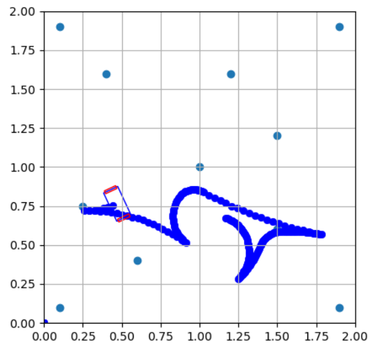
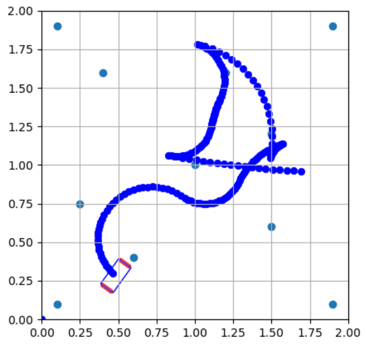
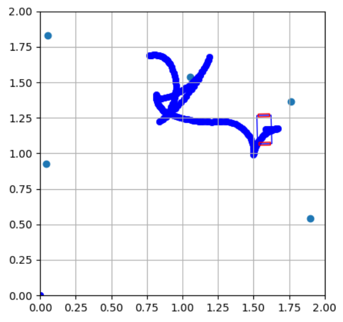
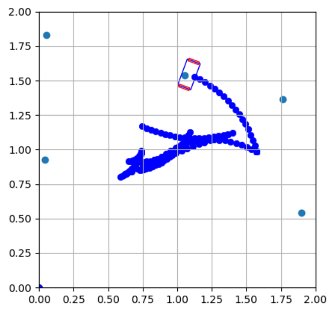
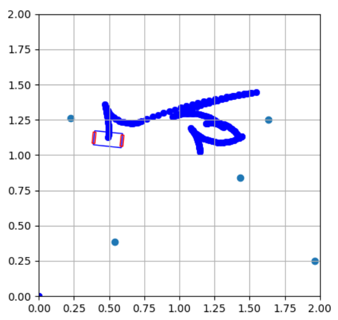
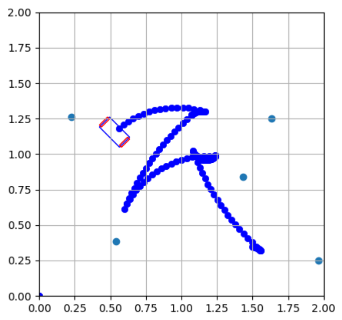
Adam Korczak ak1724
Nii Shidaa Adjei naa167

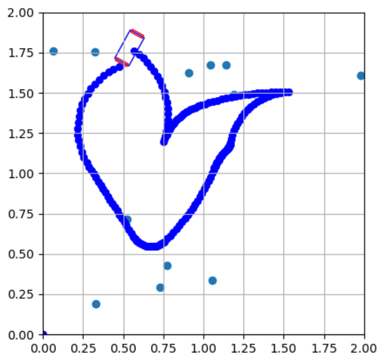
1. Setting up maps and generating controls

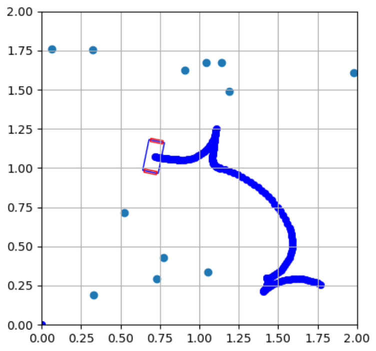
The controls were by randomly sampling values in the given bounds, then testing to see if they would keep the robot in bounds before adding them to the finalized list of controls. I used the setup of the robot from recitation 6 to visualize the motion of the robot and merely adjusted it to rely on the predetermined controls to move, rather than key input. I then added the path by plotting points generated by the x,y value of the center of the robot at each pose.

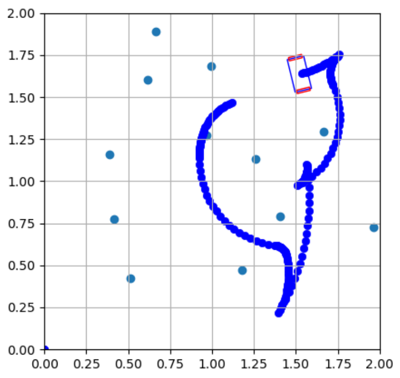





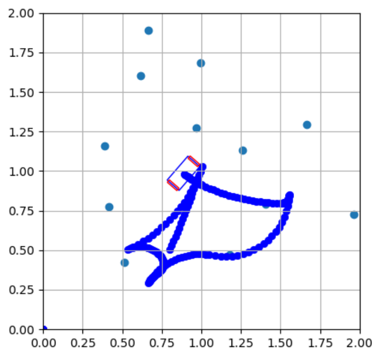












2. **Integrate noisy dynamics and collect sensor readings**

   For the landmark sensor, I implemented a function of the same format as the suggestion provided in the write up, and I used the distance formula to determine the distance between the landmarks and the ground truth position. I then used np.arctan2() to determine the angular difference between the robot and the landmark point. I added the odometry_measurements and the observation_measurements to 2 different lists and appended them alternately into another list which I saved into the readings file using dtype=object.

3. Implement Particle Filter for localization

For the particle filter, particle are initialized with the robot's initial pose. Each particle represents a hypothesis of the robot's pose. Each particle updates based on the robot's motion model. The function applies control inputs velocity and angular velocity to each particle. The weight function calculates the weight of each particle based on how well the particle's pose matches the sensor readings, given the positions of the landmarks. Particle_filter updates their positions using the motion model and calculates weights using the sensor model and then normalizes the weights. From there particles are sampled again based off these weights and focusing on particles that best match the sensor readings. After resampling, it calculates the mean estimate of the poses which represents the estimated pose of the robot. Then for each frame of the animation, we visualize the current state of the particles, odometry, and sensor readings for each resample. For the kidnapped robot problem, we determine the robot's position using only sensor readings and the environment map with an initialization of the particles done randomly across the space as the robot pose is unknown to begin with.

4. Evaluation and Experiments

The evaluation poses the ground truth from the execution file and the estimated poses from the particle filter output. From here the file visualizes the ground truth path in blue and the estimated path from the particle filter in black and plots the landmarks in blue dots. We compute the errors between the estimated poses and ground truth including the translational errors and rotational errors and see the difference in paths with both high and low noise and different frequencies of particles. We found that in low noise environments we had more reliable sensor readings and with a higher number of particles we had a higher accuracy with the particle filter with the paths being closer and more consistent.