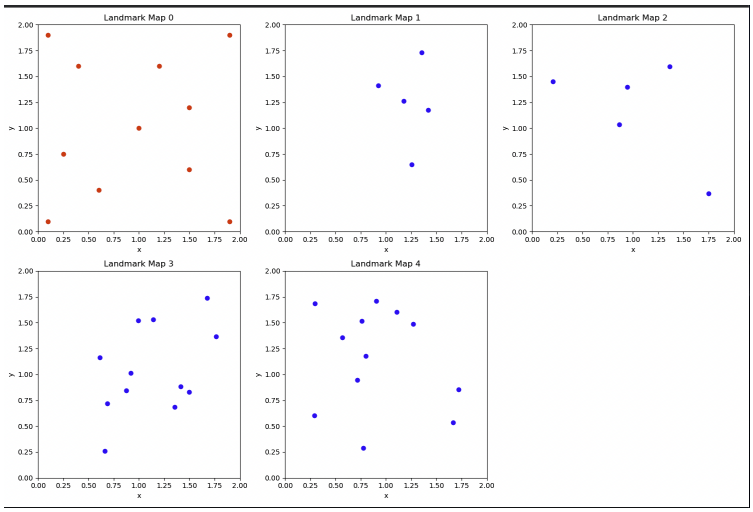
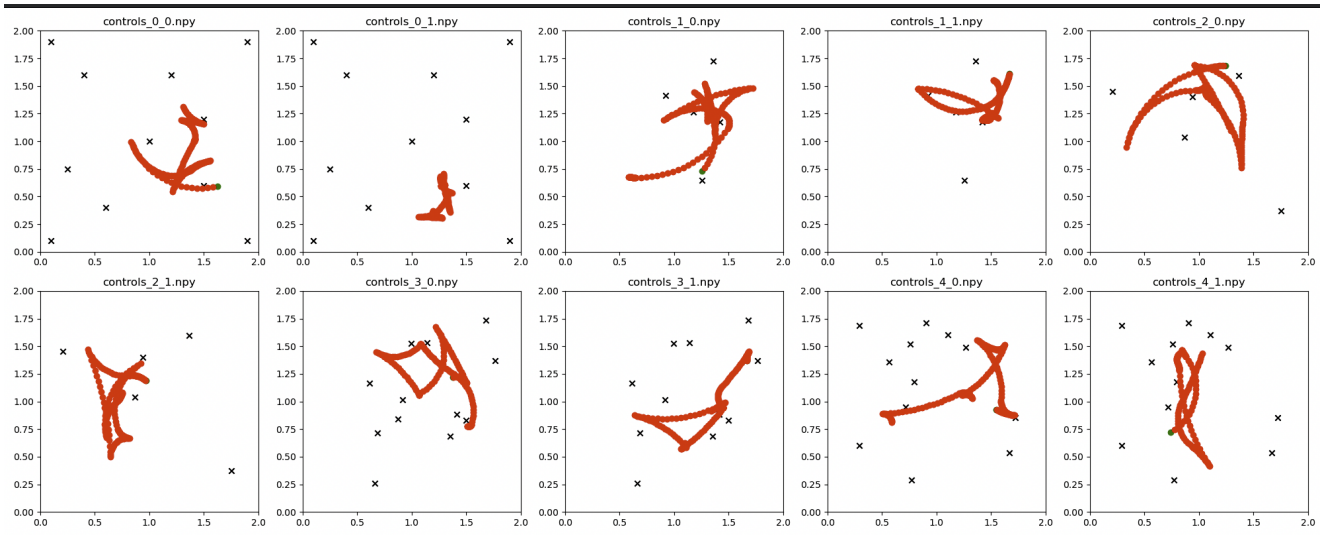
Joseph Jackson - JJ820
Satvik Srinivas - SS4144

Part 1

The control sequences are generated by initializing a random state and adding 20 random controls on top each of which are executed for 10 time steps each. If the sequence results in the robot going outside the valid zone and getting closer than 0.2 units to the boundary the sequence is discarded and the process begins again. This happens until a valid sequence is found. The visualizations are generated by starting with the robot in its initial state and integrating each control forward with the time step of 0.1 seconds following the motion model equations in the assignment.





Part 2

The landmark sensor is implemented by finding the true euclidean distance between the ground truth and the landmark as well as the true angle of the landmark with respect to the ground truth computed using the arctan2(y,x) function. Since the sensor has no range limits data is returned

for all landmarks and gaussian noise (mean 0, sd = 0.02) is added when it is interpreted independently for both the distance and angle reading.

The animations were developed by integrating forward and applying the sensed readings as well as plotting the ground_truth poses simultaneously. The effect of the noise seems to gradually accumulate as one would expect as both trajectories start off quite close together and end up drifting more and more apart. An interesting observation is that the sensor readings are generally in the right area but you can see that they are translated from the ground truth to location the readings indicate which is effectively the total error of the model.

Part 3

The particle filter was implemented using the SIR algorithm discussed in lecture. Each particle is weighted by the likelihood of the measurements given that particular particle pose which is a product of the individual likelihoods as we assume them to all be independent. In each iteration the particles are resampled according to the new distribution. We encountered significant difficulties with numerical stability but were able to address this by adding a small positive value to all the weights in an attempt to prevent particle deprivation.

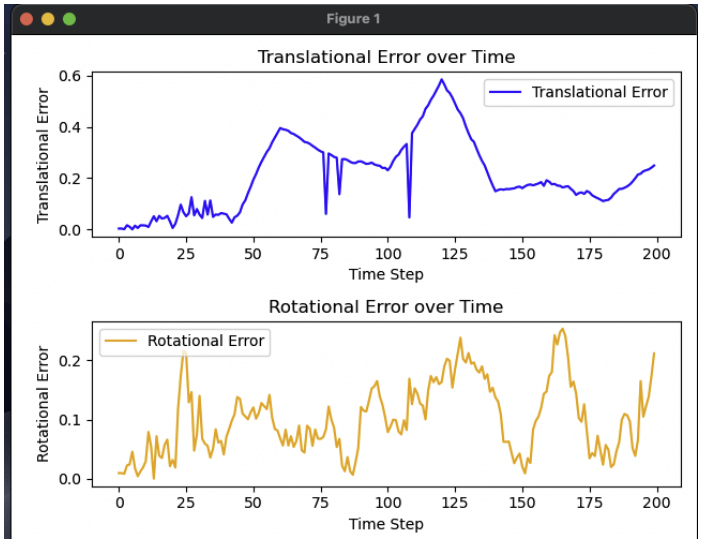
The effect of the noise seems to still be an issue but as the number of particles increases this effect is mitigated. Overall, the particles tend to get a better understanding of where the robot is as more time steps are integrated and more data is collected but this again is fundamentally dependent on the fact that particle deprivation is bypassed and the resulting distribution is a good representation of the true probabilistic distribution.

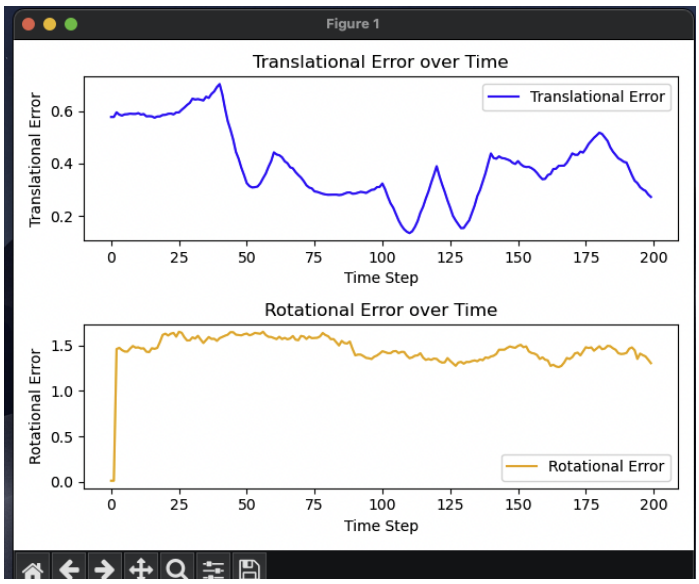
Part 4

1. Level of Noise in Readings:
Low Noise: We observed that low noise in sensor readings greatly enhances the reliability of these readings, subsequently increasing the accuracy of the particle filter. The particles tend to converge more accurately around the robot's true position since the measurements align closely with the expected values. This often results in fewer iterations required for convergence, which may slightly reduce the runtime.
High Noise: Conversely, high noise levels compromise the reliability of sensor readings, leading to a decrease in the filter's accuracy. The particles are more prone to converging at incorrect positions or may take longer to converge to the true position. This necessitates more iterations for convergence, thus potentially increasing the runtime.
2. Number of Particles:
Fewer Particles: With a smaller number of particles, the filter might inadequately represent the robot's state distribution. This underrepresentation can result in lower accuracy due to the limited information available to the algorithm. However, computationally, fewer particles lead to quicker computation per iteration, potentially reducing the overall runtime.
More Particles: Increasing the number of particles generally elevates the accuracy. A larger set of particles provides a more refined approximation of the robot's state probability distribution,

enhancing the filter's resilience against noise and uncertainties. However, this comes at the cost of increased computational workload per iteration, which can notably extend the runtime.
Comparison with Dead Reckoning:
Dead reckoning, which estimates position by integrating motion information over time without environmental feedback, is highly susceptible to cumulative errors. In contrast, particle filters, especially with a higher number of particles, tend to be more robust against noise. They continually adjust the position estimate based on sensor readings, offering an advantage over dead reckoning in environments with complex and noisy data.

Here are the 10 graphs that represent the errors. The first 5 are from the known location particle filter. The second 5 are from the kidnapped implementation.





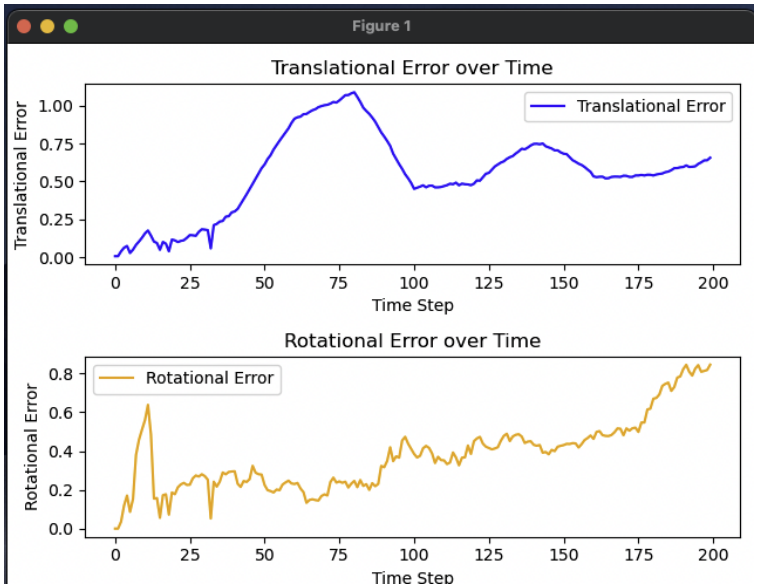


Figure 1
Translational Error over Time
Translational Error
Translational Error
1.00
0.75
0.50
0.25
0.00
0
25
50
75
100
125
150
175
200
Time Step
Rotational Error over Time
Rotational Error
Rotational Error
0.8
0.6
0.4
0.2
0.0
0
25
50
75
100
125
150
175
200
Time Step

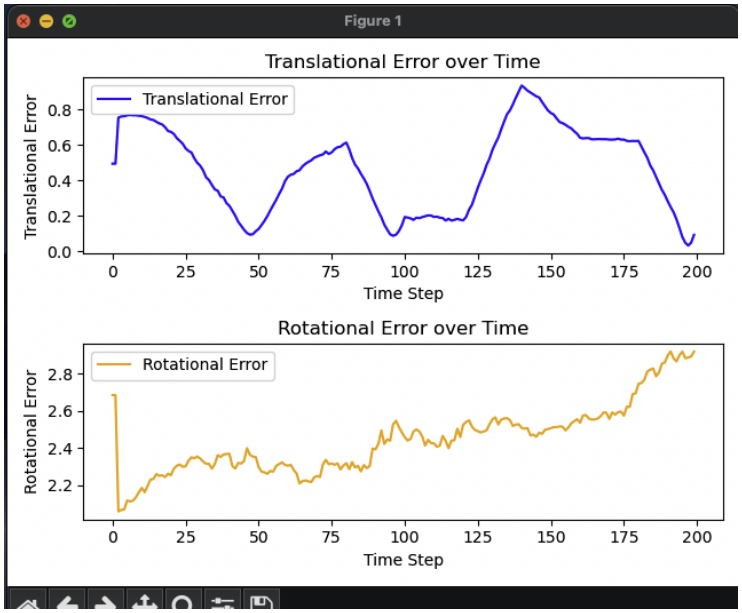


Figure 1
Translational Error over Time
Translational Error
Translational Error
0.8
0.6
0.4
0.2
0.0
0
25
50
75
100
125
150
175
200
Time Step
Rotational Error over Time
Rotational Error
Rotational Error
2.8
2.6
2.4
2.2
0
25
50
75
100
125
150
175
200
Time Step

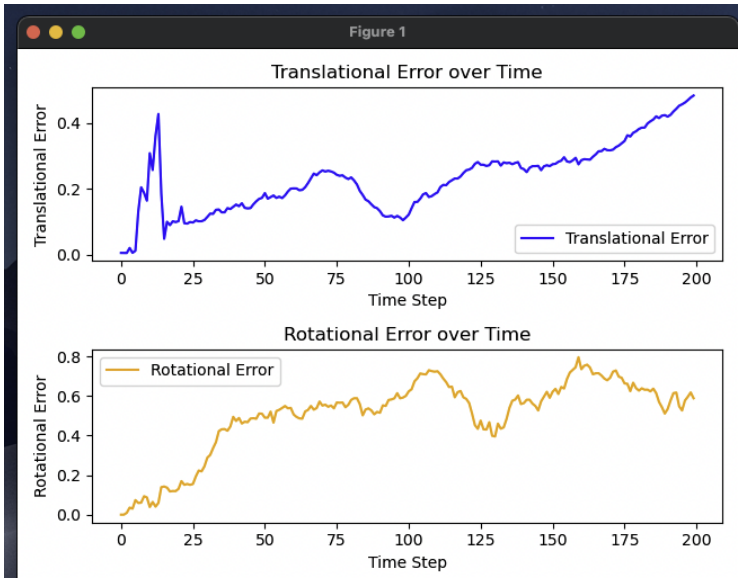


Figure 1
Translational Error over Time
Translational Error
Time Step
Translational Error
Rotational Error over Time
Rotational Error
Time Step
Rotational Error

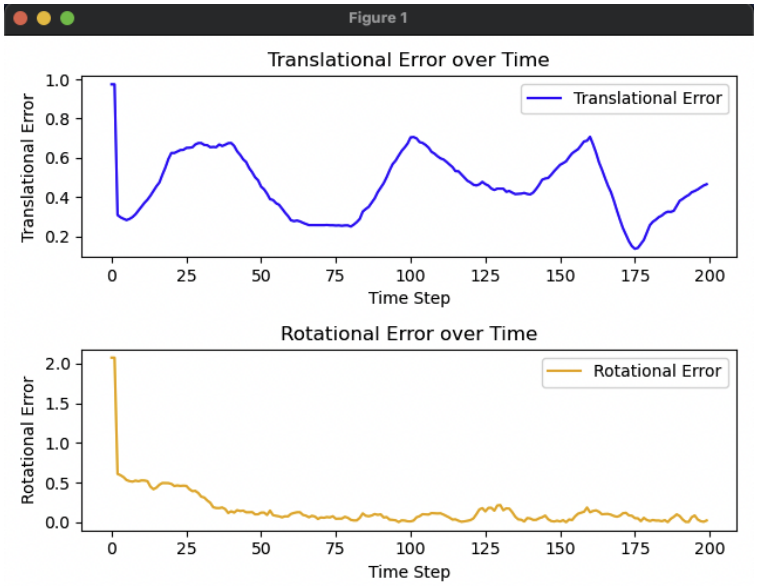


Figure 1
Translational Error over Time
Translational Error
Time Step
Translational Error
Rotational Error over Time
Rotational Error
Time Step
Rotational Error

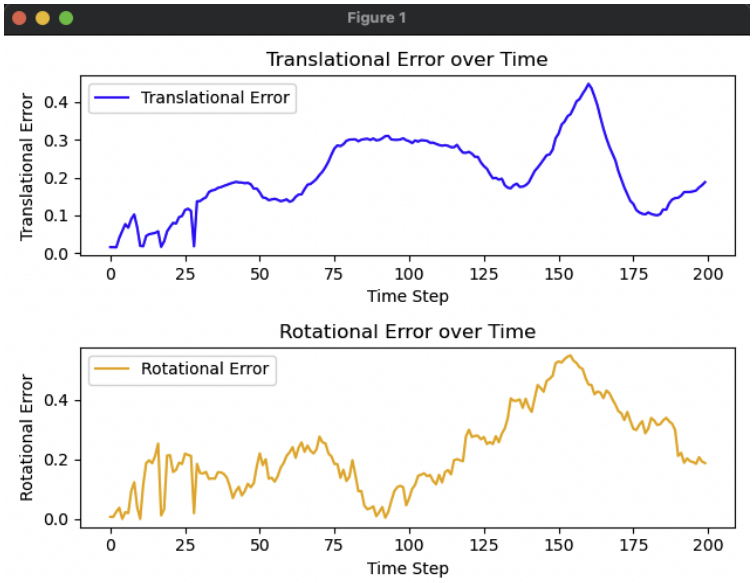


Figure 1
Translational Error over Time
Translational Error
Translational Error
Time Step
Rotational Error over Time
Rotational Error
Rotational Error
Time Step

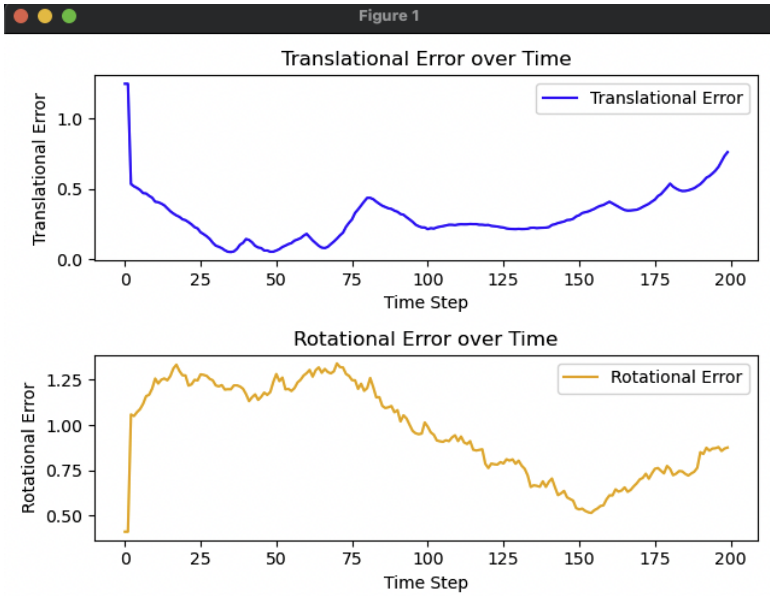


Figure 1
Translational Error over Time
Translational Error
Translational Error
Time Step
Rotational Error over Time
Rotational Error
Rotational Error
Time Step

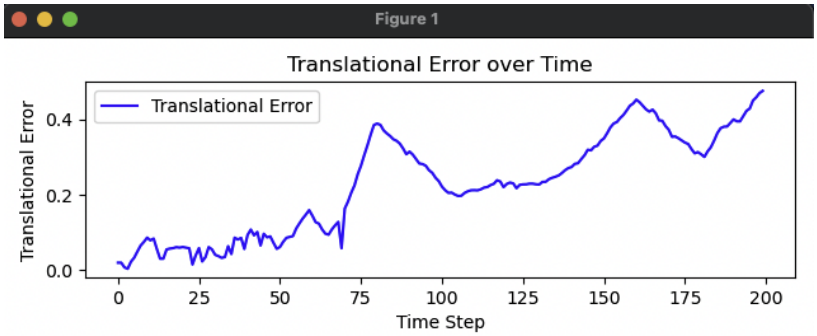
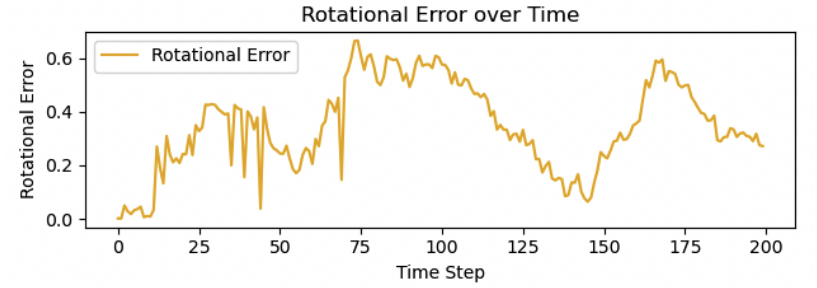


Figure 1
Translational Error over Time
Translational Error
Translational Error
0.0
0.2
0.4
0
25
50
75
100
125
150
175
200
Time Step


Rotational Error over Time
Rotational Error
Rotational Error
0.0
0.2
0.4
0.6
0
25
50
75
100
125
150
175
200
Time Step

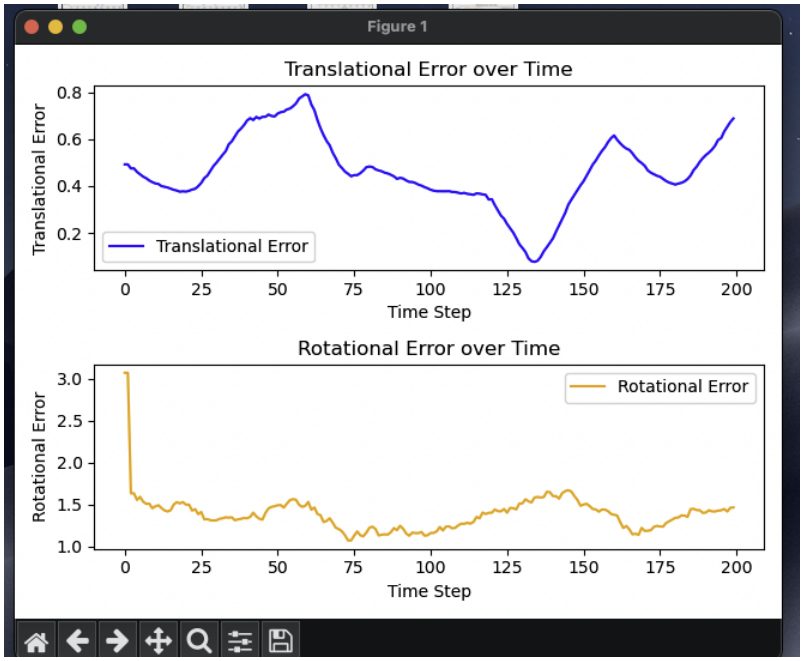


Figure 1
Translational Error over Time
Translational Error
Translational Error
0.2
0.4
0.6
0.8
0
25
50
75
100
125
150
175
200
Time Step
Rotational Error over Time
Rotational Error
Rotational Error
1.0
1.5
2.0
2.5
3.0
0
25
50
75
100
125
150
175
200
Time Step