Problem Set 6

Due June 5, 2024 at 11:59 PM Pacific

Turn in your problem set by Gradescope. Include any code you write in your solutions as well. Mark down how much time you spent on the homework set.

1. [15 Points.] **Kalman Filter with GPS:** Consider a quadrotor drone flying at a constant altitude. Let $p_t \in \mathbb{R}^2$ be its position, and $s_t \in \mathbb{R}^2$ its velocity. Let's model its dynamics as a discrete time double integrator perturbed by noise,

$$P_{t+1} = P_t + \delta t S_t \tag{1}$$
$$S_{t+1} = S_t + \delta t u_t + W_t, \tag{2}$$

where $u_t \in \mathbb{R}^2$ is the acceleration commanded by a control system (or a remote human pilot), and δt is the discrete time step. The drone makes position measurements with GPS according to the noisy measurement model

$$Y_t = P_t + V_t. \tag{3}$$

The processes W_t, V_t are Gaussian white noise with covariances Q and R, respectively, uncorrelated with each other and uncorrelated with the initial state. Let $\delta t = 1s$, $Q = \mathrm{I}_2\, m^2/s^2$, $R = 9\,\mathrm{I}_2\, m^2$, $u_t = -2.5[\cos(0.05t) \quad \sin(0.05t)]^T m/s^2$.

(a) [2 Points.] Find the system matrices A, B, C, in terms of the given parameters.

(b) [2 Points.] Simulate the trajectory of the drone as well as the measurement process using the provided simulation code (see: `q1q2_simulator_script.py`). Let the initial position and velocity be $p_0 = [1000m \quad 0m]^T$, and $v_0 = [0m/s \quad 50m/s]^T$, respectively. Attach plots of a few example trajectories (position and velocity) along with the noisy GPS position measurements.

(c) [9 Points.] Suppose your initial guess about the state of the drone has a mean $\mu_0 = [1500m \quad 100m \quad 0m/s \quad 55m/s]^T$, and covariance matrix $\Sigma_0 = [250000\mathrm{I}m^2 \quad 0; \quad 0 \quad \mathrm{I}m^2/s^2]$ (i.e. we are pretty sure about its velocity, but not its position). Use the Kalman filter to find your estimated state trajectory. Use your error ellipse script from Problem Set 2 to plot the means and error ellipses of the position estimate over the top of the true trajectory.

Important! Make sure the Kalman filter only sees the simulated measurements but does not have access to the true state! A simple way to make sure you separate the two is to first run the full simulation and then loop through the measurements for your filter.

(d) [2 Point.] Similarly, plot the mean and error ellipses of the velocity along the trajectory of the drone. We have essentially used the Kalman filter as a very fancy numerical *differentiator* to find velocity from GPS.

2. [5 Points.] **Kalman Filtering with Inertial Systems:** Consider the same model of the drone, but we only have a velocity sensor with the measurement equation

$$Y_t = S_t + V_t.$$

The parameters are the same, and the initial state estimate is $\mu_0 = [1000m \quad 0m \quad 0m/s \quad 50m/s]^T$ with covariance $\Sigma_0 = [\mathrm{I}m^2 \quad 0; \quad 0 \quad \mathrm{I}m^2/s^2]$ (i.e. we are pretty sure about the initial state). Simulate the trajectories and run the Kalman filter to estimate the position and velocity in this case. Plot them in the same way as before. Here, we have essentially used the Kalman filter as a very fancy numerical *integrator* to get position from velocity measurements.[1]

[1] This is called "dead reckoning," and as you can see, it cannot provide good position estimates for very long.

3. [30 points] **Nonlinear Filters:** Consider the discrete time nonholonomic mobile robot model we saw in class

$$P^x_{t+1} = P^x_t + \delta t\, s_t \cos\Theta_t + W^x_t$$
$$P^y_{t+1} = P^y_t + \delta t\, s_t \sin\Theta_t + W^y_t$$
$$\Theta_{t+1} = \Theta_t + \delta t\, \phi_t + W^\theta_t$$

where $P_t = [P^x_t \quad P^y_t]^T$ is the robot position relative to a base station, Θ_t is its heading angle, s_t its speed input, ϕ_t its rotation rate input, and w_t is the process noise. Suppose you only get range measurements from two base stations of the form $y_t = \|p_t - p_j\| + v_t$ for station j. Assume that the base station locations are known at $(30m, -5m)$ and $(-10m, 20m)$.

Let $Q = (0.1\delta t)\mathrm{I}_3$, and $R = 0.1\mathrm{I}_2$, and let the inputs be $s_t = 1m/s$, $\phi_t = (\sin t) rad/s$. Let the initial robot state estimate be normally distributed with mean $\mu_0 = [0m, 0m, 0rad]^T$ and covariance $\Sigma_0 = .01\mathrm{I}_3$. Use $\delta t = 0.5s$ and run for 100 timesteps (i.e., $50s$).

(a) [8 points] Implement an EKF to estimate the robot state. Check the observability matrix for the linearized C_t, A_t Jacobians at each time step. Is the linearized system observable always, sometimes, never?

(b) [8 points] Implement a UKF to estimate the robot state.

(c) [8 points] Implement a particle filter with 1000 particles to estimate the robot state.

(d) [3 points] For all the filters, plot the ground-truth trajectories and their estimated trajectories (you may find the provided simulation code useful; see `q3_simulator_script.py`). Use your code from Problem Set 2 to also plot the covariance ellipses for the position component of the state. For the particle filter, be sure to plot the samples and the sample mean. Do the filters track well? Should they be able to (why or why not)?

(e) [3 points] Note the average computation time for one loop of each of the three filters. How do they compare?

You can time the filter over the whole trajectory and divide by the trajectory length. However, you may find it more insightful to save the time of each loop into an array and then average the array. The latter approach will let you make a box-plot of the filter timing.