





(iii): When there was more channels(supervision_mode xy) the error mean and std was lower. I think that this may be the case because we have more output channels, thus our neural network can capture more patterns in the images which might allow it to distinguish between each pillar and get better accuracy.

(d)
Seed = 0, Mean position error = 5.903169269460249
Seed = 1, Mean position error = 5.111413181231658
Seed = 2, Mean position error = 12.816086505385956
Seed = 3, Mean position error = 9.329680342949958
Seed = 4, Mean position error = 6.636527110898494
avg = 7.959375281985262

(e) It was really slow to train the network.

(f)
(1): Increased train dataset to 50000.

- phi
  - Error Mean: 0.127190
  - Error Std: 0.192099
- xy
  - Error Mean: 0.022366
  - Error Std: 0.052763
- Across 5 seeds, the average mean position error was 6.596321473951926.
- It does significantly better, we can see that the error mean is halved, the error std decreased, and the mean position error was lower by 1. However, it takes significantly slower.

(2): Network deeper and wider, I added another layer in the mlp where the last layer has 1024 nodes.

- phi
  - Error Mean: 0.203506
  - Error Std: 0.313932
- xy
  - Error Mean: 0.053962
  - Error Std: 0.068444
- Across 5 seeds, the average mean position error was 8.755470262515887.
- Similar performance as before.

(3): Decreasing learning rate. $3e-4 \rightarrow 3e-5$

- phi
  - Error Mean: 0.782177
  - Error Std: 0.760470
  - xy
    - Error Mean: 0.168213
    - Error Std: 0.159474
  - Across 5 seeds, the average mean position error was 18.17905193139991. Nearly tripled.
  - Performs worse, both errors mean and std increased.

(3): Increased learning rate. $3e-4 \rightarrow 8e-4$

- phi
  - Error Mean: 0.205520
  - Error Std: 0.270819
- xy
  - Error Mean: 0.036996
  - Error Std: 0.061209
- Across 5 seeds, the average mean position error was 6.376203013655655.
- Performs slightly better!

(4): Doubled epoch size. $20 \rightarrow 40$

- phi
  - Error Mean: 0.131065
  - Error Std: 0.246064
- xy
  - Error Mean: 0.030589
  - Error Std: 0.052381

- Across 5 seeds, the average mean position error was 5.428661459701464.
- Performs significantly better!

Observations
The modifications that were helpful were increasing the epoch size, increasing the training set, and also slightly increasing the learning rate(which helped a bit but was not as significant as the other two). Besides the learning rate, most of these changes had a trade-off between runtime and accuracy. Especially increasing the training set proved to have shown the most improved accuracy, however, it also took the longest to run. For the learning rate, I think that it's best to find the learning rate that didn't step too large, or too small(slow).