**LAB 3**

**Team Members:**

Ritvik Goradia (Goradia3)

Aumkar Renavikar (aar8)

Devul Nahar (danahar2)

Toby Liang (tobyzl2)

**Problem 4 (20 points)**. Using 4 measurement directions, keep the sensor limit constant at 15, run your algorithm with the number of particles 500, 1000, 1500. Plot the error in position estimation (euclidean distance between actual position and the estimated position) and orientation estimation as a function of algorithm iterations. Since particle filtering is a randomized algorithm, run several instances of the same setup and plot the average error for each time. How does changing the number of particles influence the estimation accuracy, converging speed and computational cost of the algorithm? Record a video of one of the three runs. The video should include the turtle map window. Provide a link to the video and include it in the report.

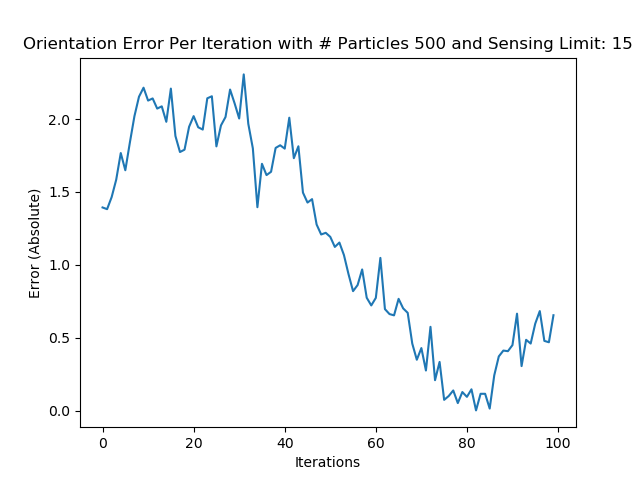
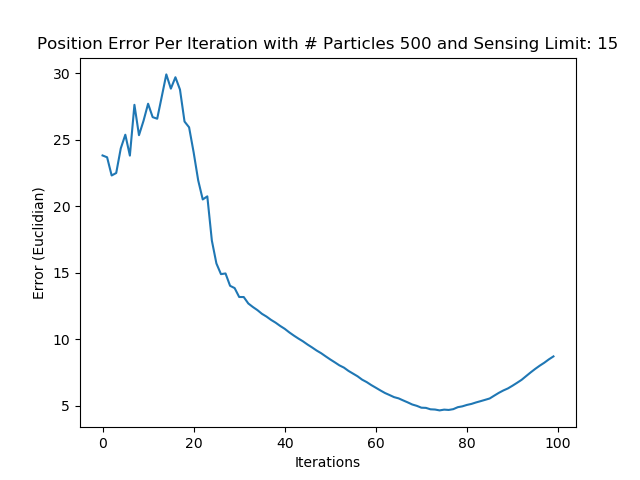
| Num\_particles,  sensing\_distance=15 |  | Position Error | Orientation Error |
| --- | --- | --- | --- |
| 500 | Trial1 | 77.8843840997 | 0.628835137829 |
| Trial2 | 4.34085777419 | 0.208750573632 |
| Trial3 | 12.3207014844 | 1.17079484435 |
| Average | 31.5153144528 | 0.66946018527 |
| 1000 | Trial1 | 17.404259332 | 2.15605767146 |
| Trial2 | 13.636669691 | 2.36438329571 |
| Trial3 | 15.732653656 | 1.11728441435 |
| Average | 15.591194227 | 1.87924179384 |
| 1500 | Trial1 | 6.36499451144 | 0.25225380372 |
| Trial2 | 25.2640681142 | 0.52913028856 |
| Trial3 | 20.9386116375 | 0.53659843740 |
| Average | 17.5225580878 | 0.43932750989 |

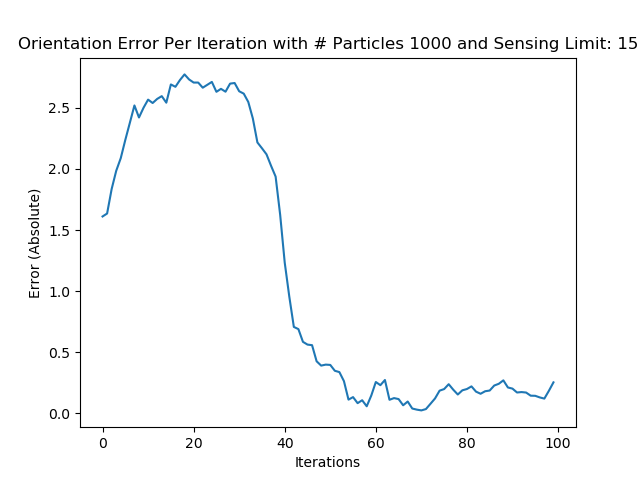
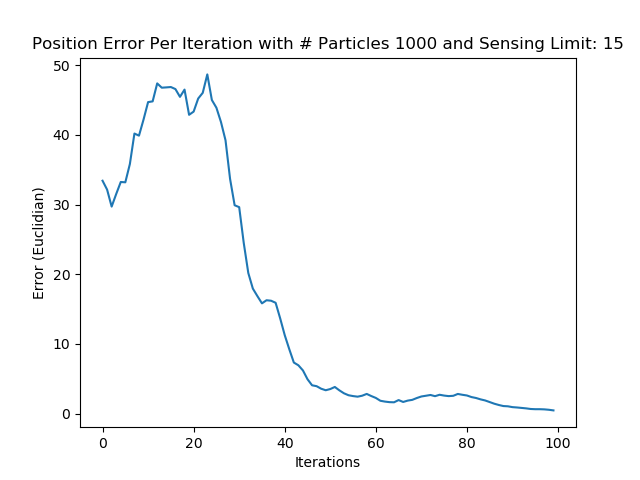
**Link to recording** for num\_particles = 1000, sensing\_distance = 15:

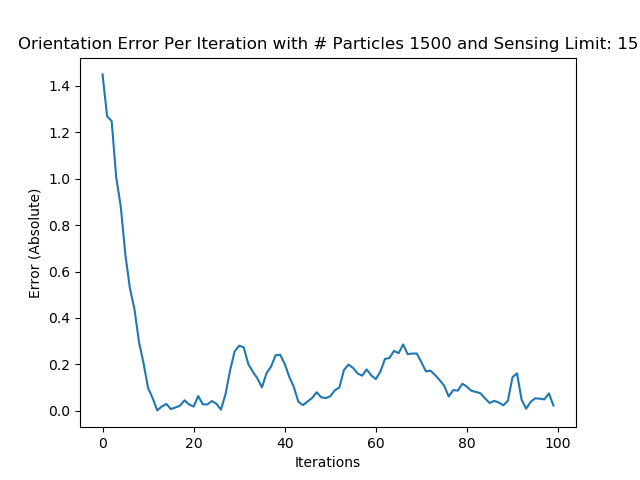
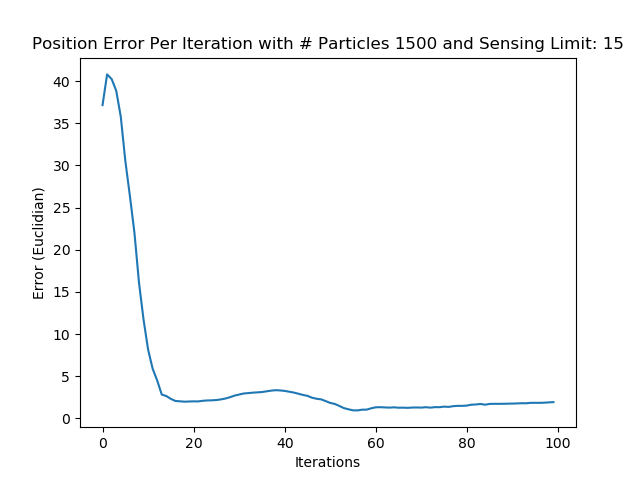
(Note: program exits after 100 iterations to print the resulting graph. Also Rviz may show 8 directions, however the program only uses 4)

<https://drive.google.com/file/d/1-tP1B-yOjjxxirsl7P6nQTDJSAhjZEfB/view?usp=sharing>

**Plots:**







From the graphs above, as the number of particles increases, the faster the error is able to converge. This makes sense since it's more likely that one of the initial particles from the uniform distribution is close to the robot position with the same orientation. This particle would have a larger weight and resampling would cause more particles near that position, which would result in faster convergence. From our trials, when the number of particles is 500, there is a larger average positional error since one of the trials did not converge. The 1000 and 1500 particle runs had similar average position error and the 1000 particle run had a significantly higher orientation error than the other two configurations. The computational cost would scale with the number of particles since most of the operations such as weight updates and resampling would need to iterate over the list of all particles. Thus, the 500 particle runs have the smallest computational cost and the 1500 particle runs have the largest computational cost.

**Problem 5 (20 points)**. Using 4 measurement directions, keep the number of particles constant at 1000, run your algorithm with sensor limit 15, 20, 25. Plot the error in position estimation (euclidean distance between actual position and the estimated position) and orientation estimation as a function of algorithm iterations. Since particle filtering is a randomized algorithm, run several instances of the same setup and plot the average error for each time. How does changing the sensor limit influence the estimation accuracy, converging speed and computation cost of the algorithm? Record a video of one of the three runs. The video should include the turtle map window. Provide a link to the video and include it in the report.

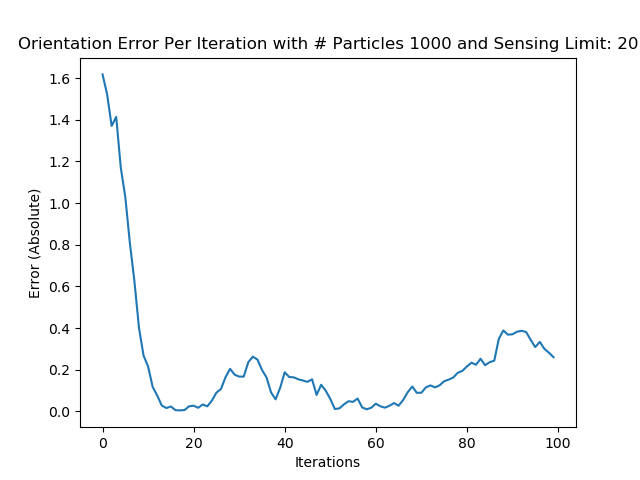
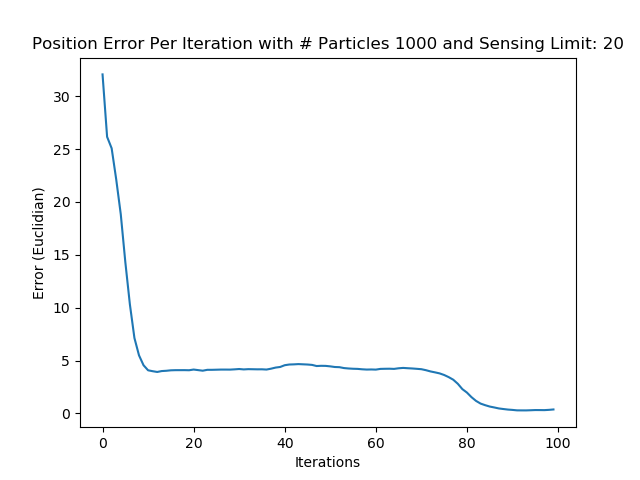
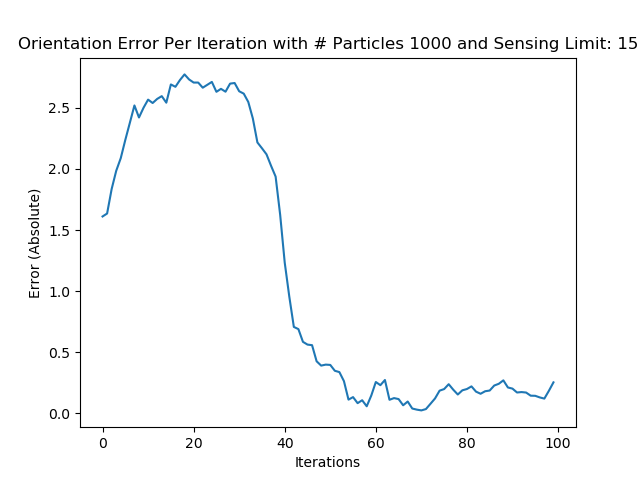
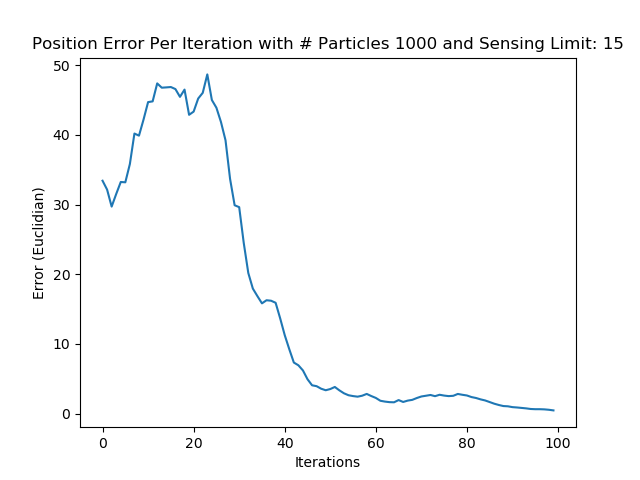
| Num\_particles = 1000  sensing\_distance |  | Position Error | Orientation Error |
| --- | --- | --- | --- |
| 15 | Trial1 | 17.404259332 | 2.15605767146 |
| Trial2 | 13.636669691 | 2.36438329571 |
| Trial3 | 15.732653656 | 1.11728441435 |
| Average | 15.591194227 | 1.87924179384 |
| 20 | Trial1 | 8.30356993364 | 0.30610023167745 |
| Trial2 | 31.1622183551 | 0.48479573695079 |
| Trial3 | 4.67984623523 | 0.23079820510440 |
| Average | 14.7152115079 | 0.34056472457755 |
| 25 | Trial1 | 14.213446554071 | 1.3885226917049 |
| Trial2 | 13.803720887580 | 1.4993112150025 |
| Trial3 | 7.2164475423447 | 0.5480786636749 |
| Average | 11.744538327999 | 1.1453041901275 |

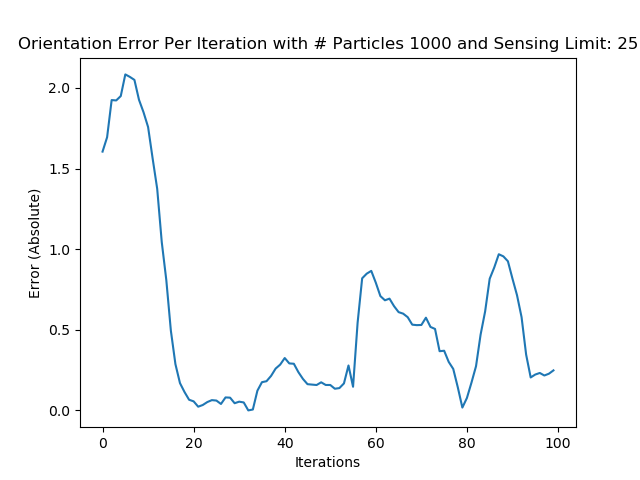
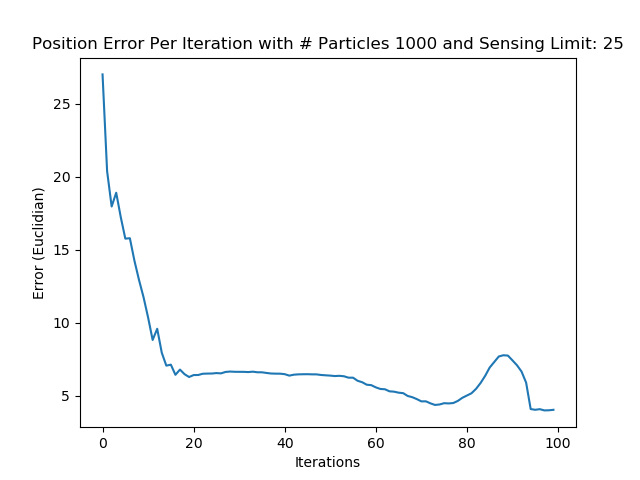
**Link to recording** for num\_particles = 1000, sensing\_distance = 20:

(Note: program exits after 100 iterations to print the resulting graph. Also Rviz may show 8 directions, however the program only uses 4)

<https://drive.google.com/file/d/15qmbNtcU9x7pNkTrLBgOyuie49G4Go8a/view?usp=sharing>

**Plots:**





As seen through the graphs above, as the sensing limit is increased, the average position error decreases, ensuring that the error converges quickly. This makes sense because the greater the sensing distance, the more obstacles our car is able to detect, and hence is able to make better predictions about its current position based on the information that it has available to it. In terms of the orientation error, it was the greatest for the least sensing limit, and was lower for the highest sensing limit. However, for the sensing limit of 20, there seems to be errors in randomization based on the seeds selected which yields very low errors for the orientation error. The computation cost would not vary greatly between the runs other than any additional time to collect the data from the sensors. This is because most of the operations are based on iteration through the particles as well as performing the kernel similarity, which only depends on the number of measurements.

**Problem 6 (10 points)**. Does the particle filter you implemented performs evenly well through the whole environment after converging? More specifically, does your particle filter have larger prediction errors in some regions of the environment than other regions? If yes, can you explain why this is happening?

At the very start of each run, the particle filter has the most error since it is still in the process of converging. The particles are often in multiple clusters at the very first iterations before converging to a point close to the true position. This is because the particles are initially distributed uniformly on the map and multiple clusters will form in areas that are in similar spaces as the car. At the first turn, the particles usually make a tighter turn but in the first corridor, the estimation position almost matches exactly with the true position. The particles continue to closely follow the car for the rest of the track as all the points have converged and simply follow the car's motion model.

**Problem 7 (10 points)**. Modify the LidarProcessing module and the sensor model so that they can make measurements in 8 directions. Run your algorithm with a number of particles 1000 and sensor limit 20. Plot the error in position estimation (euclidean distance between actual position and the estimated position) and orientation estimation as a function of algorithm iterations. Since particle filtering is a randomized algorithm, run several instances of the same setup and plot the average error for each time. How does having more sensor data influence the estimation accuracy and converging speed of the algorithm? Record a video of the run. Besides the turtle map window, the video should include the RViz window of the sensor measurements. Provide a link to the video and include it in the report.

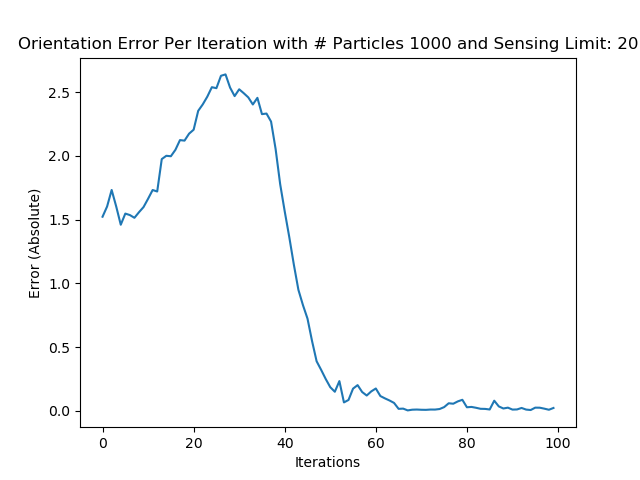
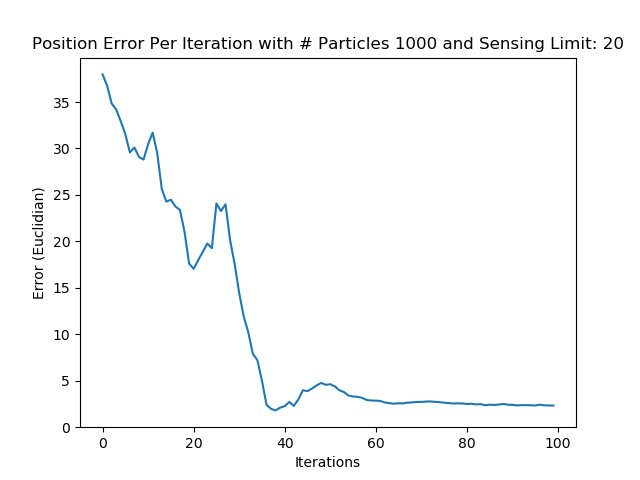
| Num\_particles,  sensing\_distance=20 |  | Position Error | Orientation Error |
| --- | --- | --- | --- |
| 1000 | Trial1 | 5.13038948813 | 0.29658147033 |
| Trial2 | 5.63997873921 | 0.28912275230 |
| Trial3 | 10.1665124567 | 0.94031772304 |
| Average | 6.97896022805 | 0.50867398189 |

**Link to recording** for num\_particles = 1000, sensing\_distance = 20:

(Note: program exits after 100 iterations to print the resulting graph)

<https://drive.google.com/file/d/1BVaLkBB1Zv8HOxdSbIV1X_hD17k-RLy5/view?usp=sharing>

**Plots:**



Note that using all 8 directions gives us the smallest average error in all variations in this lab. This means all 8 directions provide more vital information to the controller to make the decision of when and how to move the car since it is able to detect obstacles in all 8 directions instead of being limited to the conventional 4 directions, meaning it is more accurately able to describe its position on the map. The average orientation error is also quite low, meaning that the overall program converges quite quickly. The convergence time is a bit slower (about 50 iterations) when compared to the run with 4 measurements and a sensing limit of 20 (about 10 iterations), but the accuracy after convergence is higher, evidently seen through a lower error as compared to the 4 directions. There might also be higher computation costs since we are sensing in 8 directions instead of 4 and would require more time to compute the similarity between the particle sensor and the vehicle lidar using the gaussian kernel.

Code: https://drive.google.com/drive/folders/177X3KCewSToCzthWi-i0\_O\_O-UiPegg1?usp=share\_link