Assignment 2

‘Lost in the Woods Dataset’

In this second assignment we’ll investigate a nonlinear two-dimensional problem consisting of a robot driving amongst a forest of tubes. We wish to estimate the position/orientation of the robot throughout its $\approx$ 360 m traverse. We’ll use the recursive extended Kalman filter to fuse speed measurements coming from wheel odometry with range/bearing measurements coming from a laser rangefinder.

2.1 Experimental Setup

The setup consists of a mobile robot driving amongst a forest of plastic tubes as depicted in Figure 2.1. The robot and the tubes were equipped with reflective markers and tracked using a ten-camera motion capture system. This

(a) Mobile robot driving amongst a forest of tubes. Robot has an odometer to measure translational/rotational speeds and a laser rangefinder to measure range/bearing to the cylindrical landmarks. Both sensors are noisy.

(b) Vicon motion capture lab. Ten cameras work together to track markers on the robot and provide groundtruth position/orientation.

Figure 2.1: Setup for Dataset 2.

motion capture system is able to provide the position of each reflective marker to within a few millimeters. Position estimates from this motion capture system are considered to be quite a bit more accurate than estimates based on the robot's onboard sensors and thus it serves as the benchmark/groundtruth in our experiment.

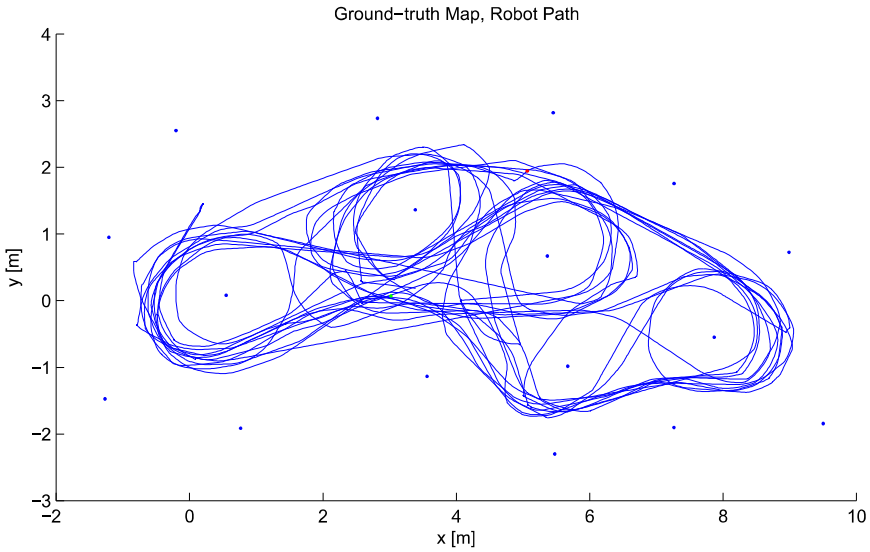


Figure 2.2: Robot path and 17 landmarks for Dataset 2.

There were 17 reflective markers placed on the tubular landmarks. The positions of these were measured using the motion capture system for 20 minutes and then averaged. The output of these steps can be seen in Figure 2.2.

The mobile robot was driven around for 20 minutes and three streams of data were logged:

- laser rangefinder scans consisting of 681 range measurements spread over a 240° horizontal field of view centered on straight ahead (Hokuyo URG-04LX sensor), logged at 10 Hz
- robot's speed based on a wheel odometry, logged at 10 Hz
- ground position of a marker on the laser rangefinder origin, logged at approximately 40 Hz

Figure 2.2 shows the robot's path over the 20 minute trial. We see that the robot's path is very twisted and looping. We must estimate the robot's position and orientation as it moves in this two-dimensional environment.

As with most real datasets, our data streams do not arrive synchronously or with evenly-spaced timesteps. Figure 2.3 shows the variability in the sampling periods of the nominally 10 Hz laser scans. For our purposes, we will assume that the data was acquired with a uniform 0.1 second sampling period. To avoid having to deal with the speed measurements and groundtruth data arriving asynchronously with the laser rangefinder scans, these measurements were linearly interpolated at the laser rangefinder timestamps. Thus, the data to be used in this assignment is synchronous with a uniform sampling period. Moreover, to avoid having to process the raw laser rangefinder scans, the range/bearing to each plastic tube was determined using groundtruth data. Figure 2.4 shows how a range measurement was fit to a laser scan.

Another important issue worth mentioning is the noise on the sensor readings. Because we have a very accurate motion capture system, we may use the groundtruth positions of everything to determine the true range/bearing

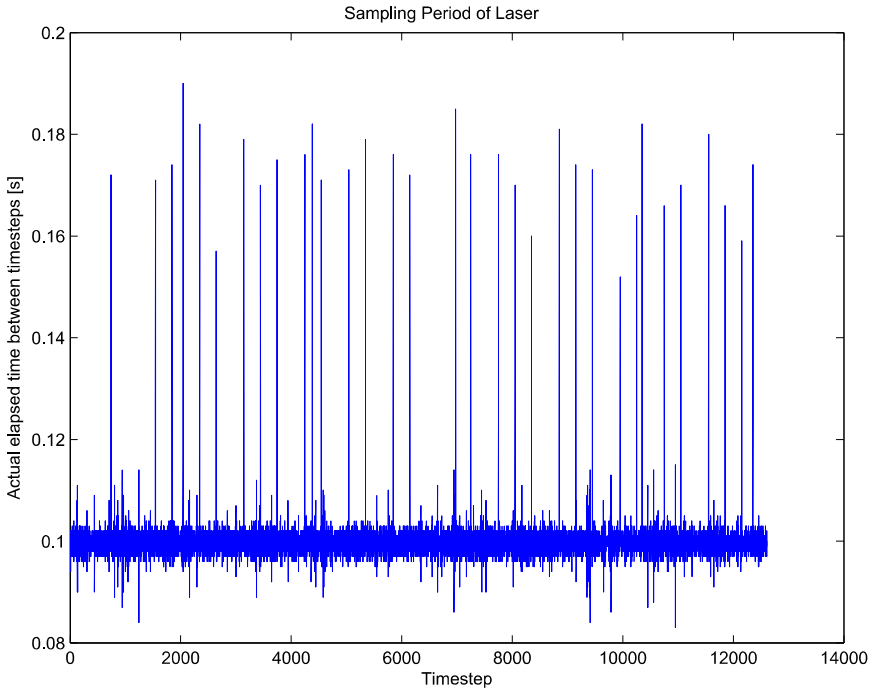


Figure 2.3: Sampling periods of the 10 Hz laser rangefinder scans. We see that the period does hover around 0.1 seconds.

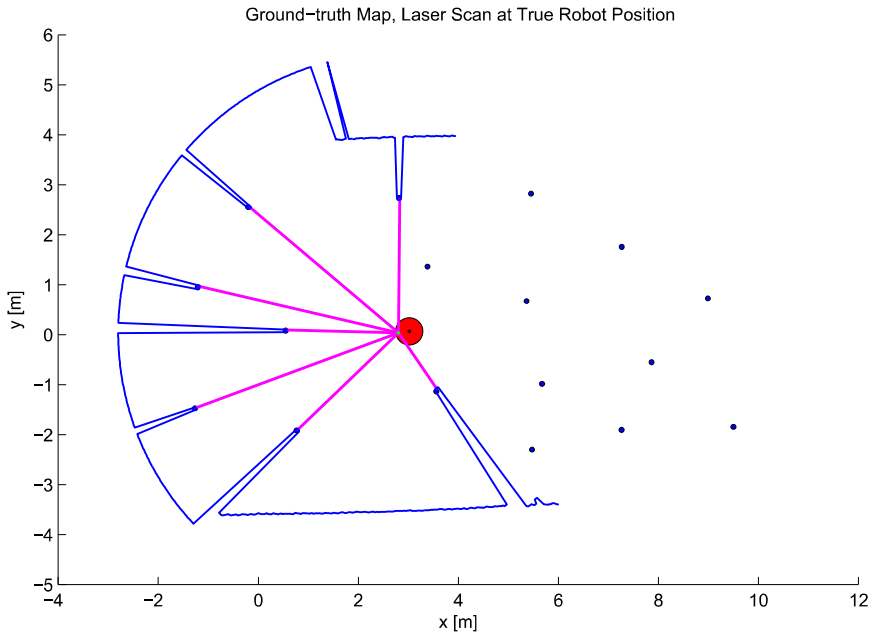


Figure 2.4: The purple line shows a range measurement from the robot's laser scanner to the cylinder's closest edge. The cylinder's radius is added to this to get the actual range to the cylinder.

readings and the true robot translational/rotational speeds at each timestep. Figure 2.5 shows histograms of the errors in the range/bearing for the 20 minute dataset. Figure 2.6 shows histograms of the errors in the translational/rotational speeds for the 20 minute dataset.

The file containing the final processed data for this assignment is called `dataset2.mat`, which is a Matlab binary file. You can view the contents by typing

```
load dataset2.mat
who
```

at the Matlab (or Octave) command prompt. The following 15 Matlab variables will be listed:

`t` : a 12609×1 array containing the data timestamps [s]

`x_true` : a 12609×1 array containing the true x-position, x_k, of the robot [m]

`y_true` : a 12609×1 array containing the true y-position, y_k, of the robot [m]

`th_true` : a 12609×1 array containing the true heading, θ_k, of the robot [m]

`true_valid` : a 12609×1 array containing a flag indicating if the groundtruth data is 'valid' or not (a value of 1 means it is valid and a value of 0 means it was interpolated due to some data dropouts)

`l` : a 17×2 array containing the true xy-positions, (x_l, y_l), of the 17 landmarks [m]

`r` : a 12609×17 array containing the range, r_k^l, between the robot's laser rangefinder and each landmark's center as measured by the laser rangefinder sensor [m] (a range of 0 indicates the landmark was not visible at that timestep)

`r_var` : the variance of the range readings (based on groundtruth) [m^2]

`b` : a 12609×17 array containing the bearing, ϕ_k^l, to each landmark's center in a frame attached to the laser rangefinder, as measured by the laser rangefinder sensor [rad] (if the range is 0 it indicates the landmark was not visible at that timestep and thus the bearing is meaningless)

`b_var` : the variance of the range readings (based on groundtruth) [rad^2]

`v` : a 12609×1 array containing the translational speed, v_k, of the robot as measured by the robot's odometers [m/s]

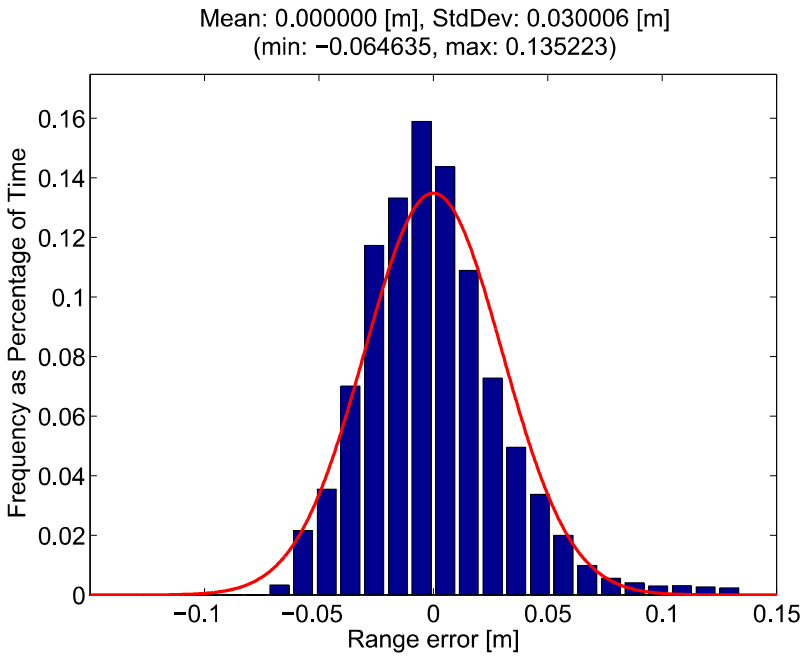
`v_var` : the variance of the translational speed readings (based on groundtruth) [m^2/s^2]

`om` : a 12609×1 array containing the rotational speed, ω_k, of the robot as measured by the robot's odometers [rad/s]

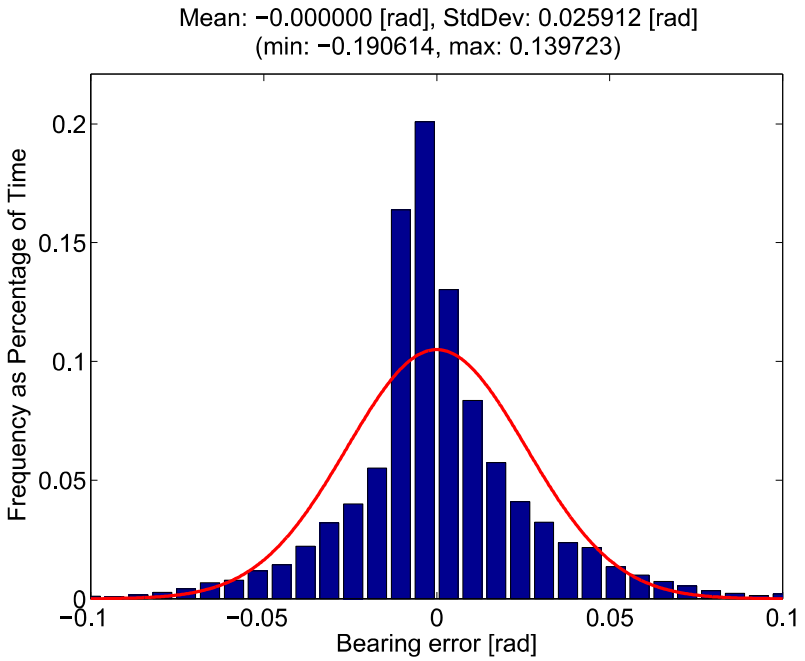
`om_var` : the variance of the rotational speed readings (based on groundtruth) [rad^2/s^2]

`d` : the distance, d, between the center of the robot and the laser rangefinder [m]

The main idea is to estimate the robot's position without using `x_true`, `y_true`, and `th_true` and then use these as a benchmark to gauge performance.

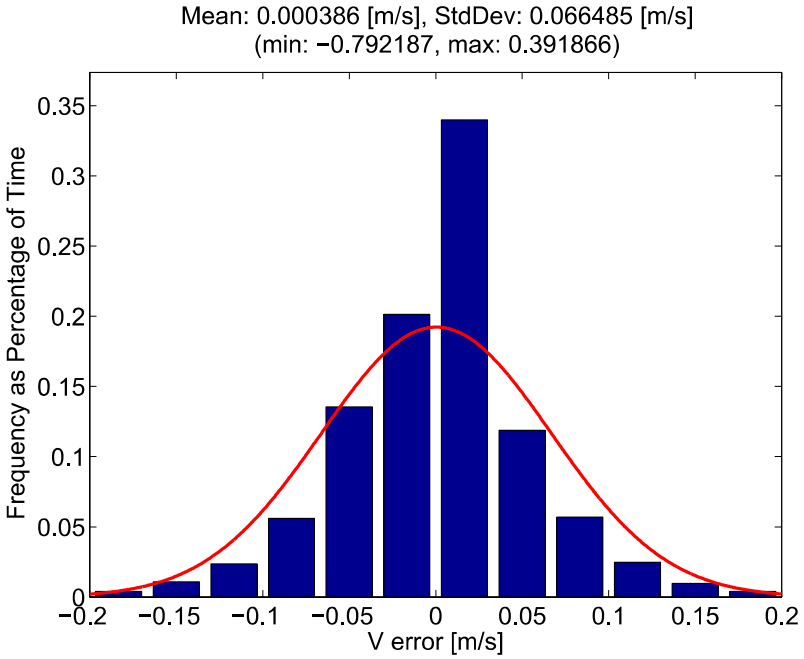


(a) Range errors.

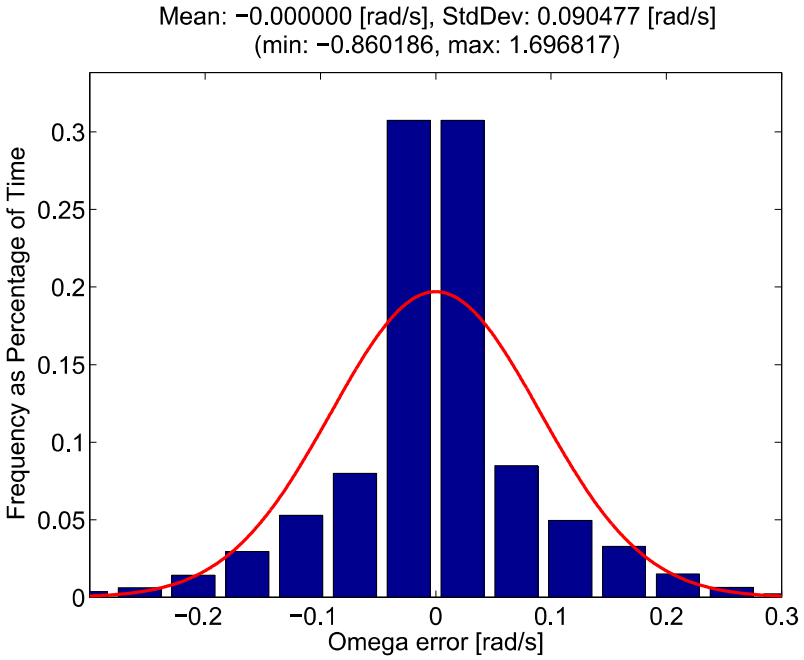


(b) Bearing errors.

Figure 2.5: Histograms of sensor errors. Red curves are Gaussians with standard deviation fit to the data. Both range and bearing have close to zero-mean errors with a spread.



(a) Translational speed errors.



(b) Rotational speed errors.

Figure 2.6: Histograms of sensor errors. Red curves are Gaussians with standard deviation fit to the data. Both translational and rotational speeds have close to zero-mean errors with a spread.

2.2 Motion and Observation Models

For this nonlinear estimation problem, use the following vehicle and sensor models:

$$\text{motion:} \quad \underbrace{\begin{bmatrix} x_k \\ y_k \\ \theta_k \end{bmatrix}}_{\mathbf{x}_k} = \underbrace{\begin{bmatrix} x_{k-1} \\ y_{k-1} \\ \theta_{k-1} \end{bmatrix}}_{\mathbf{x}_{k-1}} + T \begin{bmatrix} \cos\theta_{k-1} & 0 \\ \sin\theta_{k-1} & 0 \\ 0 & 1 \end{bmatrix} \left(\underbrace{\begin{bmatrix} v_k \\ \omega_k \end{bmatrix}}_{\mathbf{u}_k} + \mathbf{w}_k \right) \tag{2.1a}$$

$$\mathbf{h}(\mathbf{x}_{k-1}, \mathbf{u}_k, \mathbf{w}_k)$$

$$\text{observation:} \quad \underbrace{\begin{bmatrix} r_k^l \\ \phi_k^l \end{bmatrix}}_{\mathbf{y}_k^l} = \underbrace{\begin{bmatrix} \sqrt{(x_l - x_k - d\cos\theta_k)^2 + (y_l - y_k - d\sin\theta_k)^2} \\ \text{atan2}(y_l - y_k - d\sin\theta_k, x_l - x_k - d\cos\theta_k) - \theta_k \end{bmatrix} + \mathbf{n}_k^l}_{\mathbf{g}(\mathbf{x}_k, \mathbf{n}_k^l)} \tag{2.1b}$$

where (x_k, y_k, θ_k) is the robot's position/orientation, (v_k, ω_k) is the robot's translational/rotational speed (derived from odometers), $\mathbf{w}_k$ is the process noise, T is the sampling period, (r_k^l, ϕ_k^l) is the range/bearing to landmark l (derived from the laser rangefinder), (x_l, y_l) is the position of the center of landmark l, and $\mathbf{n}_k^l$ is the exteroceptive sensor noise. The distance between the robot center and the laser rangefinder is d (the laser is at the front of the robot). Note that at each timestep, k, there are multiple exteroceptive measurements (e.g., in Figure 2.4 there are 7 purple lines). Both the motion and observation models are nonlinear.

2.3 Assignment

For the assignment we'll use the extended Kalman filter to estimate the robot's position using both the odometry and laser measurements. You might wonder, why do we need to use both? In this example we could just use the wheel odometry to estimate the robot's position. Figure 2.7 shows the robot's path as estimated using only wheel odometry. It is very different that the true path in Figure 2.2; in general the difference between the two paths will grow without bound. Although there are a lot of laser-based range/bearing measurements, there are some timesteps when there are less than two such landmarks measurements. Figure 2.8 shows a histogram of the number of landmarks seen simultaneously. In fact, there are 436 out of 12609 timesteps when less than 2 landmarks are observed. This means that it is impossible to solve for the robot's position/orientation from the laser measurements alone for some timesteps. Thus, in this situation, we actually need both the odometry and laser measurements to create a reasonable estimator.

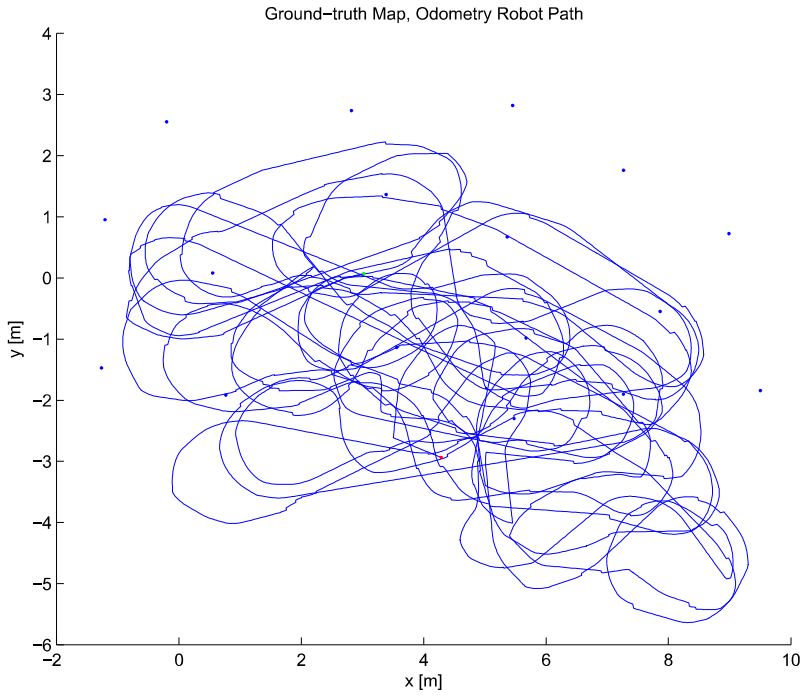


Figure 2.7: Robot path for dataset2 as computed using only wheel odometry (no laser rangefinder measurements).

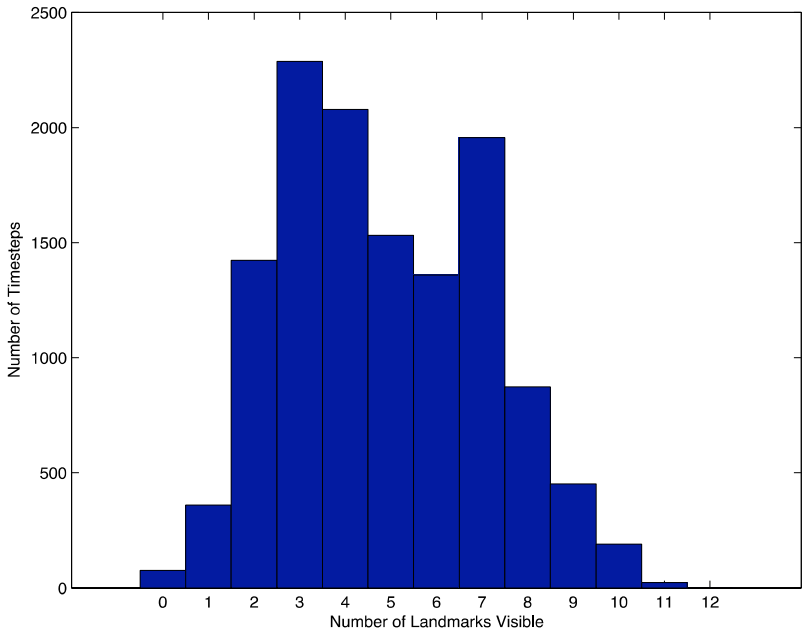


Figure 2.8: Histogram of number of landmarks seen simultaneously.

Answer the following questions in a short typeset report (marks in the margin - 25 total):

3 1. Based on the data above, is the assumption of zero-mean Gaussian noise reasonable? What values of the variances, $\mathbf{Q}_k$ and $\mathbf{R}_k^l$, should we use?

$$\mathbf{w}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_k), \qquad \mathbf{n}_k^l \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_k^l) \tag{2.2}$$

4 2. Write out expressions for all the Jacobians (of the motion and observation models) that are required in the extended Kalman filter.

4 3. Modify and write out the equations of the extended Kalman filter, modified to handle a variable number of exteroceptive measurements at each timestep.

10 4. Write a Matlab or Python script to compute an estimate the robot poses, $\hat{\mathbf{x}}_k = (\hat{x}_k, \hat{y}_k, \hat{\theta}_k)$, using the extended Kalman filter. Your code should be able to artificially limit itself to only use range/bearing landmark observations for which the range is less than a user-defined threshold, $r_{\max}$ [m]. Note, the true robot position, $\mathbf{x}_k = (x_k, y_k, \theta_k)$, is contained in the `x_true`, `y_true`, and `th_true` arrays. Make the following figures and write some short comments about your findings.

   (a) Use $\hat{\mathbf{x}}_0 = \mathbf{x}_0, \hat{\mathbf{P}}_0 = \text{diag}\{1, 1, 0.1\}$ as the initial condition (i.e., the estimator mean starts with the right answer). Plot the following quantities on the same axes:
   - the error, $\hat{x}_k - x_k$ vs. t_k, as a solid line.
   - the uncertainty envelope, $\pm 3\sigma_{x_k}$ vs. t_k, as two dotted lines. Note, the variance, $\sigma_{x_k}^2$, can be extracted from the diagonal of the covariance matrix, $\hat{\mathbf{P}}_k$.

   Make one plot for each value of $r_{\max} = 5, 3, 1$. Repeat for y and θ. That's 9 plots total.

   (b) Repeat (a) with $\hat{\mathbf{x}}_0 = (1, 1, 0.1)$, i.e., a poor initial condition. See how far you can push the initial condition away and still have your estimator converge.

   (c) Repeat (a) where you evaluate all the Jacobians at the *true robot state*, $\mathbf{x}_k$, instead of the estimated robot state, $\hat{\mathbf{x}}_k$. This is the CRLB version of the EKF. Contrast this to (a).

4 5. Make an animation of your EKF estimate for the case $r_{\max} = 1$ and $\hat{\mathbf{x}}_0 = \mathbf{x}_0, \hat{\mathbf{P}}_0 = \text{diag}\{1, 1, 0.1\}$. It should display the following information:

   - true landmarks positions; static dots (black)
   - true robot position, (x_k, y_k); a moving dot (blue)
   - estimated robot position, $(\hat{x}_k, \hat{y}_k)$; a moving dot (red)
   - 3-sigma covariance ellipse (use the top-left 2×2 of $\hat{\mathbf{P}}_k$); this should be centered on the estimated robot position; we want to see that the true robot position stays within this ellipse (but it probably won't); a moving ellipse (red)

   Hand this in as a movie file and make sure it can play in VLC.

Attach your code in an Appendix.