ME8135 State Estimation for Robotics and Computer Vision HW3

Matthew Lisondra Sajad Saeedi

June 9, 2023

Question 1

Use pyGame, or any other similar libraries, to simulate a simplified 2D robot and perform state estimation using a Particle Filter. Motion Model:

$$\dot{x} = \frac{r}{2}(u_r + u_l) + \omega_x \qquad \dot{y} = \frac{r}{2}(u_r + u_l) + \omega_y$$

$r = 0.1$ m is the radius of the wheel, u_r and u_l are control signals applied to the right and left wheels. $\omega_x = N(0, 0.1)$ and $\omega_y = N(0, 0.15)$. Simulate the system such that the robot is driven 1 m to the right. Assume the speed of each wheel is fixed and is 0.1 m/s. Use these initial values

$$x_0 = 0, y_0 = 0, P_0 = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix} \text{(initial covariance matrix)}$$

and assume the motion model is computed 8 times a second. Assume every second a measurement is given:

$$z = \begin{pmatrix} 1 & 0 \\ 0 & 2 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} r_x & 0 \\ 0 & r_y \end{pmatrix}$$

where $r_x = N(0, 0.05)$ and $r_y = N(0, 0.075)$.

SOLUTION: we need to discretize the system by first doing,

$$\dot{x} = \frac{r}{2}(u_r + u_l) + \omega_x \qquad \dot{y} = \frac{r}{2}(u_r + u_l) + \omega_y$$

$$\frac{x_k - x_{k-1}}{T} = \frac{r}{2}(u_r + u_l) + \omega_x \qquad \frac{y_k - y_{k-1}}{T} = \frac{r}{2}(u_r + u_l) + \omega_y$$

$$x_k - x_{k-1} = \frac{Tr}{2}(u_r + u_l) + T\omega_x \qquad y_k - y_{k-1} = \frac{Tr}{2}(u_r + u_l) + T\omega_y$$

$$x_k = x_{k-1} + \frac{Tr}{2}(u_r + u_l) + T\omega_x \qquad y_k = y_{k-1} + \frac{Tr}{2}(u_r + u_l) + T\omega_y$$

$$\Rightarrow \begin{pmatrix} x_k \\ y_k \end{pmatrix} = \begin{pmatrix} x_{k-1} \\ y_{k-1} \end{pmatrix} + \begin{pmatrix} \frac{Tr}{2} & \frac{Tr}{2} \\ \frac{Tr}{2} & \frac{Tr}{2} \end{pmatrix} \begin{pmatrix} u_r \\ u_l \end{pmatrix} + \begin{pmatrix} T\omega_x \\ T\omega_y \end{pmatrix}$$

and so the last line is our motion model. We set $T = 1/8$ s and hardcode $u_r = 1$ and $u_l = 0.1$ (we set our controls like this becuase we mainly want our robot to go right).

Below is the Particle filter algorithm. Using our motion model above, we see what we must define matrices A_t, B_t and R_t.

1: **Algorithm Particle_filter**($\mathcal{X}_{t-1}, u_t, z_t$):
2: $\bar{\mathcal{X}}_t = \mathcal{X}_t = \emptyset$
3: for $m = 1$ to M do
4: sample $x_t^{[m]} \sim p(x_t \mid u_t, x_{t-1}^{[m]})$
5: $w_t^{[m]} = p(z_t \mid x_t^{[m]})$
6: $\bar{\mathcal{X}}_t = \bar{\mathcal{X}}_t + \langle x_t^{[m]}, w_t^{[m]} \rangle$
7: endfor
8: for $m = 1$ to M do
9: draw i with probability $\propto w_t^{[i]}$
10: add $x_t^{[i]}$ to $\mathcal{X}_t$
11: endfor
12: return $\mathcal{X}_t$

The Particle filter algorithm [1]-[2].

We define matrices A_t, B_t and R_t as follows:

$$A_t = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \qquad B_t = \begin{pmatrix} \frac{Tr}{2} & \frac{Tr}{2} \\ \frac{Tr}{2} & \frac{Tr}{2} \end{pmatrix} \qquad R_t = \begin{pmatrix} T * 0.1 & 0 \\ 0 & T * 0.15 \end{pmatrix}$$

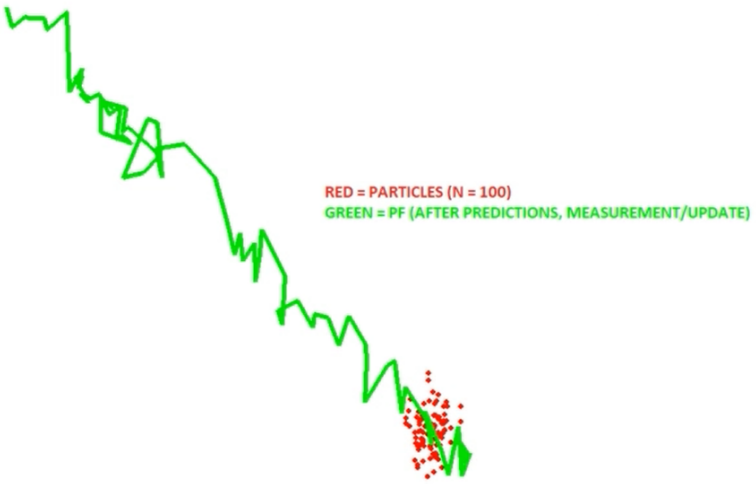
The R_t matrix is the covariance of the Gaussian random vector that models the randomness in the state transition given by the last term (the noise) in the motion model above $(T\omega_x, T\omega_y)$.

We now see from the measurement model we must define matrices C_t and Q_t as follows:

$$C_t = \begin{pmatrix} 1 & 0 \\ 0 & 2 \end{pmatrix} \qquad Q_t = \begin{pmatrix} 0.05 & 0 \\ 0 & 0.075 \end{pmatrix}$$

The Q_t matrix is the covariance of the Gaussian random vector that models the randomness in the measurement given by the last term (the noise) in the measurement model above (r_x, r_y).

Below is for $N = 100$ particles and the Particle Filter motion (in green) after eight predictions (propagating the noise) and one update per second (using weighting via importance sampling).



Robot is moving to the right (as in right side of screen) ($N = 100$ particles and Particle Filter trajectories shown) where robot moves according to most probable in non-parametric distribution [1].

The plot and animation is in a separate file. Just analyzing the trajectory, we see that the Particle Filter trajectory is jumpy, even in aiming for motion along a straight-line. A measurement does well to smoothen the path where in the end it converges and smoothens the path at end.

Question 2

Repeat the previous assignment, this time with a classic motion model and range observations made from a landmark located at $M = [10, 10]$. L is the distance between the wheel, known as wheelbase, and is 0.3 m.

$$\dot{x} = \frac{r}{2}(u_r + u_l)\cos(\theta) + \omega_x \qquad \dot{y} = \frac{r}{2}(u_r + u_l)sin(\theta) + \omega_y \qquad \dot{\theta} = \frac{r}{L}(u_r - u_l)$$

Assume

$$u_\omega = \frac{1}{2}(u_r + u_l), \qquad u_\psi = (u_r - u_l)$$

Then the equations become:

$$\dot{x} = ru_\omega\cos(\theta) + \omega_\omega \qquad \dot{y} = ru_\omega sin(\theta) + \omega_\omega \qquad \dot{\theta} = \frac{r}{L}u_\psi + \omega_\psi$$

$\omega_\psi = N(0, 0.01)$ and $\omega_\omega = N(0, 0.1)$. Program the robot such that it loops around point M.

a) Compute the Particle Filter with the linear measurement model in the previous assignment.

SOLUTION: We need to discretize the system by first doing,

$$\begin{aligned} \dot{x} &= ru_\omega\cos(\theta) + \omega_\omega \\ \frac{x_k - x_{k-1}}{T} &= ru_\omega\cos(\theta) + \omega_\omega \\ x_k - x_{k-1} &= Tru_\omega\cos(\theta) + T\omega_\omega \\ x_k &= x_{k-1} + Tru_\omega\cos(\theta) + T\omega_\omega \end{aligned} \qquad \begin{aligned} \dot{y} &= ru_\omega sin(\theta) + \omega_\omega \\ \frac{y_k - y_{k-1}}{T} &= ru_\omega sin(\theta) + \omega_\omega \\ y_k - y_{k-1} &= Tru_\omega sin(\theta) + T\omega_\omega \\ y_k &= y_{k-1} + Tru_\omega sin(\theta) + T\omega_\omega \end{aligned}$$

$$\begin{aligned} \dot{\theta} &= \frac{r}{L}u_\psi + \omega_\psi \\ \frac{\theta_k - \theta_{k-1}}{T} &= \frac{r}{L}u_\psi + \omega_\psi \\ \theta_k - \theta_{k-1} &= \frac{Tr}{L}u_\psi + T\omega_\psi \\ \theta_k &= \theta_{k-1} + \frac{Tr}{L}u_\psi + T\omega_\psi \end{aligned}$$

$$\Rightarrow \begin{pmatrix} x_k \\ y_k \\ \theta_k \end{pmatrix} = \begin{pmatrix} x_{k-1} \\ y_{k-1} \\ \theta_{k-1} \end{pmatrix} + \begin{pmatrix} Trcos(\theta) & 0 \\ Trsin(\theta) & 0 \\ 0 & Tr/L \end{pmatrix} \begin{pmatrix} u_\omega \\ u_\psi \end{pmatrix} + \begin{pmatrix} T\omega_\omega \\ T\omega_\omega \\ T\omega_\psi \end{pmatrix}$$

and so the last line is our motion model. Again, we set $T = 1/8$, but this time control u_r and u_l. If the robot distance from landmark $M = [10, 10]$ is less than a fixed distance $d = 10$ m here, then we

set $u_r = 1, u_l = 0.1$ (more favouring of going to the right). If it is greater than some fixed distance $d + 1 = 11$ m here, then we set $u_l = 1, u_r = 0.1$ (more favouring of going to the left). Note: the robot is moving counter-clockwise, hence why we favour the controls as such.

We define matrices G_t and R_t as follows:

$$G_t = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \qquad R_t = \begin{pmatrix} 0.01 & 0 & 0 \\ 0 & 0.01 & 0 \\ 0 & 0 & 0.1 \end{pmatrix}$$

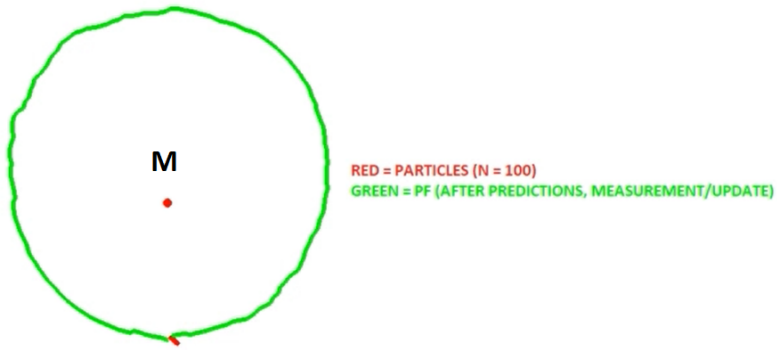
Again, the R_t matrix is the covariance of the Gaussian random vector that models the randomness in the state transition given by the last term (the noise) in the motion model above $(T\omega_\omega, T\omega_\omega, T\omega_\psi)$.

We now see from the measurement model we must define matrices H_t and Q_t as follows:

$$H_t = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{pmatrix} \qquad Q_t = \begin{pmatrix} 0.05 & 0 & 0 \\ 0 & 0.075 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

The Q_t matrix is the covariance of the Gaussian random vector that models the randomness in the measurement given by the last term (the noise) in the measurement model above $(r_x, r_y, 0)$.

Below is for $N = 100$ particles and the Particle Filter motion (in green) after eight predictions (propagating the noise) and one update per second (using weighting via importance sampling).



Robot is moving around landmark M ($N = 100$ particles and Particle Filter trajectories shown) where robot moves according to most probable in non-parametric distribution [1].

The plot and animation is in a separate file. Just analyzing the trajectory, we see that the Particle Filter trajectory is very smooth. We are zoomed out quite a bit and so the $N = 100$ particles may seem too small (shown as small red points bunched up at bottom of figure). The trajectory is nicely circular, as we hoped.

b) Compute with the range/bearing measurements of point M. Assume range noise is $N(0, 0.1)$ and bearing noise is $N(0, 0.01)$. Range is in meters, and bearing is in radians. Visualize the measurements as well.

SOLUTION: We replace our measurement model from above with a range/bearing measurement model around $M = [10, 10]$. In order to do this we must write in polar coordinates:

$$\begin{pmatrix} x_k \\ y_k \\ \theta_k \end{pmatrix} \rightarrow \begin{pmatrix} \rho_k \\ \theta_k{}' \end{pmatrix} \qquad \rho_k = \sqrt{(x_k - 10)^2 + (y_k - 10)^2}$$
$$\theta_k' = arctan\left(\frac{y - 10}{x - 10}\right)$$

where θ'_k refers to the angle from the three-state (x, y, θ) from part (a). The polar representation is assuming that the coordinate axes are not on landmark M and that we form a right-angle triangle whose hypotenuse is ρ with legs of length $x_k - 10$ and $y_k - 10$.

We also must replace H_t and Q_t as follows:

$$H_t = \begin{pmatrix} \frac{\partial \rho_k}{\partial x_k} & \frac{\partial \rho_k}{\partial y_k} & \frac{\partial \rho_k}{\partial \theta_k} \\ \frac{\partial \theta'_k}{\partial x_k} & \frac{\partial \theta'_k}{\partial y_k} & \frac{\partial \theta'_k}{\partial \theta_k} \end{pmatrix}$$

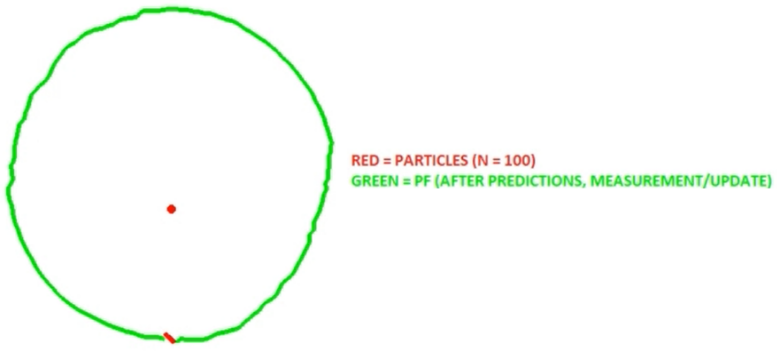
$$= \begin{pmatrix} \frac{x_k - 10}{\sqrt{(x_k - 10)^2 + (y_k - 10)^2}} & \frac{y_k - 10}{\sqrt{(x_k - 10)^2 + (y_k - 10)^2}} & 0 \\ \frac{10 - y_k}{\sqrt{(x_k - 10)^2 + (y_k - 10)^2}} & \frac{10 - x_k}{\sqrt{(x_k - 10)^2 + (y_k - 10)^2}} & -1 \end{pmatrix}$$

$$Q_t = \begin{pmatrix} 0.1 & 0 \\ 0 & 0.01 \end{pmatrix}$$

The Q_t matrix is the covariance of the Gaussian random vector that models the randomness in the measurement given by the last term (the range noise and bearing noise) in the measurement model above (rn, rbn).

Below is for $N = 100$ particles and the Particle Filter motion (in green) after eight predictions (propagating the noise) and one update per second (using weighting via importance sampling).

The plot and animation is in a separate file. Just analyzing the trajectory, we see that the Particle Filter trajectory is smooth, but more jumpier with these range/bearing measurements of point M than (a)'s basis of measurements. Again, we are zoomed out quite a bit and so the $N = 100$ particles may seem too small (shown as small red points bunched up at bottom of figure). The trajectory is circular, as we hoped, with but (a)'s circular trajectory is more well defined.



Robot is moving around landmark M ($N = 100$ particles and Particle Filter trajectories shown) where robot moves according to most probable in non-parametric distribution [1].

References

[1] Thrun, Sebastian. "Probabilistic robotics." Communications of the ACM 45.3 (2002): 52-57.

[2] Barfoot, Timothy D. State estimation for robotics. Cambridge University Press, 2017.