HW 3 Report

Algorithm

The state *x*, which we labeled as *s* in our code so as not to be confused with the x-coordinate, is a vector that includes the estimated x and y coordinates and θ orientation (in radians) of our robot in the map frame as well as the estimated x and y coordinates of each of the landmarks in the map frame. It has the form $[x_r, y_r, \theta_r, x_1, y_1, x_2, y_2, \ldots, x_n, y_n]$ for all n landmarks included in the map frame. We don't make the assumption that we have a certain number of landmarks. Instead the state vector *x* starts with only x_r, y_r, θ and dynamically appends x_i, y_i as new landmarks are detected. The final size of the state vector is $3 + 2n$, where *n* is the total number of landmarks found at a specific time. The measurement vector *z* includes all detected AprilTag poses, new and old, at a specific time. It has the form $[x_i, y_i, x_{i+1}, y_{i+1}, \ldots, x_a, y_a]$ for all detected landmarks L_i to L_a. Therefore the size of *z* is $2a$ where *a* is the number of detected landmarks, new and old.

Since we are using the linear Kalman Filter, we set our system matrix *F* to the identity matrix, assuming perfect autonomous control. Essentially, assuming that there is no motor command *Gu* or noise *w*, we are expecting the state vector *x* to remain unchanged. The control matrix *G* was simply the identity matrix since we are given the twist as *u*, so *Gu* ended up being the change in x, change in y, and change in theta: $[v_x, v_y, w_z] * \Delta t = [\Delta x, \Delta y, \Delta\theta]$. Our measurement matrix *H* needed to take the state vector *x* and transform it into the expected prediction *Hx*, which should be close to the actual prediction *z*. We derived the following form for *H*:

$$\begin{bmatrix}
0 & 0 & 0 & \left(1-\frac{{}^Mx_r}{{}^Mx_i}\right)\cos(-{}^M\theta_r) & \left(1-\frac{{}^My_r}{{}^My_i}\right)\sin(-{}^M\theta_r) & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & \left(1-\frac{{}^Mx_r}{{}^Mx_i}\right)\sin(-{}^M\theta_r) & \left(1-\frac{{}^My_r}{{}^My_i}\right)\cos(-{}^M\theta_r) & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & \left(1-\frac{{}^Mx_r}{{}^Mx_{i+1}}\right)\cos(-{}^M\theta_r) & \left(1-\frac{{}^My_r}{{}^My_{i+1}}\right)\sin(-{}^M\theta_r) & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & \left(1-\frac{{}^Mx_r}{{}^Mx_{i+1}}\right)\sin(-{}^M\theta_r) & \left(1-\frac{{}^My_r}{{}^My_{i+1}}\right)\cos(-{}^M\theta_r) & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & \left(1-\frac{{}^Mx_r}{{}^Mx_{i+2}}\right)\cos(-{}^M\theta_r) & \left(1-\frac{{}^My_r}{{}^My_{i+2}}\right)\sin(-{}^M\theta_r) \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & \left(1-\frac{{}^Mx_r}{{}^Mx_{i+2}}\right)\sin(-{}^M\theta_r) & \left(1-\frac{{}^My_r}{{}^My_{i+2}}\right)\cos(-{}^M\theta_r)
\end{bmatrix}$$

Assume that this pattern continues for all detected landmarks L_i to L_a, giving us a $2a$ x $3 + 2n$ matrix. These values were chosen assuming the differences $({}^Mx_i - {}^Mx_r)$ and $({}^My_i - {}^My_r)$ would be calculated and then multiplied to the rotation matrix elements once we get the dot product of the state vector *x* and *H*.

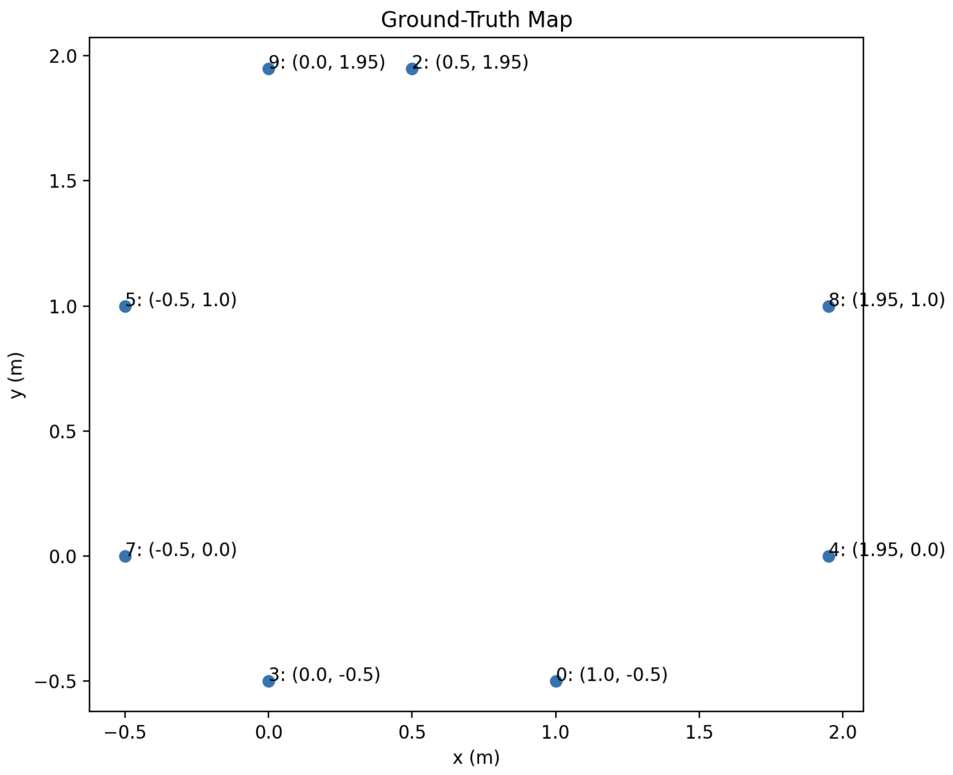
We initialized the state $x_{0|0}$ to (0, 0, 0) since the robot start pose would be considered our origin. We initialized the state covariance $\Sigma_{0|0}$ to a matrix with the diagonal elements set to a high value (100) and the off-diagonal elements set to a low value (1E-6). We selected this because starting at a large uncertainty allows for the Kalman Filter to use the AprilTag measurements to reduce the uncertainties in Σ to more accurate values. Meanwhile we expect very little correlation between the landmarks and the robot pose.

The system noise Q was initialized similarly to the state covariance matrix, but the diagonal elements were set to 0.4, except for the θ uncertainty which we set to 0.05. We thought 0.4 m was a reasonable estimate to account for the uncertainty in our robot pose estimation based on our observations. For the measurement noise, we measured pose variance by taking 10 measurements of the x and y coordinates and using the maximum for the uncertainty in our AprilTag pose estimates, which was 4E-6. We expanded Q and Σ whenever a new landmark was detected, using the same initialization uncertainty.

When a new landmark was detected, as said before, we expanded the covariance matrices Q and Σ. We also added the new landmark to the state vector, converting the poses from the camera frame to the robot frame and then to the map frame. Whenever any landmark was detected (old or new), R, z, and H were recreated, since R had to include the measurement uncertainty for all detected landmarks and z included the poses of all detected landmarks. The state x and covariance Σ were also updated. When the landmark went out of the field of view, it was no longer part of the detected landmarks. Therefore it was not included in z and not used for the update of x and Σ. When a previously detected landmark reappeared, it was again included in z and used for the update of x and Σ.

Ground-Truth Map

We chose an area of 2.45 m x 2.45 m. Note that all of the AprilTags are facing inwards of the environment, with the robot inside, at orientations 0, π/2, π, and -π/2. To make the transformation from robot to map frame easier, we set the origin to the start pose of the robot.

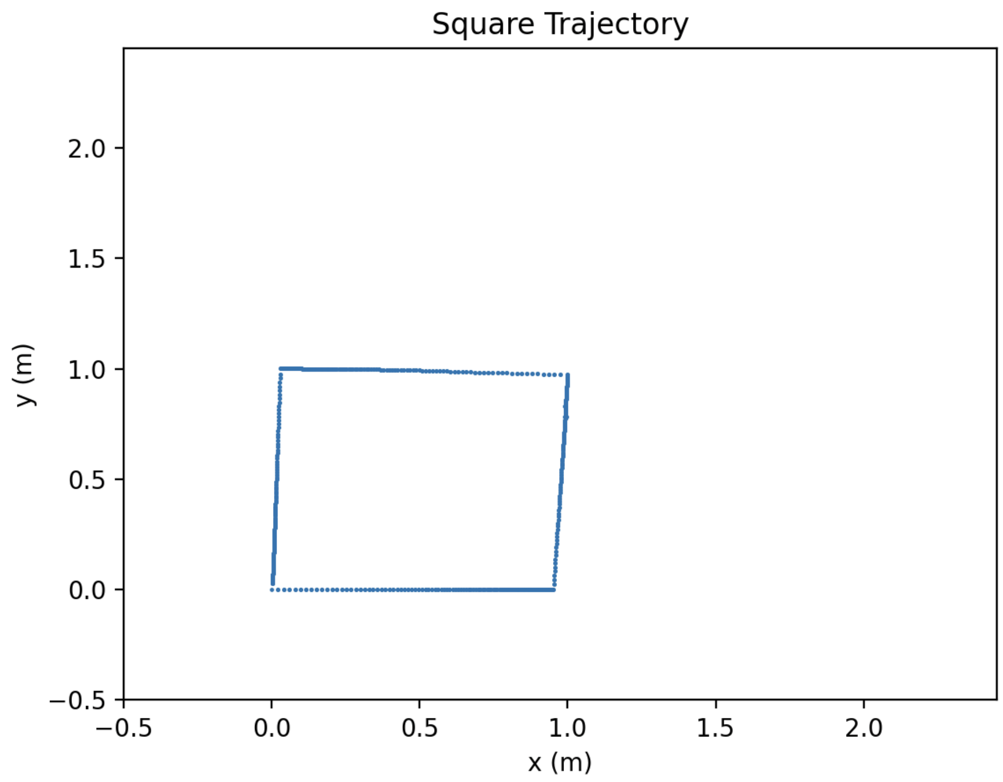


Results with Square

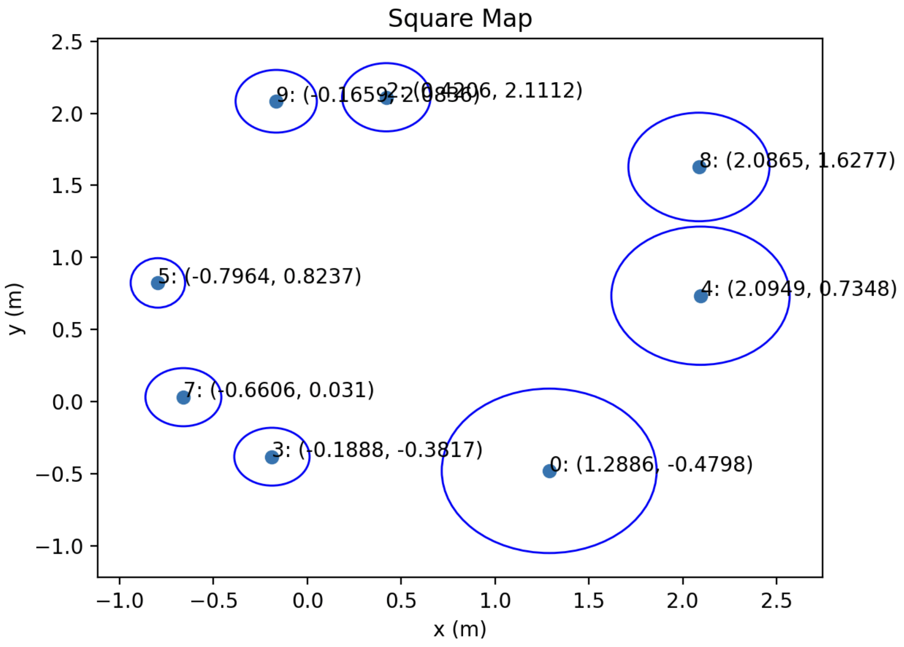
We selected the size of the square (1.00 m) based on how accurately the robot could adhere to the given path. We specifically chose the size based on how accurate the orientation of the robot was once it reached the waypoint. Although we understood that the accuracy of the measurements would possibly be greater if the size of the path was larger, we decided to pick the size that would maximize the accuracy of the trajectory.

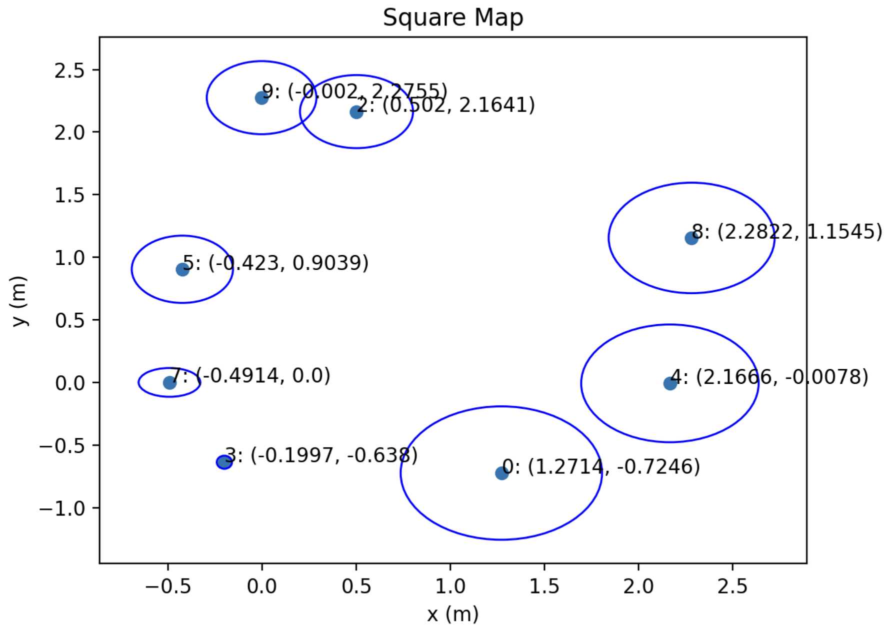
The average error of landmarks between the generated map and the ground-truth map is: 1.2495. If we run multiple times, the map does improve, giving more accurate results (1.1310 error). The bottom map is the second-run map.

Trajectory Plot:



Landmark Map:



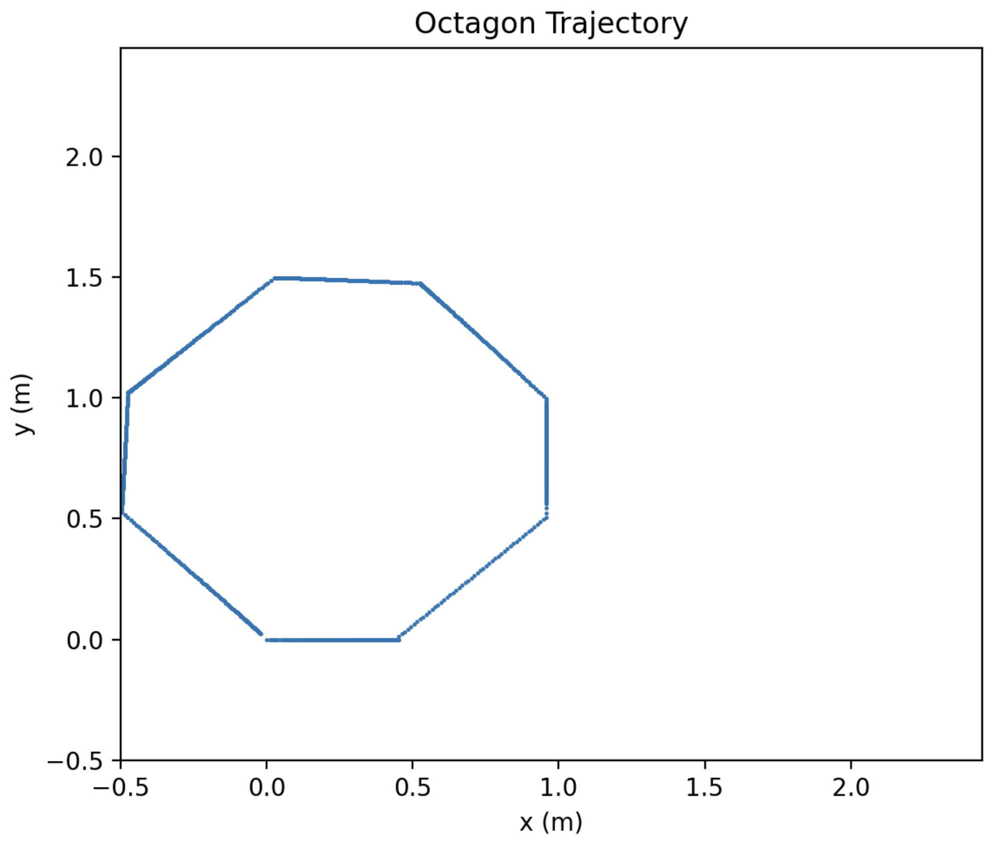


Results with Octagon

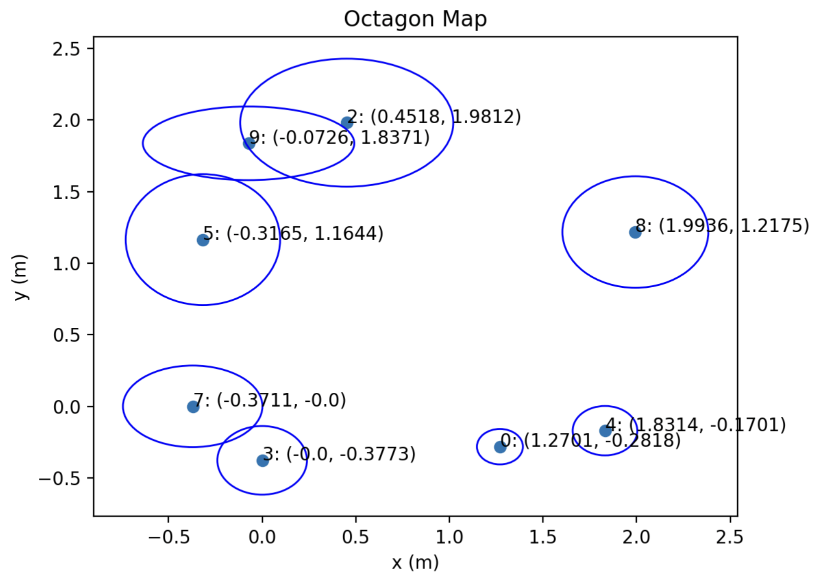
We selected the size of the octagon (0.5 m) based on how accurately the robot could adhere to the given path. We specifically chose the size based on how accurate the orientation of the robot was once it reached the waypoint. Although we understood that the accuracy of the measurements would possibly be greater if the size of the path was larger, we decided to pick the size that would maximize the accuracy of the trajectory.

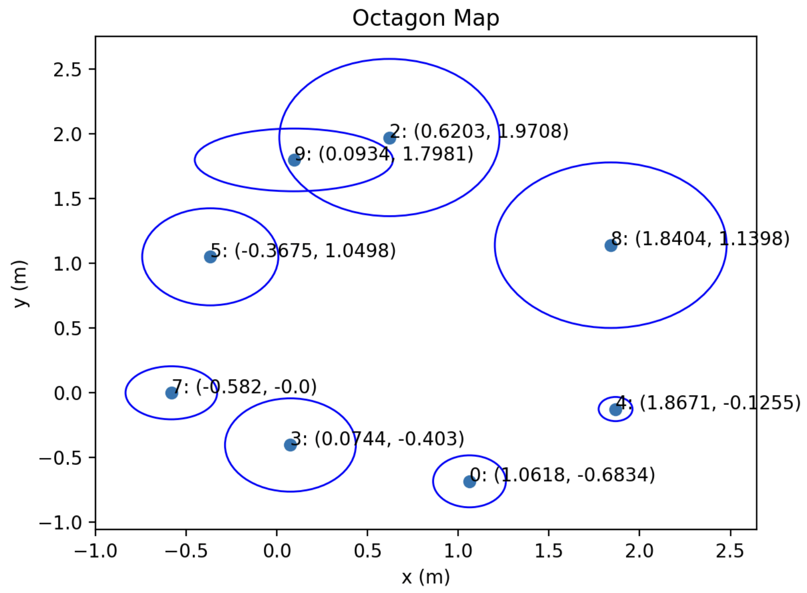
The average error of landmarks between the generated map and the ground-truth map is: 1.0322. If we run multiple times, the map does improve, giving more accurate results (0.9906 error). The bottom map is the second-run map.

Trajectory Plot:



Landmark Map:





We saw a bit more variance in the octagon map, but it was overall more accurate at finding the landmark positions, perhaps because it covered more ground area and got closer to each tag, getting more estimates of the location of every tag than the square, which had to face one direction and picked up a maximum of 4 different tags.

YouTube Links:
Square: https://youtu.be/4ykrH3w9WKo
Octagon: https://youtu.be/7lrsTF75ZSA
*Our functions are in rb5_control/kalman_filter.py but we launched the hw1_solution.py code.