CSE 571 - Robotics
Homework 2 - EKF and Particle Filter for Localization

Due Wednesday May 11th @ 11:59pm

The key goal of this homework is to get an understanding of the properties of Kalman filters and Particle filters for state estimation. There are two parts to the homework - a written assignment and a programming assignment. There are two written questions which help you derive some useful quantities for the EKF implementation. In the programming assignment, you will be implementing an Extended Kalman Filter (EKF) and a Particle Filter (PF) for landmark based localization. You will also analyze their performance under various conditions. The zip file containing the starter code for this homework can be found at `https://courses.cs.washington.edu/courses/cse571/22sp/homeworks/assignment2.zip`.

Useful reading material: Lecture notes, Chapters 3,4,5,7 & 8 of Probabilistic Robotics, Thrun, Burgard and Fox (pdf shared with class).

Collaboration: Students can discuss questions, but each student MUST write up their own solution, and code their own solution. We will be checking code/PDFs for plagiarism.

Late Policy: This assignment may be handed in up to 5 days late (May 16th @ 11:59pm). If you have used up your 6 late days this quarter, there will be a penalty of 10% of the maximum grade per day.

1 Writing assignments

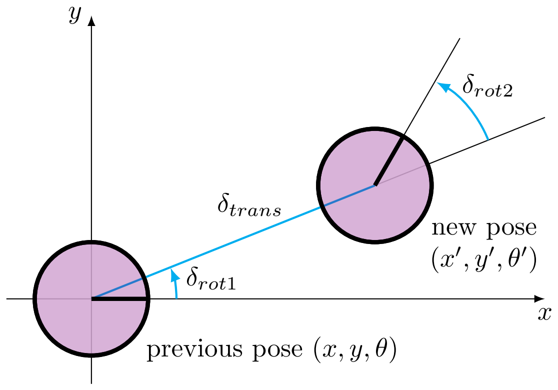

Figure 1: Odometry-based motion model

1.1 Motion Model Jacobian [10 points]

Figure 1 describes a simple motion model. The state of the robot is its 2D position and orientation: $s = [x, y, \theta]$. The control to the robot is $u = [\delta_{rot1}, \delta_{trans}, \delta_{rot2}]$, i.e. the robot rotates by δ_{rot1}, drives straight forward δ_{trans}, then rotates again by δ_{rot2}.

The equations for the motion model g are as follows:

$$\begin{aligned}
x_{t+1} &= x_t + \delta_{trans} * cos(\theta_t + \delta_{rot1}) \\
y_{t+1} &= y_t + \delta_{trans} * sin(\theta_t + \delta_{rot1}) \\
\theta_{t+1} &= \theta_t + \delta_{rot1} + \delta_{rot2}
\end{aligned}$$

where $s_{t+1} = [x_{t+1}, y_{t+1}, \theta_{t+1}]$ is the prediction of the motion model. Derive the Jacobians of g with respect to the state $G = \frac{\partial g}{\partial s}$ and control $V = \frac{\partial g}{\partial u}$:

$$G = \begin{pmatrix} \frac{\partial x'}{\partial x} & \frac{\partial x'}{\partial y} & \frac{\partial x'}{\partial \theta} \\ \frac{\partial y'}{\partial x} & \frac{\partial y'}{\partial y} & \frac{\partial y'}{\partial \theta} \\ \frac{\partial \theta'}{\partial x} & \frac{\partial \theta'}{\partial y} & \frac{\partial \theta'}{\partial \theta} \end{pmatrix} \qquad V = \begin{pmatrix} \frac{\partial x'}{\partial \delta_{rot1}} & \frac{\partial x'}{\partial \delta_{trans}} & \frac{\partial x'}{\partial \delta_{rot2}} \\ \frac{\partial y'}{\partial \delta_{rot1}} & \frac{\partial y'}{\partial \delta_{trans}} & \frac{\partial y'}{\partial \delta_{rot2}} \\ \frac{\partial \theta'}{\partial \delta_{rot1}} & \frac{\partial \theta'}{\partial \delta_{trans}} & \frac{\partial \theta'}{\partial \delta_{rot2}} \end{pmatrix}$$

1.2 Observation Model Jacobian [10 points]

Assume there is a landmark l at position (l_x, l_y).

1. Suppose the robot receives measurement of the landmark in terms of the bearing angle β. Write down the estimation for β in terms of the landmark's position (l_x, l_y) and the robot's state (x_t, y_t, θ_t).
2. Derive Jacobian of the measurement with respect to the robot state:

$$H = \begin{bmatrix} \frac{\partial \beta}{\partial x_t} & \frac{\partial \beta}{\partial y_t} & \frac{\partial \beta}{\partial \theta_t} \end{bmatrix}$$

2 Programming assignments

In the programming component of this assignment, you will implement an Extended Kalman Filter (EKF) and Particle Filter (PF) for localizing a robot based on landmarks.

We will use the odometry-based motion model you derived in question 1.2. We assume that there are landmarks present in the robot's environment. The robot receives the bearings (angles) to the landmarks and the ID of the landmarks as observations: (bearing, landmark ID).

We assume a noise model for the odometry motion model with parameters α ([1] Table 5.6) and a separate noise model for the bearing observations with parameter β ([1] Section 6.6). See the provided starter code for implementation details.

At each timestep, the robot starts from the current state and moves according to the control input. The robot then receives a landmark observation from the world. You will use this information to localize the robot over the whole time sequence with an EKF and PF.

Code Overview

The starter code is written in Python and depends on NumPy and Matplotlib. The conda environment you installed for HW1 should suffice for this homework. This section gives a brief overview of each file.

- `localization.py` – This is your main entry point for running experiments.
- `soccer_field.py` – This implements the dynamics and observation functions, as well as the noise models for both. Add your Jacobian implementations here!
- `utils.py` – This contains assorted plotting functions, as well as a useful function for normalizing an angle between $[-\pi, \pi]$.
- `policies.py` – This contains a simple policy, which you can safely ignore.
- `ekf.py` – Add your extended Kalman filter implementation here!
- `pf.py` – Add your particle filter implementation here!

Command-Line Interface

To visualize the robot in the soccer field environment, run

```
$ python localization.py --plot none
```

The blue line traces out the robot's position, which is a result of noisy actions. The green line traces the robot's position assuming that actions weren't noisy. After you implement a filter, the filter's estimate of the robot's position will be drawn in red.

```
$ python localization.py --plot ekf
$ python localization.py --plot pf
```

To see other command-line flags available to you, run

```
$ python localization.py -h
```

Data Format

- state: $[x, y, \theta]$
- control: $[\delta_{rot1}, \delta_{trans}, \delta_{rot2}]$
- observation: $[\theta_{bearing}]$

Hints

- Make sure to call `utils.minimized_angle` any time an angle or angle difference could exceed $[-\pi, \pi]$.
- Make sure to use the low-variance systematic sampler from lecture. It gives you a smoother particle distribution and also requires only a single random number per resampling step.
- Turn off plotting for a significant speedup.

2.1 EKF Implementation [40 points]

Implement the extended Kalman filter algorithm in `ekf.py`. You will need to fill in `ExtendedKalmanFilter.update` and the `Field` methods `G`, `V`, and `H`. Your results from a successful EKF implementation should be comparable to the following results.

```
$ python localization.py ekf --seed 0
...
Mean position error: 12.447285515095215
Mean Mahalanobis error: 7.011714529349076
ANEES: 2.337238176449692
```

(a) Plot the real robot path and the filter path under the default (provided) parameters.

(b) Plot the position error and ANEES as the noise factors for both the data and the filter range over $r = [1/64, 1/16, 1/4, 4, 16, 64]$. You should run 10 trials per value of r with different random seeds and plot the mean (optionally standard deviation). One run might look something like:

```
$ python localization.py ekf --data-factor 4 --filter-factor 4
```

Note that the value for `--data-factor` should be the same as `--filter-factor`. Discuss anything interesting you observe.

(c) Plot the position error and ANEES (average normalized estimation error squared) as the filter noise factors α, β vary over r (as above) while the data noise factor is kept as default. You should run 10 trials per value of r and plot the mean (optionally standard deviation). Note that you should not pass the `--data-factor` argument here. How does the results compare with what you get using the default parameters? Discuss anything interesting you observe.

2.2 PF Implementation [40 points]

Implement the particle filter algorithm in `pf.py`. You will need to fill in `ParticleFilter.update` and `ParticleFilter.resample`.

```
$ python localization.py pf --seed 0
...
Mean position error: 10.09583008726548
Mean Mahalanobis error: 9.054361831769514
ANEES: 3.018120610589838
```

(a) Plot the real robot path and the filter path under the default parameters.

(b) Plot the position error and ANEES as both the data and the filter noise factors range over $r = [1/64, 1/16, 1/4, 4, 16, 64]$. You should run 10 trials per value of r and plot the mean (optionally standard deviation). Discuss anything interesting you observe.

(c) Plot the position error and ANEES as the filter noise factors α, β vary over r (as above) while the data noise factor is kept as default. You should run 10 trials per value of r and plot the mean (optionally standard deviation). How does the results compare with what you get using the default parameters? Discuss anything interesting you observe.

(d) Plot the mean position error and ANEES as α, β range over r (as in (c)) and the number of particles varies over $[20, 50, 500]$. You should run 10 trials per value of r and plot the mean (optionally standard deviation). You should include three plots, one per each particle quantity. Discuss anything interesting you observe.

3 Submission

You will be using Gradescope `https://www.gradescope.com/` to submit the homework. Please submit the written assignment answers as a PDF. For the code, please **submit ekf.py, pf.py, and soccer_field.py**.

References

[1] Thrun, Burgard and Fox (2005), Probabilistic Robotics.