NA 568 Mobile Robotics: Methods & Algorithms
Winter 2020 – PS2

Maani Ghaffari
University of Michigan

January 27, 2020

This problem set counts 10% of your course grade. 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 Sunday, February 9. You are to upload your assignment directly to the Gradescope website as two attachments:

1. A `.tar.gz` or `.zip` file *containing a directory* named after your uniqname with the structure shown below.

   ```
   alincoln_ps2.tgz:
   alincoln_ps2/
   alincoln_ps2/EKF.m
   alincoln_ps2/UKF.m
   alincoln_ps2/PF.m
   alincoln_ps2/InEKF.m
   alincoln_ps2/lieTocartesian.m
   ```

2. A PDF with the written portion of your write-up. 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_ps2.pdf`.

Homework received after 11:55 pm is considered late and will be penalized as per the course policy. The ultimate timestamp authority is the one assigned to your upload by Gradescope. No exceptions to this policy will be made.

1 Velocity Motion Model

This is the velocity motion model from Chapter 5 of 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. It is still necessary to take this command into consideration 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 to not be able to estimate the heading correctly). This is described further on pg.129 of Probabilistic Robotics. This type of circular motion model as seen in Probabilistic Robotics is used so the robot can rotate and translate concurrently.

2 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)

$$\begin{aligned} X_{k+1} &= X_k exp(\xi_k \Delta t) \\ \Sigma_{k+1} &= \Sigma_k + \mathrm{Ad}_{X_k} Q_k \mathrm{Ad}_{X_k}^{\mathsf{T}} \end{aligned} \tag{4}$$

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 $\xi_k = \mathrm{vec}(\omega_k, v_k) \in \mathbb{R}^3$), and $w_k \sim \mathcal{N}(0, Q_k)$.

Task 1 (40 points): Derive the propagation and correction equations for the following filter.

A. (10 pts) Extended Kalman Filter (EKF) using the velocity motion model and measurement model displayed on Slide 19 of the Kalman filtering slides.

B. (10 pts) Unscented Kalman Filter (UKF) using the velocity motion model and measurement model displayed on Slide 19 of the Kalman filtering slides.

C. (10 pts) Particle Filter (PF) using the velocity motion model and measurement model displayed on Slide 19 of the Kalman filtering slides.

D. (10 pts) Right-Invariant EKF (RI-EKF) using the SE(2) motion model and right-invaraint measurement model seen in the Invariant EKF slides.

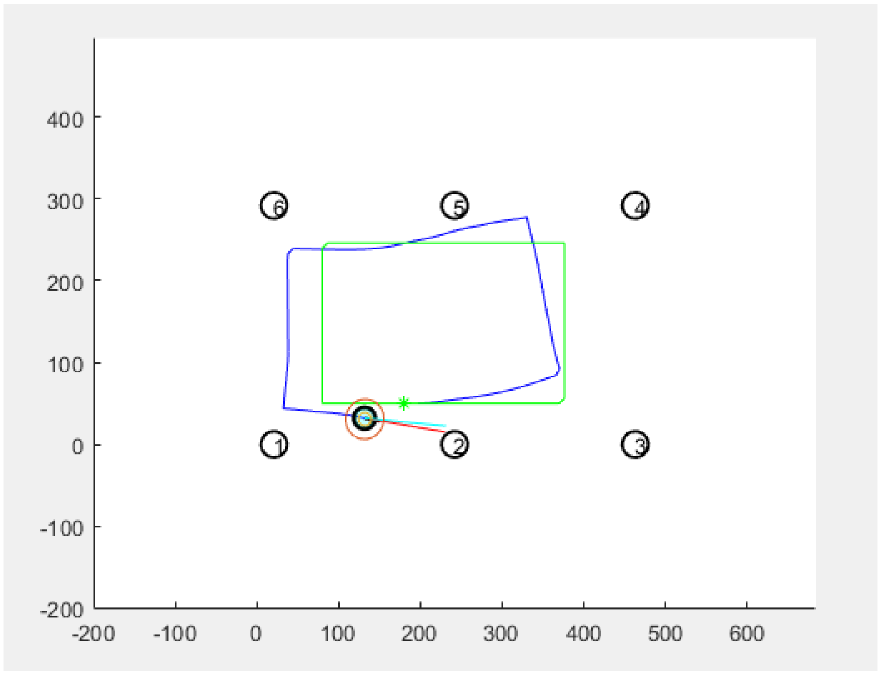

Figure 1: The simulated environment for landmark localization.

Task 2: Use the derived filters to localize the robot within the given map. Include the plots generated at the end of the run and a video for each filter. (60 points)

The environment is composed of six landmarks and 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 on-board 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 a given landmark in the environment. The measurement order is [bearing range 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,

- The robot moves and an on-board sensor measures the linear and angular velocities in the body frame of the robot.
- This angular and linear velocity measurements (and γ) must be converted to a skew symmetric matrix in the lie algebra of SE2. This describes the motion of the robot 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.
- The skew symmetric 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 filters are initialized using the functions `filter_intialization.m` and `system_initialization.m`. You may need to view these functions since the initialized properties are used in the filtering process.
- The filters are considered to be 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. Run the `.p` code to see what this should look like.
- An example of how to run the code is as follows, `run(100,0,0,"EKF")`. This will run the environment for 100 time-steps and localize using the EKF filter. To switch between filters enter either `"EKF"`, `"UKF"`, `"PF"` or `"InEKF"` into the `run` function. Enter a `1` into the thrid input of the run function to create a video.
- 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.

A. (15 pts) Fill in the prediction and correction functions within the Matlab class EKF located in the file `EKF.m`.

B. (15 pts) Fill in the prediction and correction functions within the Matlab class UKF located in the file `UKF.m`.

C. (15 pts) Fill in the prediction and correction function within the Matlab class PF located in the file `PF.m`.

D. (15 pts) Fill in the prediction, correction and propagation functions within the Matlab class InEKF located in the file `InEKF.m`.

E. (5 Extra credits) The covariance while using the InEKF is kept within the Lie algebra of the matrix group. Update the function `lieToCartesian.m` in order to map the covariance from lie algebra to Cartesian coordinates, (x, y, θ), and then use the function `mahalanobis.m` to measure the performance of the filter. Include a plot of the filter performance. Save the Cartesian mean and covariance into `filter.mu_cart` and `filter.sigma_cart` respectively. (If you choose to complete this uncomment mahalanobis and lie to Cartesian functions in the InEKF filter switch.)