Introduction

In this assignment we were tasked to implement a constrained version of the EKF-Slam algorithm. The first section answers some basic math and theory questions core to the algorithm. The next section goes over the implementation of the algorithm and initial results. The final section discusses what the results mean and the practical effects of various components of the algorithm.

Collaborators: Andrew Wong (awong2), David Robinson (davidr1)

Math and Equations

A. $\boldsymbol{p}_{t+1} = \begin{bmatrix} x_{t+1} \\ y_{t+1} \\ \theta_{t+1} \end{bmatrix} = \begin{bmatrix} x_t \\ y_t \\ \theta_t \end{bmatrix} + \begin{bmatrix} d_t \cos(\theta_t) \\ d_t \sin(\theta_t) \\ \alpha_t \end{bmatrix}$

B. To get the predicted uncertainty of the robot at time t+1 we must first take the partial derivative of the above equation with respect to each of the state variables. This will get us a Jacobian which we will call G_{t+1}

$$G_{t+1} = \begin{bmatrix} \frac{\delta x}{\delta_{\mu_t x}} & \frac{\delta x}{\delta_{\mu_t y}} & \frac{\delta x}{\delta_{\mu_t \theta}} \\ \frac{\delta y}{\delta_{\mu_t x}} & \frac{\delta y}{\delta_{\mu_t y}} & \frac{\delta y}{\delta_{\mu_t \theta}} \\ \frac{\delta \theta}{\delta_{\mu_t x}} & \frac{\delta \theta}{\delta_{\mu_t y}} & \frac{\delta \theta}{\delta_{\mu_t \theta}} \end{bmatrix} = \begin{bmatrix} 1 & 0 & -d_t \sin(\theta_t) \\ 0 & 1 & d_t \cos(\theta_t) \\ 0 & 0 & 1 \end{bmatrix}$$

We also need the Jacobian of the motion model with respect to the motion parameters, V_t.

$$V_{t+1} = \begin{bmatrix} \cos(\theta_t) & 0 & 0 \\ \sin(\theta_t) & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Now that we have our two Jacobians, we can calculate the predicted state via the following equation.

$$\bar{\Sigma}_{t+1} = G_{t+1} \bar{\Sigma}_t G_{t+1}^T + V_{t+1} M_t V_{t+1}^T$$

$$where\ M_t\ is\ the\ control\ covariance = \begin{bmatrix} \sigma_x^2 & 0 & 0 \\ 0 & \sigma_y^2 & 0 \\ 0 & 0 & \sigma_\alpha^2 \end{bmatrix}$$

C. The estimated position of a landmark is given by:

$$\begin{bmatrix} l_x \\ l_y \end{bmatrix} = \begin{bmatrix} p_x + (r + n_r) \cos(\beta + n_\beta + p_\theta) \\ p_y + (r + n_r) \sin(\beta + n_\beta + p_\theta) \end{bmatrix}$$

D. The predicted measurement and bearing range is given by:

$$r = \sqrt{(l_x - p_x)^2 + (l_y - p_y)^2}$$

$$\beta = \tan^{-1}\left(\frac{l_y - p_y}{l_x - p_x}\right) - p_\theta$$

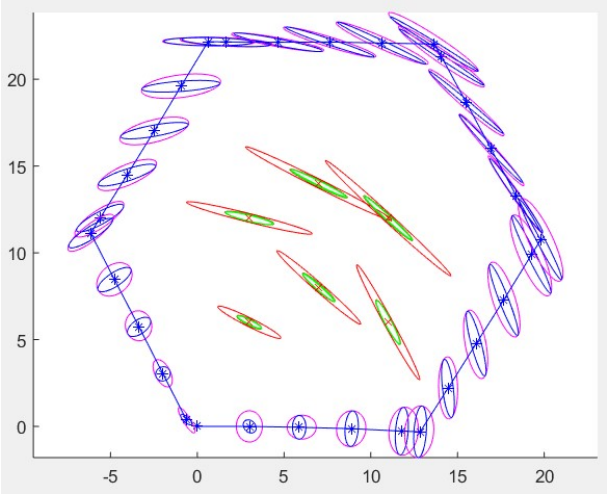
E. $H_p = \begin{bmatrix} -\left(\frac{l_x - p_x}{r}\right) & -\left(\frac{l_y - p_y}{r}\right) & 0 \\ \left(\frac{l_y - p_y}{r^2}\right) & -\left(\frac{l_x - p_x}{r^2}\right) & -1 \end{bmatrix}$

F. $H_l = \begin{bmatrix} \frac{(l_x - p_x)}{r} & \frac{l_y - p_y}{r} \\ -\left(\frac{l_y - p_y}{r^2}\right) & \frac{l_x - p_x}{r^2} \end{bmatrix}$

We do not need to calculate the measurement Jacobian with respect to other landmarks because we are assuming that the landmarks are independent (i.e. one landmark does not tell us anything about the other landmarks.)

Implementation and Evaluation

A. There are 6 landmarks being observed.
B. $\sigma_x = 0.25, \sigma_y = 0.1, \sigma_\alpha = 0.1, \sigma_\beta = 0.01, \sigma_r = 0.08$



Ground truth represented by red x's

C. EKF-Slam relies on both a control input to move the robot and measurement of landmarks to estimate where the robots position is. To start a prediction is made about where the robot is using a control input. Because we don't live in a noiseless, errorless world we can't be 100% confident that that control input is actually what happened, so we use a 2D gaussian to represent our estimated position with the mean being the exact prediction from the control input.
   Now we want to compensate for this uncertainty which we can do using landmark measurements. First, we make a prediction about where the landmarks should be based on our previous measurements and the robots predicted new pose. Then we read in what the measurements actually were and compare the actual measurements to our predicted measurements. Again, because we don't live in a perfect world, we end up with an estimated area of possibilities, represented again by an uncertainty ellipse.
   The last step is to compare the uncertainty ellipse of the predicted state from odometry and predicted state from the measurement and decide where the robot might actually be, usually somewhere in between. How much we actually trust all of these measurements is a design choice made at the beginning based on prior knowledge of sensor accuracy.
D. Each red x (ground truth) is roughly centered inside it's corresponding ellipse meaning that the prediction uncertainty has a mean near the ground truth (ie our prediction is very accurate.)

   Euclidean distances: 0.0086 0.0200 0.0261 0.0187 0.0125 0.0302

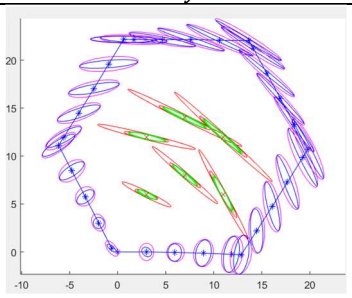
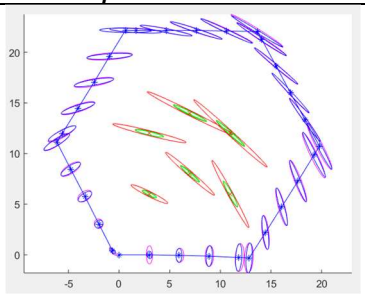
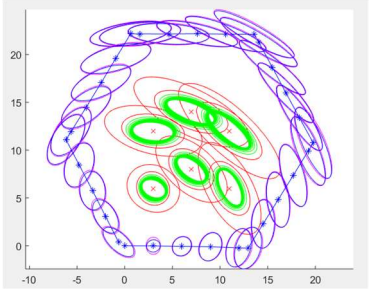
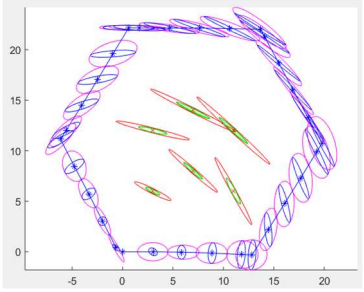
Mahalanobis distances: 0.0014 0.0081 0.0078 0.0081 0.0059 0.0182

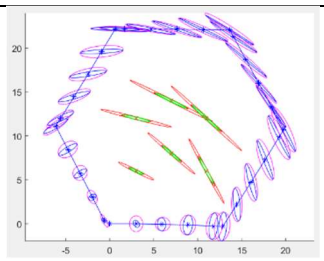
The Euclidean shows how close the prediction physically is to the ground truth. The Mahalanobis distance shows us effectively how many standard deviations away from the ground truth the prediction is.

Discussion

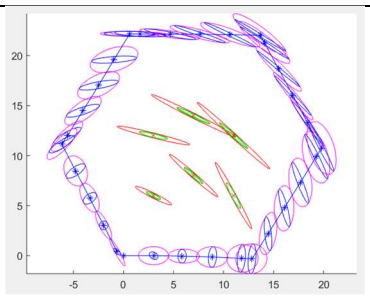
A. The landmark covariance matrix is initialized with all 0's off the diagonal. This encodes the initial assumption that all of the landmarks are independent. As the robot moves along it's loop and the Kalman filter continues updating, the algorithm is effectively learning that the independence assumption is not true, and that one landmark measurement does in fact give you information about the other measurements. Essentially, the algorithm learns that the landmarks are static and unchanging.
When building the full covariance matrix, it is assumed that the robot's location does not tell us anything about the landmarks location and vice versa.

B. When we fix the initial pose covariances and adjust the measurement covariances, somewhat predictably, the main thing that changes is our estimate uncertainty of the landmark locations. When we increase the covariances, our uncertainty ellipses grow as we are inherently stating that we are not confident in the measurements being read in. This effect propagates to our pose estimate as we can't as confidently use the landmark locations for reference, so we aren't as confident in the robot's position. Decreasing the values of the measurement covariances has the opposite effect. However, if we drop them too low, the algorithm becomes overconfident and starts making predictions that aren't correct while believing that they are.
Conversely, fixing the measurement covariances and adjusting pose covariances mainly effects the robots pose estimates. Similar to above, increasing the covariances increases uncertainty so the ellipses get much bigger. Decrease the covariances and the uncertainty ellipses get much smaller.

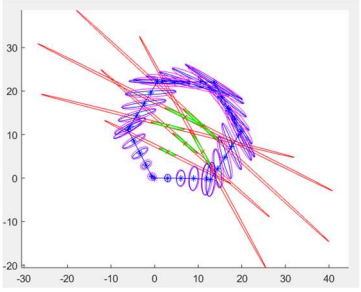
Fixed: $\sigma_x = 0.25, \sigma_y = 0.1, \sigma_\alpha = 0.1$	**Fixed: $\sigma_\beta = 0.01, \sigma_r = 0.08$**
 $\sigma_\beta = 0.1, \sigma_r = 0.1$	 $\sigma_x = 0.1, \sigma_y = 0.1, \sigma_\alpha = 0.1$
 $\sigma_\beta = 0.1, \sigma_r = 0.8$	 $\sigma_x = 0.5, \sigma_y = 0.2, \sigma_\alpha = 0.2$



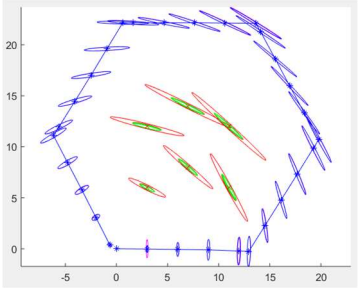
$\sigma_\beta = 0.05, \sigma_r = 0.05$



$\sigma_x = 0.5, \sigma_y = 0.5, \sigma_\alpha = 0.5$



$\sigma_\beta = 0.8, \sigma_r = 0.1$



$\sigma_x = 0.025, \sigma_y = 0.01, \sigma_\alpha = 0.01$