Coding Problem Set 4 - EKF/UKF Landmark Localization

Joshua Mangelson (Adapted from Ryan Eustice)

September 12, 2022

Purpose: The purpose of this lab is to give you hands on experience implementing and the ability to play around with landmark-based EKF and UKF localization.

1 Setup and Initial Instructions

1.1 Pulling the Code

You will be able to access the code for each of the labs in this course via Git. For this lab specifically, you will be able to access it via the following link: `https://bitbucket.org/byu_mobile_robotic_systems/lab4-ekf-ukf`

2 Submission Instructions

Your assignment must be received by 11:55p on the deadline posted on Learning Suite. You are to upload your assignment directly to LearningSuite with the following three attachments:

1. A .tgz or .zip file *containing a directory* named after your netid with the structure shown below:

```
alincoln_lab4.tgz:
alincoln_lab4/
alincoln_lab4/code/__init__.py
alincoln_lab4/code/run.py
alincoln_lab4/code/ekfUpdate.py
alincoln_lab4/code/ukfUpdate.py
alincoln_lab4/code/pfUpdate.py
alincoln_lab4/code/fieldSettings.py
alincoln_lab4/code/generator.py
alincoln_lab4/code/helpers.py
alincoln_lab4/data/data.npz
alincoln_lab4/video_ekf.{avi,mp4,gif}
alincoln_lab4/video_ukf.{avi,mp4,gif}
```

2. A PDF with the written portion of your writeup. Scanned versions of hand-written documents, converted to PDFs, are perfectly acceptable. No other formats (e.g. .doc) are acceptable. Your PDF file should adhere to the following naming convention: alincoln_lab4.pdf.

3. A 1 minute video of yourself demonstrating your completed lab and your understanding of the topics explored. Your video should adhere to the following naming convention: alincoln_summary.{avi,mp4}

3 Lab Overview

The key goal of this exercise is to get an understanding of the properties of Extended Kalman Filters (EKFs) and Unscented Kalman Filters (UKFs) for state estimation. You should try to "play around" with the parameters of each algorithm, so as to see how they deal with different levels of noise. For this task you will be implementing landmark-based robot localization (Fig. 1) using an EKF and UKF with known data association. Essentially, your job is to replicate the results shown in figures 7.11 and 7.15 in the ProbRob text.

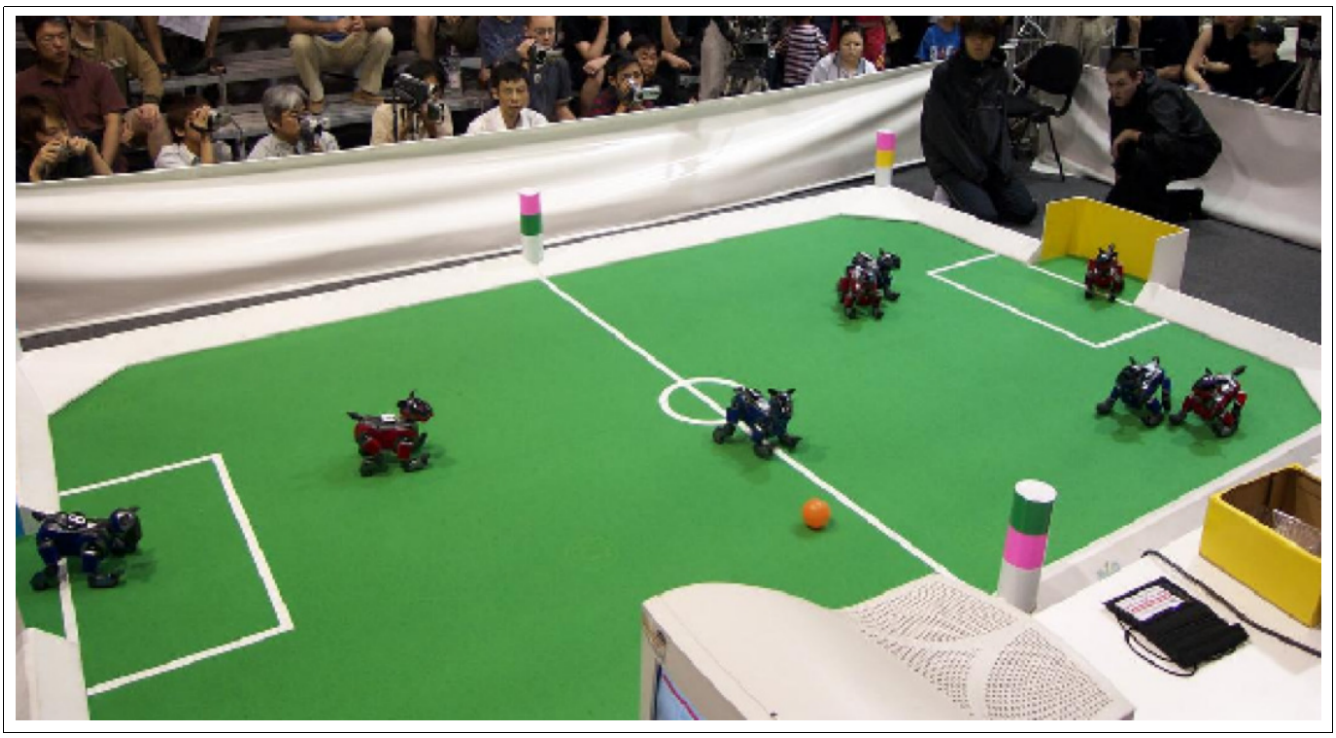

Figure 1: Beacon-based robot localization.

3.1 Code

The bitbucket repository linked to above contains a folder called *code* with a collection of Python files that replicate the landmark-based planar localization simulation used in Chapters 7 and 8 of the ProbRob text. Below you will find descriptions of the files included in the .zip file. You may end up not using every single file. Some are utilities for other files, and you don't really need to bother with them. Some have useful utilities, so you won't have to reinvent the wheel. Some have fuller descriptions in the files themselves.

Things to Implement

- *run.py* – Main update loop, should call *ekfUpdate*() and *ukfUpdate*()
- *ekfUpdate.py* - EKF update
- *ukfUpdate.py* - UKF update

Utilities (you should not need to modify these files)

- *fieldSettings.py* – defines the field
- *generator.py* – generates data according to initial mean and noise parameters
- *helpers.py* – contains many useful helper functions that you might want to utilize
- *testbench.py* – a blank script file you can use to test your functions
- *pfUpdate.py* – particle filter update

Please take a look at *helpers.py* to get a good handle on the functions there that you can use.

Data Format

- State: $[x, y, \theta]$; (cm, cm, radians) - Unknown to Your Filter
- Observation: [bearing to landmark, landmark ID]; (radians, integer) - Noisy Version Known to Your Filter
- Motion Control: $[\delta_{rot1}, \delta_{trans}, \delta_{rot2}]$; (radians, cm, radians) - Noise-free Version Known to Filter (Real motion followed by robot is corrupted by noise)

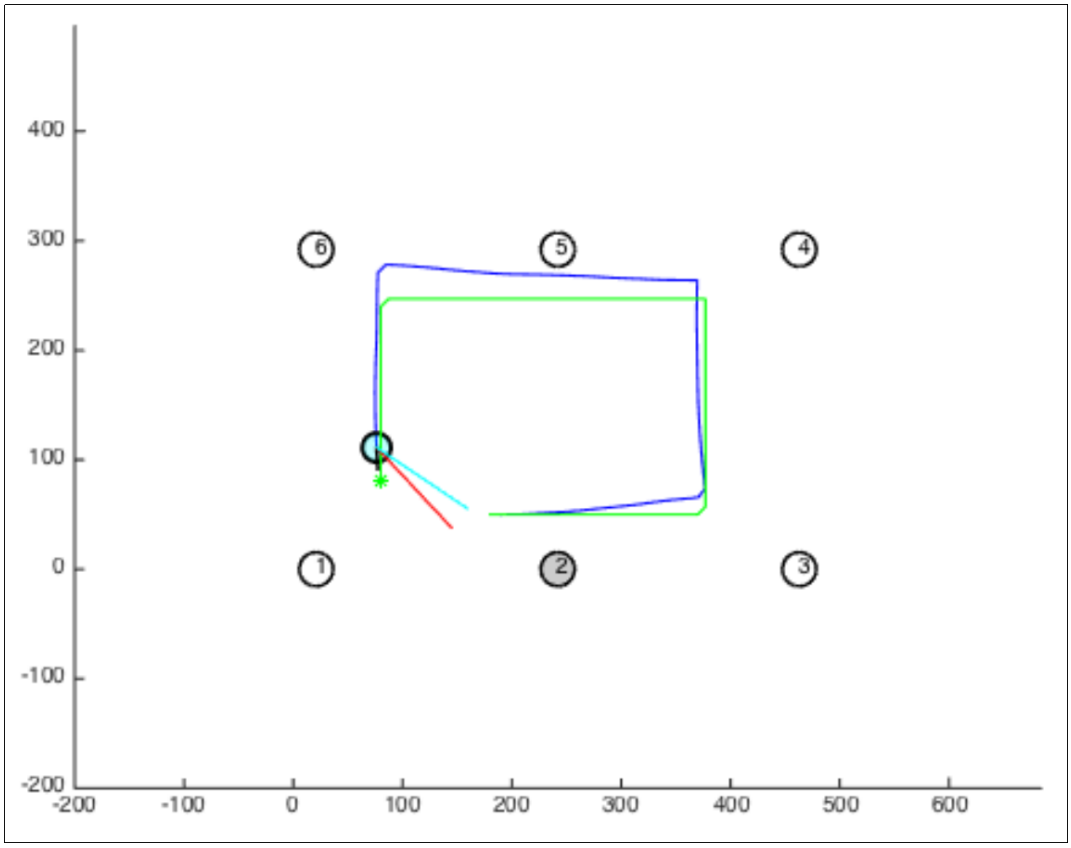

Figure 2: Robot simulator used for this lab.

4 Main Task - EKF & UKF Localization (25 points)

The function *run.py* generates motion information according to the odometry-based motion model of Chap- ter 5. Observations are landmark detections sensed through noisy bearing measurements. Each landmark has a unique ID and so data association is known for this exercise. Calling *./run.py* will generate a random simulation of 100 time steps. Here's a description of what you will see (refer to Fig. 2):

- The numbered circles represent landmarks 1 thru 6. The circle colored in gray is the landmark currently being observed.
- The light blue circle represents the current robot pose—orientation is depicted by the black bar.
- The green path is the ideal trajectory of the robot had there been no noise in realizing the motionCommand variable. This is where the robot would "think" it is based upon the commanded odometry sequence.
- The blue path is the actual trajectory of the robot—this state is hidden to the filter and is known only to the simulation. This is the actual path that the robot took because of noise in executing the motionCommand variable.
- The cyan line is the true, noise-free, landmark bearing observation, which is unavailable to the filter.
- The red line is the noise corrupted landmark bearing observation variable, this is available to the filter.

For this exercise, the *data* folder contains a data file named *data.npz*. Calling *./run -d ../data/data.npz* will run the simulation for the given action/observation sequence stored in *data*. Your job is to write a UKF and EKF for robot localization using this data sequence. You can make the plotting run faster using the *-w* flag - see *./run.py -h*.

In order to get quantitative results, you should generate plots that show the different evaluation criteria on the abscissa, and the corresponding filter error on the ordinate. For instance, for a particle filter you could plot number of samples versus average localization error, where error is measured by the distance between the true robot position

and the mean estimate of the particles. In particular, modify *run.py* to generate the following plots, which should be included in your submitted report:

1. Implement EKF and UKF-based robot localization using odometry and bearing-only observations to features in a landmark map.

   (a) For the EKF and UKF, plot the filter estimated mean position and 3-sigma covariance ellipsoid overlaid ontop of the simulation figure at every time step. If your filter is working correctly, the robot should lie within the 3-sigma ellipse at least 98.89% of the time.

   Include videos in your submission for the EKF and UKF under nominal conditions.

2. Create plots of pose error versus time i.e., a plot of $\hat{x} - x$ vs. $t, \hat{y} - y$ vs t, and $\hat{\theta} - \theta$ vs. t where $(\hat{x}, \hat{y}, \hat{\theta})$ is the filter estimated pose and (x, y, θ) is the ground-truth actual pose known only to the simulator. Plot the error in blue and in red plot the $\pm 3\sigma$ uncertainty bounds. Your state error should lie within these bounds approximately 99.73% of the time (assuming Gaussian statistics).

3. Once your filters are implemented, please investigate some properties of them. How do they behave

   (a) as the sensor or motion noise go toward zero?

   (b) if the noise parameters underestimate or overestimate the true noise paramters?

5 Extra Credit Option 1 (3 extra points)

Additionally implement particle filter localization by completing the *pfUpdate.py* file and updating *run.py*.

1. Plot the sample distribution every time step, it should be centered on the robot's true position.

2. Plot the x, y, and θ error as before, but use the sample mean and variance of the particles.

3. Answer how the filter behaves as the number of particles got to zero.

4. Submit the *pfUpdate.py* and *resample.py* files with your submission.

5. Also a video of your system running entitled *video_pf.{avi,mp4,gif}*

6 Extra Credit Option 2 (1 for EKF/UKF only, 2 for EKF/UKF/PF)

Try to see how the different filters handle the global localization and the kidnapped robot problems.