Jae Seok Oh

jaeseoko

3 P4: Problems [Due Nov. 24, 2020]

Exercise 1. In this final part of the project, you will design and implement an **Extended Kalman Filter Simultaneous Localization and Mapping (EKF SLAM)**. In previous parts, we assume that all state measurements are available. However, it is not always true in the real world. Localization information from GPS could be missing or inaccurate in the tunnel, or closed to tall infrastructures. In this case, we do not have direct access to the global position X, Y and heading ψ and have to estimates them from $\dot{x}$, $\dot{y}$, $\dot{\psi}$ on the vehicle frame and range and bearing measurements of map features.

Consider the discrete-time dynamics of the system:

$$\begin{aligned} X_{t+1} &= X_t + \delta t \dot{X}_t + \omega_t^x \\ Y_{t+1} &= Y_t + \delta t \dot{Y}_t + \omega_t^y \\ \psi_{t+1} &= \psi_t + \delta t \dot{\psi}_t + \omega_t^\psi \end{aligned} \tag{1}$$

Substitute $\dot{X}_t = \dot{x}_t \cos\psi_t - \dot{y}_t \sin\psi_t$ and $\dot{Y}_t = \dot{x}_t \sin\psi_t + \dot{y}_t \cos\psi_t$ in to (1),

$$\begin{aligned} X_{t+1} &= X_t + \delta t(\dot{x}_t \cos\psi_t - \dot{y}_t \sin\psi_t) + \omega_t^x \\ Y_{t+1} &= Y_t + \delta t(\dot{x}_t \sin\psi_t + \dot{y}_t \cos\psi_t) + \omega_t^y \\ \psi_{t+1} &= \psi_t + \delta t \dot{\psi}_t + \omega_t^\psi \end{aligned} \tag{2}$$

The input u_t is $[\dot{x}_t \quad \dot{y}_t \quad \dot{\psi}_t]^T$. Let $p_t = [X_t \quad Y_t]^T$. Suppose you have n map features at global position $m^j = [m_x^j \quad m_y^j]^T$ for $j = 1, ..., n$. The ground truth of these map feature positions are static but unknown, meaning they will not move but we do not know where they are exactly. However, the vehicle has both range and bearing measurements relative to these features. The range measurement is defined as the distance to each feature with the measurement equations $y_{t,distance}^j = \|m^j - p_t\| + v_{t,distance}^j$ for $j = 1, ..., n$. The bearing measure is defined as the angle between the vehicle's heading (yaw angle) and ray from the vehicle to the feature with the measurement equations $y_{t,bearing}^j = atan2(m_y^j - Y_t, m_x^j - X_t) - \psi_t + v_{t,bearing}^j$ for $j = 1, ..., n$.

Let the state vector be

$$\boldsymbol{x_t} = \begin{bmatrix} X_t \\ Y_t \\ \psi_t \\ m_x^1 \\ m_y^1 \\ m_x^2 \\ m_y^2 \\ \vdots \\ m_x^n \\ m_y^n \end{bmatrix} \tag{3}$$

The measurement system be

$$\boldsymbol{y_t} = \begin{bmatrix} \|m^1 - p_t\| + v^1_{t,distance} \\ \vdots \\ \|m^n - p_t\| + v^n_{t,distance} \\ atan2(m^1_y - Y_t, m^1_x - X_t) - \psi_t + v^1_{t,bearing} \\ \vdots \\ atan2(m^n_y - Y_t, m^n_x - X_t) - \psi_t + v^n_{t,bearing} \end{bmatrix} \tag{4}$$

Derive all the necessary equations and matrices for EKF SLAM to estimate the vehicles state X, Y, ψ and feature positions $m^j = [m^j_x \quad m^j_y]^T$ simultaneously. [Hints: write out the complete dynamical system with the state in (3) and then follow the standard procedure of deriving EKF. Think what is the dynamic for static landmarks?]

$$f(x_{t-1}, u_t)$$

$$= \begin{bmatrix} x_t + \delta t \cos\psi_t \dot{x}_t - \delta t \sin\psi_t \dot{y}_t \\ y_t + \delta t \sin\psi_t \dot{x}_t + \delta t \cos\psi_t \dot{y}_t \\ \psi_t + \delta t \dot{\psi}_t \\ m_x^1 \\ m_y^1 \\ \vdots \\ m_x^n \\ m_y^n \end{bmatrix}$$

$$F_t = \left.\frac{\partial f}{\partial x}\right|_{\hat{x}_{t-1|t-1}, u_t}$$

$$= \left[\begin{array}{ccc|c} 1 & 0 & -\delta_t \sin\psi_t \dot{x}_t - \delta_t \cos\psi_t \dot{y}_t & \\ 0 & 1 & \delta_t \cos\psi_t \dot{x}_t - \delta_t \sin\psi_t \dot{y}_t & O_{3\times 2n} \\ 0 & 0 & 1 & \\ \hline & O_{2n\times 3} & & Id_{2n} \end{array}\right]$$

$$h(x_t) = \begin{bmatrix} \|m^1 - p_t\| \\ \vdots \\ \|m^n - p_t\| \\ \operatorname{atan2}(m_y^1 - y_t, m_x^1 - x_t) - \psi_t \\ \vdots \\ \operatorname{atan2}(m_y^n - y_t, m_x^n - x_t) - \psi_t \end{bmatrix}$$

where $\|m - p_t\| = \sqrt{(m_x - x_t)^2 + (m_y - y_t)^2}$

$$H_t = \left.\frac{\partial h}{\partial x}\right|_{\hat{x}_{t|t-1}}$$

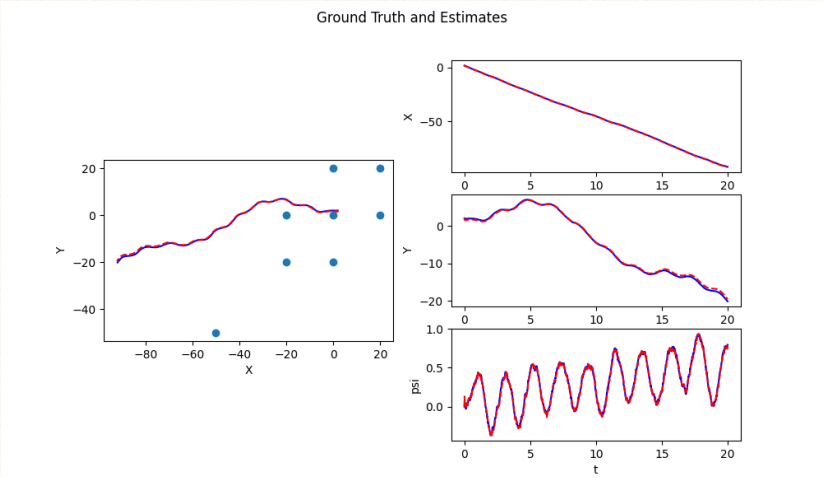
$$= \begin{bmatrix}
\frac{x_t - m_x^1}{\|m^1 - p_t\|} & \frac{y_t - m_y^1}{\|m^1 - p_t\|} & 0 & \frac{m_x^1 - x_t}{\|m^1 - p_t\|} & \frac{m_y^1 - y_t}{\|m^1 - p_t\|} & 0 & \cdots & 0 \\
\frac{x_t - m_x^2}{\|m^2 - p_t\|} & \frac{y_t - m_y^2}{\|m^2 - p_t\|} & 0 & 0 & 0 & \frac{m_x^2 - x_t}{\|m^2 - p_t\|} & \frac{m_y^2 - y_t}{\|m^2 - p_t\|} & 0 \cdots 0 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\frac{x_t - m_x^n}{\|m^n - p_t\|} & \frac{y_t - m_y^n}{\|m^n - p_t\|} & 0 & 0 & \cdots & 0 \cdots & \frac{m_x^n - x_t}{\|m^n - p_t\|} & \frac{m_y^n - y_t}{\|m^n - p_t\|} \\
\hline
\frac{-(y_t - m_y^1)}{\|m^1 - p_t\|^2} & \frac{-(m_x^1 - x_t)}{\|m^1 - p_t\|^2} & -1 & \frac{-(m_y^1 - y_t)}{\|m^1 - p_t\|^2} & \frac{-(x_t - m_x^1)}{\|m^1 - p_t\|^2} & 0 & \cdots & 0 \\
\frac{-(y_t - m_y^2)}{\|m^2 - p_t\|^2} & \frac{-(m_x^2 - x_t)}{\|m^2 - p_t\|^2} & -1 & 0 & 0 & \frac{-(m_y^2 - y_t)}{\|m^2 - p_t\|^2} & \frac{-(x_t - m_x^2)}{\|m^2 - p_t\|^2} & \cdots 0 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\frac{-(y_t - m_y^n)}{\|m^n - p_t\|^2} & \frac{-(m_x^n - x_t)}{\|m^n - p_t\|^2} & -1 & 0 & \cdots & 0 \cdots & \frac{-(m_y^n - y_t)}{\|m^n - p_t\|^2} & \frac{-(x_t - m_x^n)}{\|m^n - p_t\|^2}
\end{bmatrix}$$

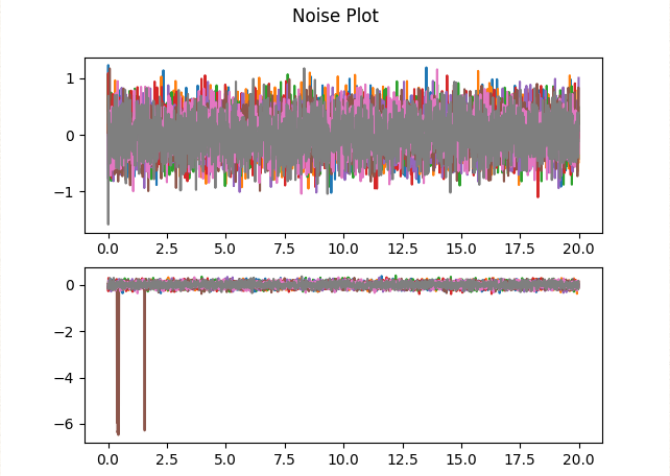
Exercise 2. For this exercise, you will implement EKF SLAM in the `ekf_slam.py` file. You can use the same controller from your previous parts or write a new one. Before integrating this module with Webots, you can run the script by `$ python ekf_slam.py` to test your implementation. You should get some reasonable plots. Feel free to write your own unit-testing scripts. You should not use any existing Python package that implements EKF. [Hints: remember to wrap heading angles to $[-\pi, \pi]$]

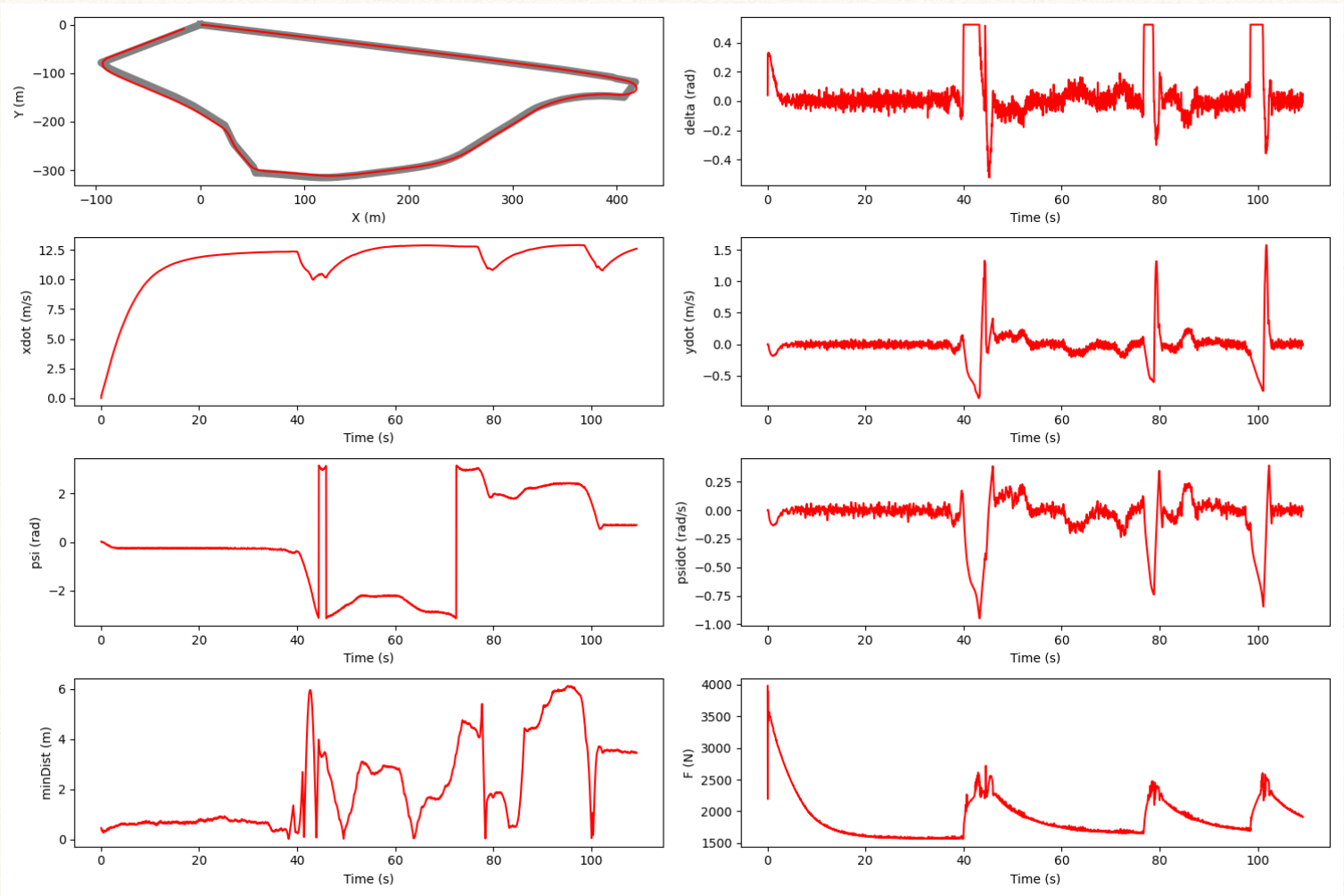
Check the performance of your controller by running the Webots simulation. You can press the play button in the top menu to start the simulation in real-time, the fast-forward button to run the simulation as quickly as possible, and the triple fast-forward to run the simulation without rendering (any of these options is acceptable, and the faster options may be better for quick tests). If you complete the track, the scripts will generate a performance plot via `matplotlib`. This plot contains a visualization of the car's trajectory, and also shows the variation of states with respect to time.

Submit `your_controller_ekf_slam.py`, `ekf_slam.py`, and the final completion plot as described on the title page. Your controller is **required** to achieve the following performance criteria to receive full points:

1. Time to complete the loop = 250 s
2. Maximum deviation from the reference trajectory = 10.0 m
3. Average deviation from the reference trajectory = 5 m


Ground Truth and Estimates
X
Y
psi
t


Noise Plot


Y (m)
X (m)
delta (rad)
Time (s)
xdot (m/s)
Time (s)
ydot (m/s)
Time (s)
psi (rad)
Time (s)
psidot (rad/s)
Time (s)
minDist (m)
Time (s)
F (N)
Time (s)