NA 568 Mobile Robotics: Methods & Algorithms Winter 2024 – Homework 3 – Localization

Maani Ghaffari, Minghan Zhu
University of Michigan

Feb 16, 2024

This is a reminder that no late HW is accepted. We are using Gradescope for turning in HW; see relevant information on the course Canvas site. For Homeworks, the lowest grade will be automatically dropped for everyone. SLAM homework cannot be dropped (mandatory). Deadlines are on each Friday at 11:55 pm (EST) and Gradescope submission remains open until Monday at 9 am (EST) with no penalty.

You are encouraged to talk at the conceptual level with other students, but you must complete all work individually and may not share any non-trivial code or solution steps. See the syllabus for the full collaboration policy.

Submission Instructions

Your assignment must be received by 11:55 pm on Friday, March 8th (Anywhere on Earth Time). This is selected out of fairness to all our students, including those who take the course remotely. You are to upload your assignment directly to the Gradescope website:

1. Attach the five Python files mentioned in the homework for submission, including **filter/EKF.py**, **filter/UKF.py**, **filter/PF.py**, **filter/InEKF.py**, and **utils/utils.py**. You can either submit them via a `.zip` file containing all five files or separately submit each of the five files.
2. We only require the code for submission, but students are welcome to use visualization or plotting tools to aid their development. Please refer to the "Guide to Develop and Execute HW Python Code".

Announcement

Change log : March 1, 2024

- Revision made in `filter/PF.py, data/generate_data.py, utils/filter_initialization.py`:
  - To ensure deterministic results of the filter in the autograder, a random seed is assigned in the filter and data generation part.
  - If you want to observe varying results in different runs caused by the randomness, you can unset the random seed. **Set the random seed** when you submit your code to the autograder.

The revised version of HW3 has been posted on Canvas. Students are still required to submit five Python files, including **filter/EKF.py**, **filter/UKF.py**, **filter/PF.py**, **filter/InEKF.py**, and **utils/utils.py**. Students who have not yet begun working on the particle filter are advised to begin their work with this file. For students who have successfully passed the particle filter test, there is no requirement for resubmission.

1 Velocity Motion Model

This is the velocity motion model from Chapter 5 of the Probabilistic Robotics book. The discrete dynamical equations for the state of a robot given an input of linear and angular velocities (v and ω, respectively) are as follows.

$$x_{k+1} = x_k - \frac{\hat{v}}{\hat{\omega}}\sin(\theta_k) + \frac{\hat{v}}{\hat{\omega}}\sin(\theta_k + \hat{\omega}\Delta t), \tag{1}$$

$$y_{k+1} = y_k + \frac{\hat{v}}{\hat{\omega}}\cos(\theta_k) - \frac{\hat{v}}{\hat{\omega}}\cos(\theta_k + \hat{\omega}\Delta t), \tag{2}$$

$$\theta_{k+1} = \theta_k + \hat{\omega}\Delta t + \hat{\gamma}\Delta t. \tag{3}$$

The variable γ is considered a third input to the system, but its commanded value is always zero. This command is still necessary when deriving the Jacobians for this motion model. This input is necessary because, without it, the posterior pose will be located on a two-dimensional manifold and thus will be degenerate (causing the filter not to be able to estimate the heading correctly). This is described further on page 129 of Probabilistic Robotics. As seen in Probabilistic Robotics, this type of circular motion model is used so the robot can rotate and translate concurrently.

$$\hat{v} = v + \epsilon_v\ , \quad \epsilon_v \sim \mathcal{N}(0, \alpha_1 v^2 + \alpha_2 \omega^2), \tag{4}$$

$$\hat{\omega} = \omega + \epsilon_\omega\ , \quad \epsilon_\omega \sim \mathcal{N}(0, \alpha_3 v^2 + \alpha_4 \omega^2), \tag{5}$$

$$\hat{\gamma} = \epsilon_\gamma\ , \quad \epsilon_\gamma \sim \mathcal{N}(0, \alpha_5 v^2 + \alpha_6 \omega^2). \tag{6}$$

In this velocity motion model, the motion noise terms are indicated in (4), (5), and (6), where ϵ_v, ϵ_ω and ϵ_γ are uncorrelated Gaussian noises with 0 mean, and $\alpha_1, \dots, \alpha_6$ are robot-specific error parameters. This is described further on pages 127-129 of Probabilistic Robotics.

2 Measurement Model

For the implementation of EKF, UKF, and PF, We use the following measurement model that is similar to that shown on the Nonlinear Kalman Filtering slides, where (m_x, m_y) is the known position of the observed landmark.

$$z_k = \begin{bmatrix} \operatorname{atan2}(m_y - y_k, m_x - x_k) - \theta_k \\ \sqrt{(m_y - y_k)^2 + (m_x - x_k)^2} \end{bmatrix} + q_k\ , \quad q_k \sim \mathcal{N}(0, Q_k). \tag{7}$$

For the implementation of RI-EKF, we use the one shown on Slide 33 of the Invariant EKF slides.

$$Y_k = \bar{X}_k^{-1} b + \tilde{V}_k$$

$$\begin{bmatrix} y_k^1 \\ y_k^2 \\ 1 \end{bmatrix} = \begin{bmatrix} \bar{R}_k^\top & -\bar{R}_k^\top p_k \\ 0 & 1 \end{bmatrix} \begin{bmatrix} m \\ 1 \end{bmatrix} + \begin{bmatrix} v_k \\ 0 \end{bmatrix}, \quad v_k \sim \mathcal{N}(0, V_k) \tag{8}$$

3 Motion Model using 2D Special Euclidean Group

Now we model the the motion model using SE(2). This model correctly respects the geometry of the 2D plane and 3 DOF of the robot motion. The discrete-time motion model of the robot is given by (using the right-invariant definition of the error)

$$\begin{aligned} X_{k+1} &= X_k \exp(\xi_k^\wedge \Delta t) \\ \Sigma_{k+1} &= \Sigma_k + \mathrm{Ad}_{X_k} W_k \mathrm{Ad}_{X_k}^\mathsf{T} \end{aligned} \tag{9}$$

where the robot pose $X_k = \begin{bmatrix} R_k & p_k \\ 0 & 1 \end{bmatrix} \in \mathrm{SE}(2)$, the twist $\xi_k^\wedge = \begin{bmatrix} \omega_k^\wedge & v_k \\ 0 & 0 \end{bmatrix} \in \mathfrak{se}(2)$ (or $\xi_k = \mathrm{vec}(v_k, \omega_k) \in \mathbb{R}^3$), with the motion model noise $w_k \sim \mathcal{N}(0, W_k)$.

Task 1: Derive the propagation and correction equations for the following filters.

A. Extended Kalman Filter (EKF) using the velocity motion model and measurement model in (7).

B. Unscented Kalman Filter (UKF) using the velocity motion model and measurement model in (7).

C. Particle Filter (PF) using the velocity motion model and measurement model in (7).

D. Right-Invariant EKF (RI-EKF) using the SE(2) motion model and right-invariant measurement model in (8).

Task 2: Use the derived filters to localize the robot within the given map.

The environment is composed of six landmarks and is shown in Figure 1. The robot has a desired trajectory which can be seen in green. Due to noise in the robot actuators, it does not follow the given trajectory. The filtering process for the EKF, UKF, and PF is as follows,

- The robot moves, and an onboard sensor measures the linear and angular velocities in the body frame of the robot. The motion measurements are input to the filter as a 3 x 1 vector where the values are $[v, \omega, \gamma]$ as described in the motion model. The additional angular velocity, γ, is also an input because of the degeneracy issue described in the motion model.
- The robot obtains noisy range and bearing measurements to two given landmarks in the environment. For each landmark, the measurement order is $[\mathrm{bearing}, \mathrm{range}, \mathrm{Landmark\_ID}]$.
- The velocity and sensor readings are used for the prediction and correction steps of the filter, respectively.
- The robot moves again, and the process repeats.

The filtering process when using the Invariant EKF is slightly different. It is as follows.

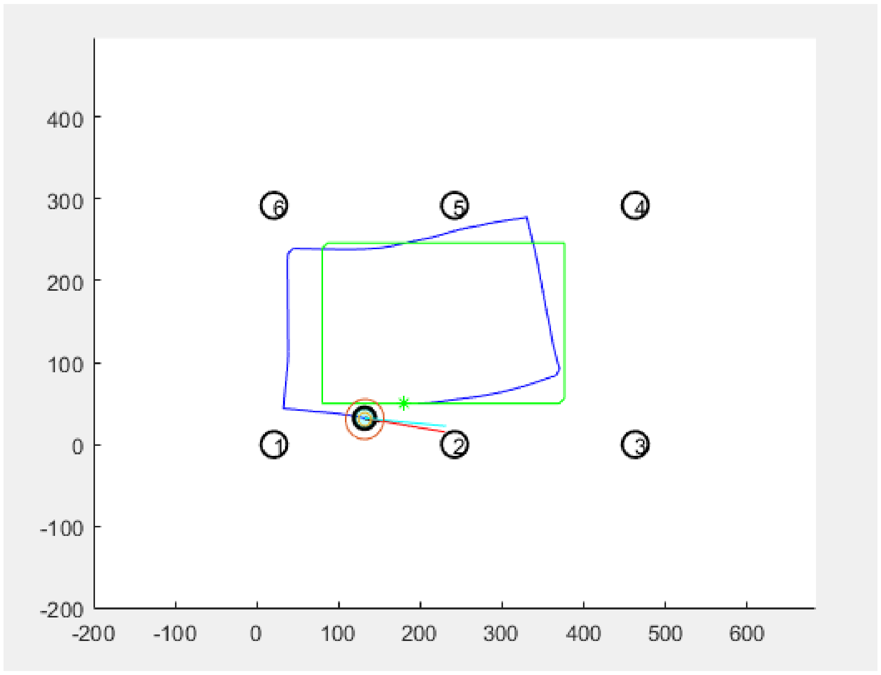

Figure 1: The simulated environment for landmark localization.

- The robot moves, and an onboard sensor measures the linear and angular velocities in the body frame of the robot. The motion measurements are input to the filter as a 3 x 1 vector where the values are $[v, \omega, \gamma]$ as described in the motion model. The additional angular velocity, γ, is also an input because of the degeneracy issue described in the motion model.
- The angular and linear velocity measurements are converted to the format of Lie algebra of SE(2) via the velocity motion model (provided in the code). This describes the robot's motion in the body frame and is needed for the prediction step of the filter.
- The robot obtains two noisy measurements, which are the relative X and Y locations of two separate landmarks in the body frame of the robot. The marker ID for each landmark is also received. Two landmarks are necessary since the Observability Gramian is not full rank when only one landmark is viewed (see slides 35-36 of the Invariant EKF slides).
- The twist matrix describing relative motion is used for the prediction step. The relative landmark measurements (denoted Y1 and Y2 in the code) and global landmark locations (denoted landmark_x, landmark_y, landmark_x2, landmark_y2 in the code) are used in the correction.
- The robot moves again, and the process repeats.

Other important information is as follows,

- The Δt seen in the discrete dynamical equations is one second.
- You will have to try different values for sensor noise to find the optimal values for the filters.
- The filters are considered consistent if the ground truth robot pose is always maintained within the 3-sigma contour of the multivariate Gaussian distribution over the state. You should see your robot stay within the ellipse.

You can implement your code in **Python**. Below is a list of some instructions for running the code. Also, please refer to the README.md file for information on how to run Python3 with ROS for this homework.

Guide to Develop and Execute HW Python Code

- The filters are initialized using the functions `utils/filter_intialization.py` and `utils/system_initialization.py`. You may need to view these functions since the initialized properties are used in the filtering process.
- An example of how to run the code is as follows, **python3 run.py**.
  This will run the environment for 100 time-steps and localize using a dummy filter. To switch between filters, change `filter_name` in `config/settings.yaml` to either `"EKF"`, `"UKF"`, `"PF"` or `"InEKF"`.
- You may freely use the function **plot_error** in `utils/utils.py` to visualize and observe the results of your code. You may refer to an example code to run **plot_error** which is included in the `RobotSystem.py` script.

In each score distribution, the points are allocated as follows:

- **EKF**: Prediction (12 points), Correction (13 points)
- **UKF**: Prediction (12 points), Correction (13 points)
- **PF**: Prediction (12 points), Correction (13 points)
- **InEKF**: Propagation (8 points), Prediction (8 points), Correction (9 points)

Additionally, there are 10 extra points available, bringing the total to 110 points!

A. (25 pts) Fill in the prediction and correction functions within the EKF class located in the file:
Python: `filter/EKF.py`.

B. (25 pts) Fill in the prediction and correction functions within the UKF class located in the file:
Python: `filter/UKF.py`.

C. (25 pts) Fill in the prediction and correction functions within the PF class located in the file:
Python: `filter/PF.py`.

D. (25 pts) Fill in the prediction, correction, and propagation functions within the InEKF class located in the file:
Python: `filter/InEKF.py`.

E. (10 Extra credits) **The covariance while using the InEKF is kept within the Lie algebra of the matrix group.** We wish to map the covariance from Lie algebra to Cartesian coordinates, (x, y, θ).

For extra credits, update the function `lieToCartesian()`, `func()` in `utils/utils.py` to map the covariance from Lie algebra to Cartesian coordinates, and then use the function `mahalanobis()` to measure the performance of the filter. Save the Cartesian mean and covariance into `mu_cart` and `Sigma_cart`, respectively. (If you choose to complete this, change `Lie2Cart` in `config/settings.yaml` to `True`.)

FAQ

1. Q: Why my filter works pretty well, but the covariance of the heading is very large?
   A: Please check your heading angle is wrapped between $-\pi$ to π using the provided wrapToPi function or an equivalent.

2. Q: In UKF implementation, why the green (red if using Python) ellipse is too small/large or moving away?
   A: If the green ellipse follows your trajectory, please tune your motion noise to make it more visible. If it moves away, please check your implementation.

3. Q: In the PF task, should I use sequential or batch update?
   A: Either way is acceptable.

4. Q: In the InEKF task, the marker runs very slowly in Rviz compared to other filters. Mine is updated every 1-2 seconds instead of the 2-3 updates per second for other filters. Is this expected, or if this points to an error in my implementation?
   A: That is expected. We won't check runtime performance. It's mainly because our code has to transform the covariance from Lie to Cartesian and plot it using Markers in ROS.

5. Q: In the EKF task, the information matrix $S = H\Sigma^{-1}H^\top + VRV^\top$ where $V = \frac{\partial h}{\partial v}$ and R is the sensor noise covariance. In the stack form, this leads to a 4x4 matrix for the first term ($H\Sigma^{-1}H^\top$). But I'm unsure what the $\bar{R}$ is in stacked form.
   A: R_k is the covariance on the noise q_k. When using the stacked measurement model, we need to look at the derivation behind its closed-form solution. After linearization, the measurement is written as $z_k = H_k x_{k|k-1} + q_k$ and the innovation $\nu_k = z_k - \hat{z}_k$ can be reformulated as:

$$\begin{aligned}\nu_k &= z_k - H_k \mu_{k|k-1} \\ &= H_k x_{k|k-1} - H_k \mu_{k|k-1} + q_k \\ &= H_k \left(x_{k|k-1} - \mu_{k|k-1}\right) + q_k\end{aligned}$$

   The mean of the innovation can be calculated as follows:

$$\begin{aligned}\mathbb{E}[\nu_k] &= \mathbb{E}[H_k \left(x_{k|k-1} - \mu_{k|k-1}\right) + q_k] \\ &= H_k \mathbb{E}[x_{k|k-1} - \mu_{k|k-1}] + \mathbb{E}[q_k] \\ &= H_k \left(\mathbb{E}[x_{k|k-1}] - \mathbb{E}[\mu_{k|k-1}]\right) \quad \text{(Since } q_k \text{ is zero-mean white noise)} \\ &= H_k \left(\mu_{k|k-1} - \mu_{k|k-1}\right) = 0\ .\end{aligned}$$

Thus, the innovation covariance S_k is written as follows:

$$\begin{aligned}
S_k &= \mathbb{E}[\left(H_k\left(x_{k|k-1} - \mu_{k|k-1}\right) + q_k\right)\left(H_k\left(x_{k|k-1} - \mu_{k|k-1}\right) + q_k\right)^\top] \\
&= \mathbb{E}[H_k\left(x_{k|k-1} - \mu_{k|k-1}\right)\left(x_{k|k-1} - \mu_{k|k-1}\right)^\top H_k^\top] + \mathbb{E}[q_k q_k^\top] \\
&+ \mathbb{E}[q_k\left(x_{k|k-1} - \mu_{k|k-1}\right)^\top H_k^\top] + \mathbb{E}[H_k\left(x_{k|k-1} - \mu_{k|k-1}\right)q_k^\top] \\
&= H_k \Sigma_{k|k-1} H_k^\top + \mathbb{E}[q_k q_k^\top] + \mathrm{Cov}\left(q_k, x_{k|k-1}\right)H_k^\top + H_k \mathrm{Cov}\left(x_{k|k-1}, q_k\right) \\
&= H_k \Sigma_{k|k-1} H_k^\top + \mathbb{E}[q_k q_k^\top] \quad \text{(Since } x_{k|k-1} \text{ and } q_k \text{ are uncorrelated)} \\
&= H_k \Sigma_{k|k-1} H_k^\top + R_k.
\end{aligned}$$

Here, since we are using the stacked measurement model, we have $q_k = \begin{bmatrix} q_1 & q_2 \end{bmatrix}_k^\top$ where $q_1 \in \mathbb{R}^{2\times 1}$ is the noise on the first measurement and $q_2 \in \mathbb{R}^{2\times 1}$ is the noise on the second measurement. Thus:

$$\mathbb{E}[q_k q_k^\top] = \mathbb{E}[\begin{bmatrix} q_1 q_1^\top & q_1 q_2^\top \\ q_2 q_1^\top & q_2 q_2^\top \end{bmatrix}] = \begin{bmatrix} \mathbb{E}[q_1 q_1^\top] & \mathbb{E}[q_1 q_2^\top] \\ \mathbb{E}[q_2 q_1^\top] & \mathbb{E}[q_2 q_2^\top] \end{bmatrix}.$$

Since q_1 and q_2 are uncorrelated, i.e., $\mathbb{E}[q_1 q_2^\top] = R_k \delta_{12} = 0$, we have:

$$\bar{R}_k = \mathbb{E}[q_k q_k^\top] = \begin{bmatrix} \mathbb{E}[q_1 q_1^\top] & 0_{2\times 2} \\ 0_{2\times 2} & \mathbb{E}[q_2 q_2^\top] \end{bmatrix} = \begin{bmatrix} R_k & 0_{2\times 2} \\ 0_{2\times 2} & R_k \end{bmatrix}.$$