Mini Project 1: Localization and Mapping

Patrick Donelan
661779804
ECSE 4490
Project Cluster 1: MATLAB

18 February 2021

1 Abstract

This project serves as hands-on experience with localization and mapping for a mobile robot using various sensor measurements and the extended Kalman filter. The state estimation and mapping results were analyzed under the effects of different initial state covariances, process noise covariances, and observation noise covariances in order to study their effect. It was found that high observation noise covariance had a much more significant effect on localization than high initial state covariance and process noise covariance. For mapping, high observation noise covariance had a drastic effect and high initial state covariance had a weaker but still significant effect. This project also lays the foundation for the next evolution - path planning.

2 Problem Statement

The goal of this project is to use MATLAB to implement localization and mapping simulation for a simple rectangular robot in the room portrayed by Figure 1.

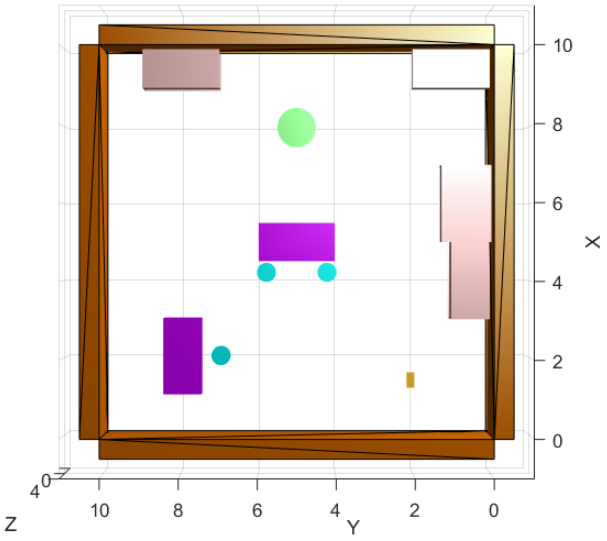

Figure 1: Environment

The robot and environment are created using MATLAB collision bodies from the Robotics System Toolbox. A variety of different sensors are available for the simulation, such as a proximity sensor, an inertial measurement unit (IMU), local GPS range sensors, a bearing sensor, and a lidar. In order to mimic the real world, the sensors have associated observation noise. Not only do the sensors

have noise, but the robot itself is also assumed to have process noise. Noise is added to the sensor and robot models in order to better mimic the real world where noise is inevitable.
One application of SLAM is robot vacuums. Some robot vacuums simply drive until they crash, rotate, then drive forward again repeatedly. This algorithm, as demonstrated later in this report, is an inefficient way to traverse a room. The performance of the robot could be increased using localization and mapping. If a robot vacuum knows where it is and where it has been, it can avoid those same locations and focus on cleaning new areas. With mapping, the robot knows where obstacles are and can therefore avoid situations where it might get stuck. SLAM is also useful for self-driving cars, which could use GPS, a compass, and an IMU for localization as well as proximity sensors and lidar for the mapping of other vehicles, street signs, houses, etc. Using these sensor readings, the autonomous car can create an efficient travel path that is also safe from collisions with other obstacles in the environment.

3 Approach

3.1 Localization

In the localization problem, the goal is to estimate the state

$$q = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix}$$

based on a state transition function and state observations from sensors.

3.1.1 State Transition

The state transition function

$$q_{k+1} = f(q_k, u_k)$$

uses the previous state q_k and the control vector

$$u = \begin{bmatrix} v_{linear} \\ v_{angular} \end{bmatrix}$$

to determine the current state q_{k+1}. The state transition function allows returns a transition matrix

$$F = \nabla_q f(q_k, u_k)$$

For a mobile robot, the state transition function is defined as

$$f(q_k, u_k) = q_k + \begin{bmatrix} \cos q_k(3) & 0 \\ \sin q_k(3) & 0 \\ 0 & 1 \end{bmatrix} u_k * t_s + w$$

where t_s is the sampling time and w is zero mean additive Gaussian process noise with covariance matrix Q. The state transition matrix is therefore

$$F = \begin{bmatrix} 1 & 0 & -\sin q_k(3) * u_k(1) * t_s \\ 0 & 1 & \cos q_k(3) * u_k(1) * t_s \\ 0 & 0 & 1 \end{bmatrix}$$

3.1.2 State Observation

A crucial consideration for the state observation calculations is not all sensors will be used at a given observation time. For example, if the nearest obstacle is out of range from the proximity sensor, there will be no proximity sensor observation. Thus, it is important to keep track of which sensors are used to obtain the observation vector. Each sensor, if used, returns an observation vector

$$z = h(q)$$

and an observation matrix

$$H = \nabla_q h(q)$$

3.1.3 Proximity Sensor

The proximity sensor observes the closest distance between a point on the robot's surface and a point on an obstacle's surface.

$$z = \min_i \|P_i - P_r\| = \min_i \left\| \begin{bmatrix} P_i(1) - P_r(1) \\ P_i(2) - P_r(2) \\ P_i(3) - r_z \end{bmatrix} \right\| = \|r\|$$

$$H = \begin{bmatrix} \frac{-r(1)}{\|r\|} & \frac{-r(2)}{\|r\|} & 0 \end{bmatrix}$$

where P_i is the location of the point on the obstacle's surface, P_r is the location of the point on the robot's surface, and r_z is the height of the center of the robot. P_r can be thought of as

$$P_r = \begin{bmatrix} q(1) \\ q(2) \\ r_z \end{bmatrix} + \Delta$$

where Δ is the offset of the surface point from the center. Note if z is greater than the specified proximity range, then the proximity sensor is not used.

3.1.4 IMU Acceleration

IMU acceleration can be observed using the history of the input variable or using the history of the state variable. For this project, it was decided to calculate

linear acceleration based on previous states as it yields more information for estimation.

$$z = \frac{1}{t_s^2} \left\| \begin{bmatrix} (q_k(1) - q_{k-1}(1)) - (q_{k-1}(1) - q_{k-2}(1)) \\ (q_k(2) - q_{k-1}(2)) - (q_{k-1}(2) - q_{k-2}(2)) \end{bmatrix} \right\| = \frac{1}{t_s^2} \|a\|$$

$$H = \begin{bmatrix} \frac{a(1)}{t_s^2 \|a\|} & \frac{a(2)}{t_s^2 \|a\|} & 0 \end{bmatrix}$$

Note if $k < 3$, then the IMU acceleration sensor is not used.

3.1.5 IMU Angular Velocity

Similar to the IMU acceleration, the IMU angular velocity can be observed using the input variable history or the state variable history. Again, the state variable history is used for the measurement.

$$z = \frac{1}{t_s}(q_k(3) - q_{k-1}(3))$$

$$H = \begin{bmatrix} 0 & 0 & \frac{1}{ts} \end{bmatrix}$$

Note if $k < 2$, then the IMU angular velocity sensor is not used.

3.1.6 Local GPS

For this project, there are four range sensors in the upper corners of the room that observe the distance to the center of the robot.

$$z_i = \left\| P_i - \begin{bmatrix} q(1) \\ q(2) \\ r_z \end{bmatrix} \right\| = \left\| \begin{bmatrix} P_i(1) - q(1) \\ P_i(2) - q(2) \\ P_i(3) - r_z \end{bmatrix} \right\| = \|r_i\|$$

$$H_i = \begin{bmatrix} \frac{-r_i(1)}{\|r_i\|} & \frac{-r_i(2)}{\|r_i\|} & 0 \end{bmatrix}$$

where P_i is the position of GPS sensor $i = 1, 2, 3, 4$. Note if z_i if greater than the specified GPS range, then the corresponding sensor is not used.

3.1.7 Bearing Sensor

The bearing sensor observes the relative angle between the robot heading and the large cylinder in the room.

$$z = \arctan \frac{P(2) - q(2)}{P(1) - q(1)} - q(3) = \arctan \frac{\Delta_y}{\Delta_x} - q(3)$$

$$H = \begin{bmatrix} \frac{-\Delta_y}{\Delta_x^2 + \Delta_y^2} & \frac{\Delta_x}{\Delta_x^2 + \Delta_y^2} & -1 \end{bmatrix}$$

3.1.8 Lidar

The lidar sensor observes the closest distance from the robot to an obstacle in M equally spaced directions. The implementation of the lidar sensor is not as trivial as the other sensors. Rays are created as very thin cylinder collision objects and rotated to point in the desired direction. Then, a collision check is performed for each obstacle with the ray. If there is no collision, the ray length is increased. If there is a collision, the length of the ray is decreased. This is done in a binary search manner due to its efficiency. The lidar sensor proved to be an issue, which will be discussed in the results section.

$$z_i = \left\| P_i - \begin{bmatrix} q(1) \\ q(2) \\ r_z \end{bmatrix} \right\| = \left\| \begin{bmatrix} P_i(1) - q(1) \\ P_i(2) - q(2) \\ P_i(3) - r_z \end{bmatrix} \right\| = \|r_i\|$$

$$H_i = \begin{bmatrix} \frac{-r_i(1)}{\|r_i\|} & \frac{-r_i(2)}{\|r_i\|} & 0 \end{bmatrix}$$

where P_i is the location on the nearest obstacle's surface in the direction of angle $i = 1, 2, ..., M$ from the robot's center.

3.1.9 Extended Kalman Filter

$$\begin{aligned} q_{pred} &= f(q_{est}, u) \\ P_{pred} &= F * P_{est} * F^T + q \\ z_{pred} &= h(q_{pred}) \\ y &= z_{true} - z_{pred} \\ S &= H * P_{pred} * H^T + R \\ K &= P_{pred} * H^T * S^{-1} \\ q_{est} &= q_{pred} + K * y \\ P_{est} &= (I - K * H) * P_{pred} \end{aligned}$$

Note z_{true}, z_{pred}, H, and R are indexed such that the same sensors are used for the true measurement as the estimated measurement (i.e. if there is no proximity measurement for the true position but there is for the estimated measurement, the proximity measurement is removed from z_{pred}, H, and R).

3.2 Simultaneous Localization and Mapping

SLAM is performed using the Extended Kalman Filter by adding the range and bearing sensor positions to the state vector

$$x = \begin{bmatrix} q & P_{range,1} & ... & P_{range,4} & P_{bear} \end{bmatrix}^T$$

where q is the robot state, $P_{range,i}$ is the $\begin{bmatrix} x & y & z \end{bmatrix}$ position of the ith range sensor, and P_{bear} is the $\begin{bmatrix} x & y & z \end{bmatrix}$ position of the bearing sensor. Since the state vector is extended, the state covariance matrix has to be expanded as well.

3.2.1 State Transition

The transition of q is the same as the transition used in the localization only problem. The landmarks are assumed to be stationary and thus the state transition matrix is

$$F = \begin{bmatrix} \begin{bmatrix} 1 & 0 & -\sin q_k(3) * u_k(1) * t_s \\ 0 & 1 & \cos q_k(3) * u_k(1) * t_s \\ 0 & 0 & 1 \end{bmatrix} & 0 \\ 0 & I_{15} \end{bmatrix}$$

3.2.2 State Observation

Localization is performed assuming the location of the landmarks is known and the state of the robot is unknown, hence the state observation matrix H only treats q as a variable when computing the gradient. Mapping is performed assuming the state of the robot is known and the landmark locations are unknown, hence the state observation matrix H treats the landmark locations as variables when computing the gradient. For SLAM, the observation matrix needs to be computed treating q as a variable with all landmark positions fixed and by treating the landmark positions as variables with the robot state fixed. Four range sensors and one bearing sensor are used for SLAM. A 2D map of the room is created using lidar measurements along with the the estimated state of the robot.

3.2.3 Local GPS

The new observation matrix for the local GPS is

$$H_i = \begin{bmatrix} \frac{-r_i(1)}{\|r_i\|} & \frac{-r_i(2)}{\|r_i\|} & 0 & \dots & \frac{r_i(1)}{\|r_i\|} & \frac{r_i(2)}{\|r_i\|} & \frac{r_i(3)}{\|r_i\|} & \dots \end{bmatrix}$$

where the new non-zero elements occur at $\begin{bmatrix} 4 & 5 & 6 \end{bmatrix} + 3(i-1)$ where i is the index of the range sensor being used. All other elements of H_i are zero except for the ones shown.

3.2.4 Bearing Sensor

The new observation matrix for the bearing sensor is

$$H = \begin{bmatrix} \frac{-\Delta_y}{\Delta_x^2+\Delta_y^2} & \frac{\Delta_x}{\Delta_x^2+\Delta_y^2} & -1 & \dots & \frac{\Delta_y}{\Delta_x^2+\Delta_y^2} & \frac{-\Delta_x}{\Delta_x^2+\Delta_y^2} & 0 \end{bmatrix}$$

where the new non-zero elements occur at $\begin{bmatrix} 16 & 17 & 18 \end{bmatrix}$. All other elements of H are zero except for the ones shown.

3.2.5 Extended Kalman Filter

The algorithm for the EKF is the same for SLAM as it is for localization.

3.3 Path Planning

The path planning algorithm currently implemented is very trivial. First, the robot attempts to move forward. If it cannot move forward, then the robot tries to rotate clockwise. If the robot cannot rotate clockwise, it tries to drive backwards. The robot gives up and the simulation ends if the robot cannot move backwards.

4 Results

4.1 Localization

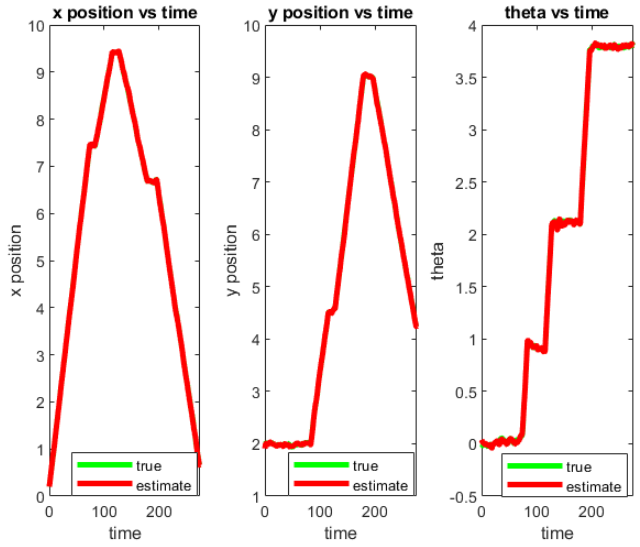

Figure 2: Localization

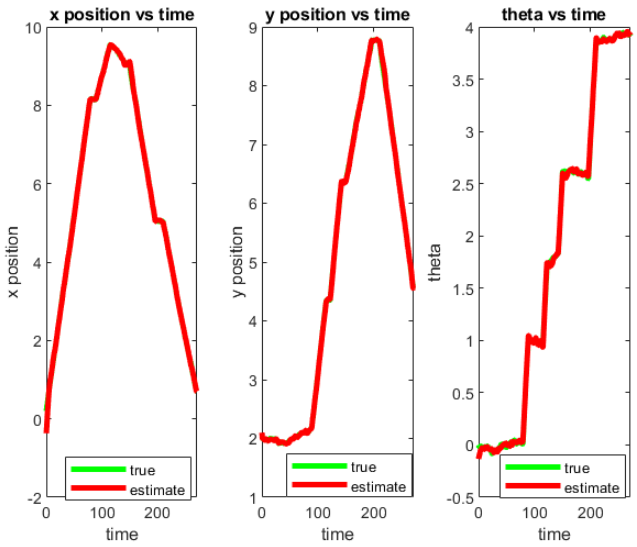

Figure 3: Localization with High Initial P

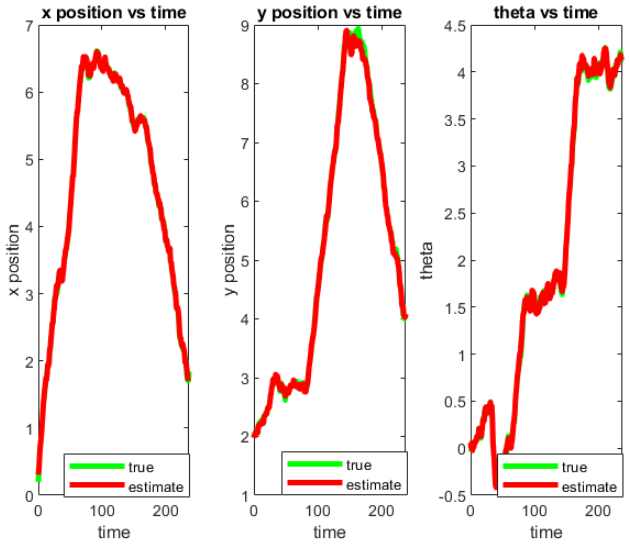

Figure 4: Localization with High Q

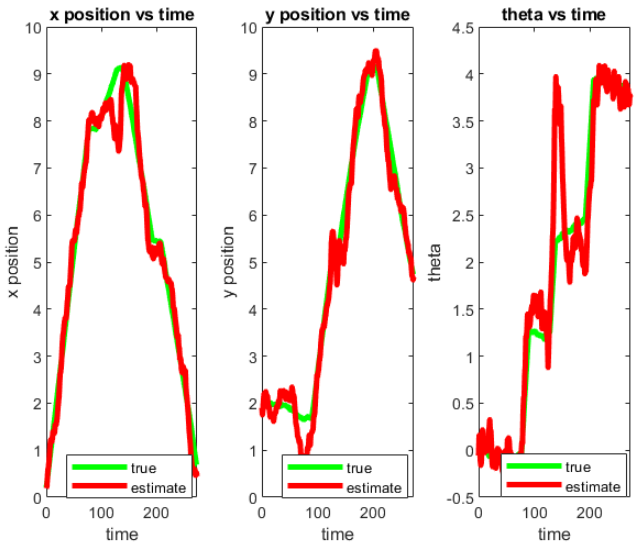

Figure 5: Localization with High R

Figure 2 illustrates robot state estimation vs truth to use as a baseline for comparing the effects of varying P, Q, and R. The estimation in this figure proves to be very accurate as P, Q, and R are reasonably low. Increasing initial position covariance P as in Figure 3 does not suggest much of a performance decrease. Figure 4 portrays a slight decrease in performance compared to the baseline estimation when increasing the process noise covariance, but nothing too drastic. Simulations with higher observation noise R did show drastic changes in estimation performance. In Figure 5, the x and y positions of the robot have a decent error, but generally oscillate about the true value. The orientation of the robot, θ, on the other hand suffers larger error. The oscillations about the true value are larger than seen in the x and y graphs and there is even a case of a very inaccurate estimate, seen at about 125s. Luckily, the Kalman filter quickly rights itself, but there are still sizeable errors for the rest of the duration. The above simulations involved all sensors except the lidar. Once lidar is introduced, performance becomes inconsistent. On some occasions, performance is significantly decreased, as illustrated in Figure 6. On other occasions, the estimator performance is decent as depicted in Figure 7.

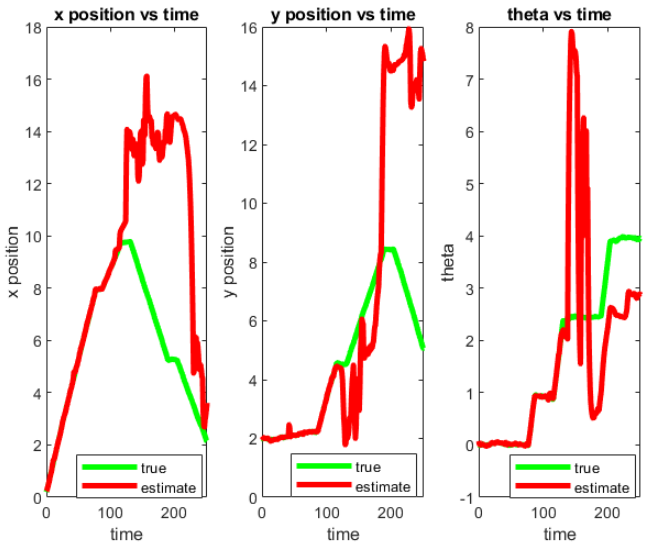

Figure 6: Poor Localization with Lidar

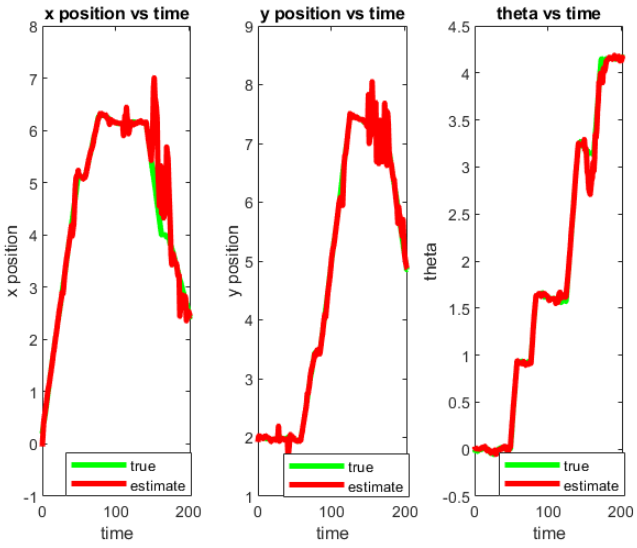

Figure 7: Decent Localization with Lidar

The two above figures were generated using 8 lidar scan angles. When the number of scan angles is increased, there is generally a slight improvement in the estimator performance, but not always. At first, this was assumed to be

an error with the observation matrix H associated with the lidar, but then Professor Wen shared that lidar alone is not sufficient for localization and needs to be combined with other computer vision areas such as feature extraction and registration. Thus, lidar is not used in the EKF for SLAM, and is instead only used to generate map points along with the state estimate from the EKF.

4.2 Simultaneous Location and Mapping

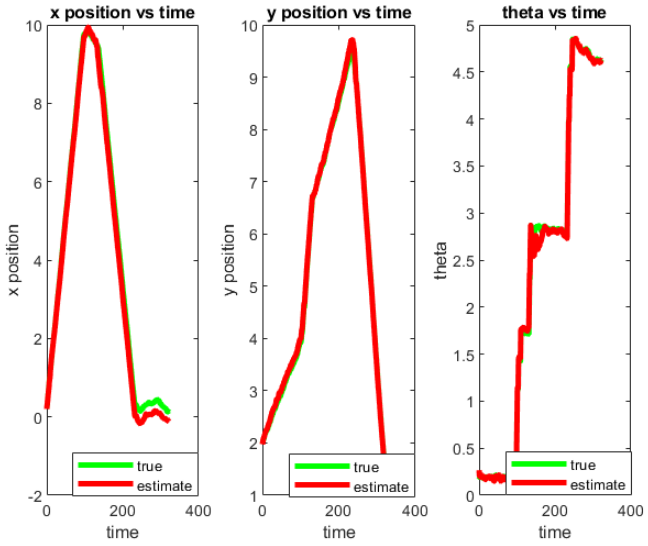

Figure 8: SLAM Robot Localization

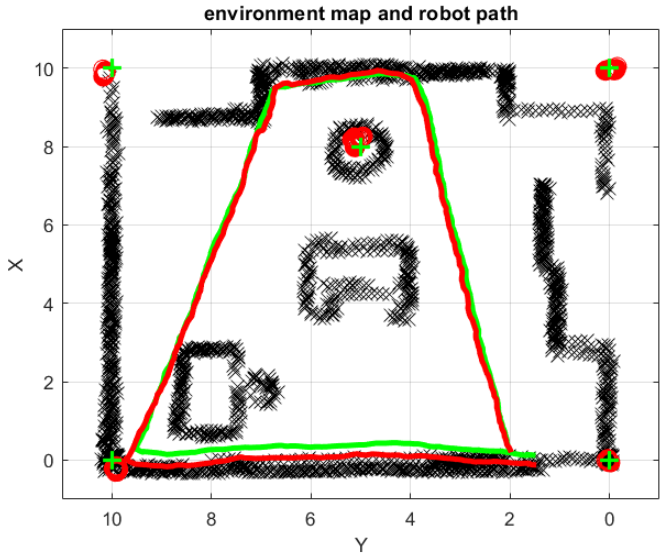

Figure 9: SLAM Mapping and Trajectory

Figures 8 and 9 will be used as a comparison baseline for varying P, Q, and R. In Figure 8, strong estimates for y and θ are shown, with the estimate of x being strong until around 250s, where it is slightly lower than the truth. Figure 9 shows an accurate map of object detections (black x's) as well as accurate estimates of the sensor positions (green +'s are the true position, red circles are the estimate positions over time).

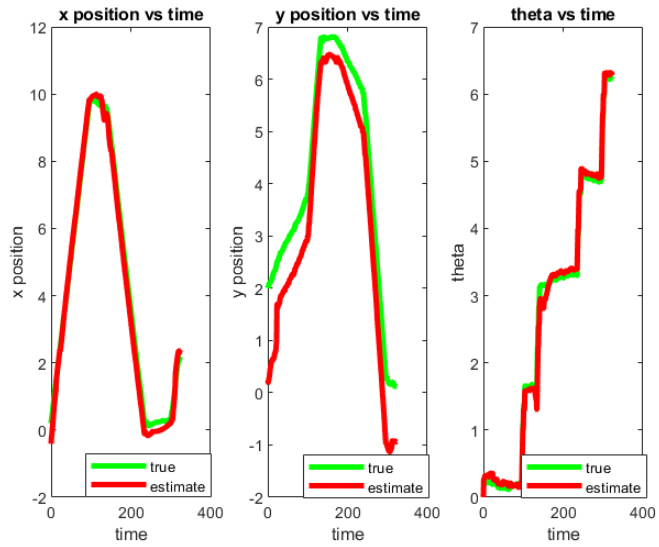

Figure 10: SLAM Robot Localization with High Initial P

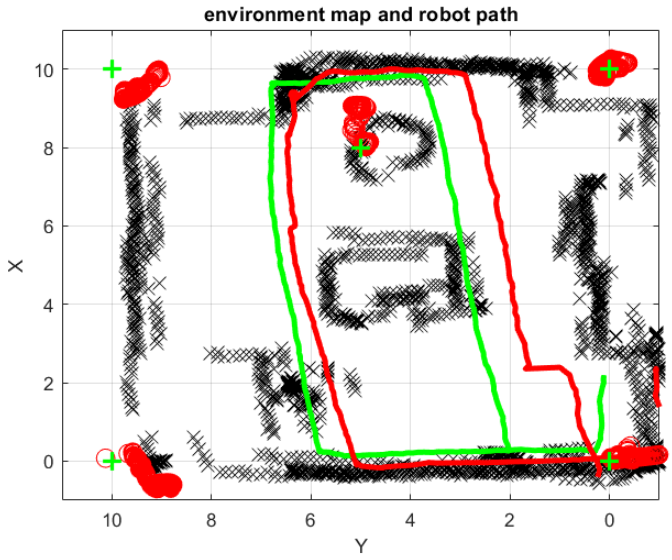

Figure 11: SLAM Mapping and Trajectory with High Initial P

The estimated states of the robot are poor in Figures 10 and 11 due to the poor sensor location estimates seen in Figure 11. Poor sensor location estimates results in higher measurement innovations. The lidar mapping of the room loses

accuracy because the map is calculated from the real lidar measurements and the estimated state of the robot, so an inaccurate robot state estimation results in a weaker map creation.

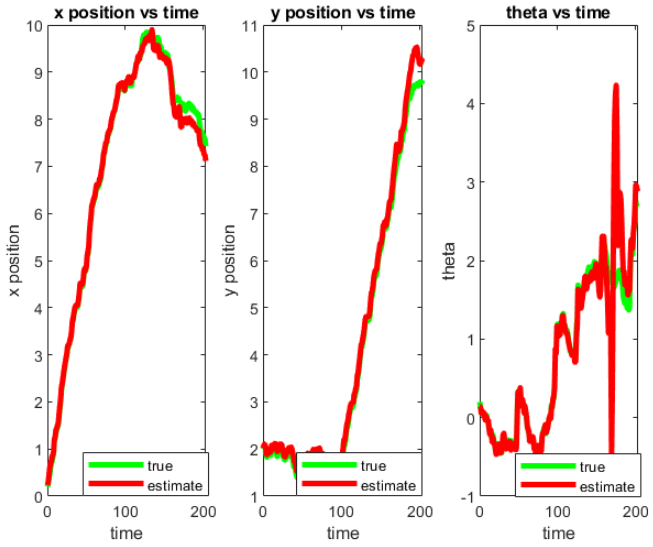

Figure 12: SLAM Robot Localization with High Q

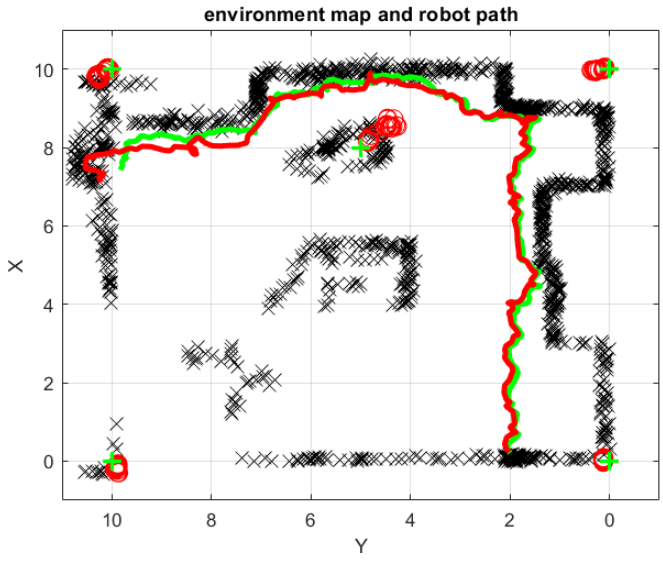

Figure 13: SLAM Mapping and Trajectory with High Q

Higher process noise covariance does not affect the state estimation or map generation too drastically, as portrayed in Figures 12 and 13. There are two instances of very erroneous θ estimates, but the EKF managed to reel in the error.

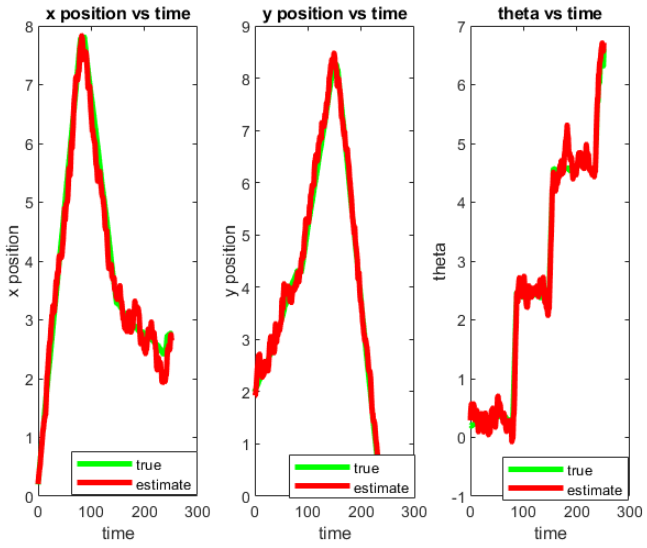

Figure 14: SLAM Robot Localization with High R

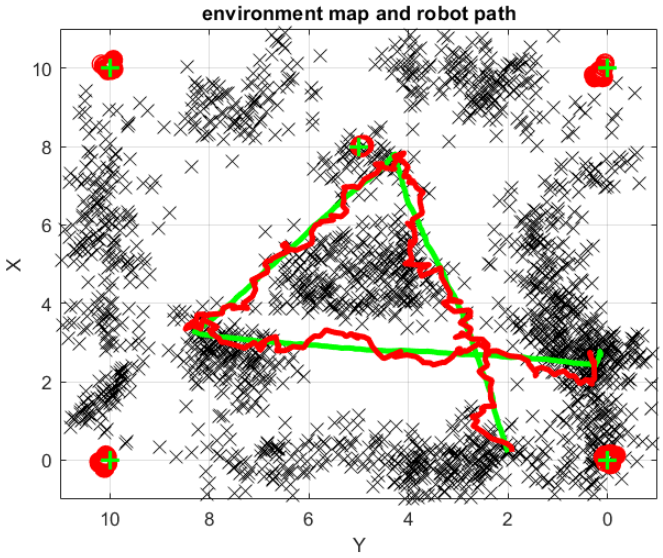

Figure 15: SLAM Mapping and Trajectory with High R

Higher observation noise covariances has a greater effect on the lidar mapping than it does the state estimation. The robot state estimates in Figure 14 are all reasonable despite some oscillation about the truth value. The map in Figure 15, however, suffered greatly from the increase in the lidar covariance. In Figures 11 and 13, the outlines of obstacles and walls are pretty clear and there is a lot of open space in between the obstacles while in Figure 15, it is very difficult to make out the outline of the obstacles and walls and the free space is much less abundant.

4.3 Path Planning

The simple path planning algorithm is a decent solution for testing without requiring manual keyboard entry, but is not productive for room coverage. The robot tends to ride edges and walls into corners where it gets stuck. The paths shown in Figures 9, 11, 13, and 15 were all generated using this algorithm.

5 Conclusions and Future Work

This project was a fantastic way to gain hands-on experience and practical knowledge with localization and mapping by use of a powerful tool, the extended Kalman filter. Not only did this project teach localization and mapping, but also it taught how to simulate a variety of different sensors. All of these skills are relevant in the real world and will certainly be applied upon graduation.
A huge area of improvement with this project is path planning, which will

hopefully be expanded upon in Mini Project 2. It would be very interesting to see if it is possible to devise an efficient path that will cover the room while also taking the sensors into account to minimize state estimation error along the way. Other areas that could be expanded upon are comparing different SLAM algorithms, such as the particle filter, in order to learn more about their pros and cons or comparing results for non-Gaussian distributions. Another expansion of interest would be incorporating the lidar measurements into state estimation using the techniques mentioned by Professor Wen. This would be a great practical example for applying knowledge on those new techniques.

6 References

`https://github.com/donelanp/localization_and_mapping.git`