# EECE 5550 Mobile Robotics - Section 2
# Lab #2

Due: Oct 31, 2025 by midnight

## Problem 1: Extended Kalman Filter

In this exercise you will apply the extended Kalman filter to solve a basic 2D robot localization problem. To that end, we will consider a ground robot that is navigating in a 2D environment containing two landmarks at *known positions* $\mathbf{l_i} = [l_x^i, l_y^i] \in \mathbb{R}^2$ for $i \in \{1, 2\}$.

(a) We will model our robot as a simple point moving in the plane (i.e. ignoring its orientation), and assume that it is possible to directly control its velocity. Denoting its position and velocity at time $t$ by $p(t) = (p_x(t), p_y(t)) \in \mathbb{R}^2$ and $v(t) = (v_x(t), v_y(t)) \in \mathbb{R}^2$ (respectively), write down the discrete-time state transition model that predicts the robot's position $p(t + \Delta t)$ at time $t + \Delta t$ given its position $p(t)$ at time $t$, assuming constant velocity and additive mean-zero Gaussian noise with covariance $R \in \mathbb{S}_+^2$ on the final position.

(b) At each timestep, we assume that the robot is able to collect a noisy measurement of the *range* to each of the two landmarks in the environment, subject to additive mean-zero Gaussian noise with covariance $Q \in \mathbb{S}_+^2$. Write down the corresponding measurement model for the resulting measurements.

(c) As we saw in class, implementing an extended Kalman filter requires linearizing any nonlinear state transition or measurement models about the current state estimate, and using the resulting linear approximations in the standard Kalman filter equations. Derive the Jacobians for the state transition and measurement models in parts (a) and (b).

(d) Using your results from parts (a)–(c), derive the corresponding state propagation and measurement update equations for an extended Kalman filter to solve this localization problem.

(e) Using the EKF that you designed in parts (a)–(d), write a script that applies it to solve the following simulated localization scenario:

- The landmarks are located at $l_1 = (5, 5)$ and $l_2 = (-5, 5)$.
- The covariances for the robot state transition and measurement models are given by $R = .1I_2$, $Q = .5I_2$ for all $t \in [0, 40]$.
- The robot's initial belief over its position is $p(0) \sim \mathcal{N}(0, I_2)$.
- The timestep interval is given by $\Delta t = .5$, and the sequence of velocity commands is given by:
  * $v = (1, 0)$ for all $0 \leq t \leq 10$,
  * $v = (0, -1)$ for all $10 < t \leq 20$,
  * $v = (-1, 0)$ for all $20 < t \leq 30$,

* $v = (0, 1)$ for all $30 < t \leq 40$.

Generate a plot showing:

- The *true* locations of the landmarks and the sequence of *true* robot positions.
- The means and $3\sigma$ confidence bounds for the sequence of *estimated* beliefs over the robot's position computed using your Kalman filter.

## Problem 2: Scan matching using Iterative Closest Point

We saw in class that laser scan matching can be used to recover high-precision estimates of the relative transformation between two sensor frames. In practice, this is often used to produce estimates of robot motion (i.e. odometry) that are *far* more accurate than what can be achieved using e.g. wheel odometry or IMU integration.

In this exercise, you will implement the Iterative Closest Point (ICP) algorithm, and use it to estimate the rigid transformation that optimally aligns two 3D pointclouds. Recall that given two pointclouds $X, Y \subset \mathbb{R}^d$ and an initial guess $(t_0, R_0) \in \mathrm{SE}(d)$ for the optimal rigid registration $y = Rx + t$ aligning $X$ to $Y$, ICP (Algorithm 1) proceeds by alternating between the following two computations:

- Estimating a set of correspondences $C = \{(i_k, j_k)\}_{k=1}^K$ between points in the two pointclouds, given an estimate for the optimal registration $(\hat{t}, \hat{R}) \in \mathrm{SE}(d)$ (lines 5–10),
- Computing a rigid registration $(\hat{t}, \hat{R}) \in \mathrm{SE}(d)$ that optimally aligns the set $\{(x_{i_k}, y_{j_k})\}_{k=1}^K$ of corresponding points in the least-squares sense:

$$(\hat{t}, \hat{R}) \in \underset{(t,R) \in \mathrm{SE}(d)}{\operatorname{argmin}} \sum_{k=1}^{K} \| y_{j_k} - (R x_{i_k} + t) \|_2^2. \tag{1}$$

  Recall that Horn's method (Algorithm 2) provides an efficient algorithm for computing the optimal registration $(\hat{t}, \hat{R}) \in \mathrm{SE}(d)$ in (1).

(a) Implement a function `EstimateCorrespondences`$(X, Y, t, R, d_{\max})$ that constructs the list $C = \{(i_k, j_k)\}_{k=1}^K$ of estimated point correspondences, using the procedure of lines 5–10 in Algorithm 1. Your function should take as input the pointclouds $X, Y \subset \mathbb{R}^d$, the estimated rigid transformation $(t, R) \in \mathrm{SE}(d)$ aligning $X$ to $Y$, and the maximum admissible distance $d_{\max}$ for associating two points, and return a list $C = \{(i_k, j_k)\}_{k=1}^K$ of estimated point correspondences.

(b) Implement the function `ComputeOptimalRigidRegistration`$(X, Y, C)$ shown in Algorithm 2.

(c) With the aid of your results from parts (a) and (b), write a function that implements the ICP algorithm (Algorithm 1). Your implementation should accept as input the two pointclouds $X$ and $Y$, an initial guess $(t_0, R_0) \in \mathrm{SE}(d)$ for the rigid transformation aligning $X$ to $Y$, the maximum admissible distance $d_{\max}$ for associating points, *and* the number `num_ICP_iters` of ICP iterations (lines 3–12 of Algorithm 1) to perform.

(d) Now you will apply your ICP implementation to register two 3D pointclouds. Download the files `pclX.txt` and `pclY.txt` containing the pointclouds from the course Canvas website, and run your ICP implementation with the following parameters:

**Algorithm 1** Iterative closest point

**Input:** Pointclouds $X = \{x_i\}_{i=1}^{n_X} \subset \mathbb{R}^d$ and $Y = \{y_i\}_{i=1}^{n_Y} \subset \mathbb{R}^d$, initial guess $(t_0, R_0) \in \mathrm{SE}(d)$ for a rigid registration aligning pointcloud $X$ to pointcloud $Y$, maximum admissible distance $d_{\max} > 0$ for associating points.

**Output:** An estimated set $C = \{(i_k, j_k)\}_{k=1}^{K}$ of correspondences between points in $X$ and $Y$, and the rigid registration $T = (\hat{t}, \hat{R}) \in \mathrm{SE}(d)$ that optimally aligns corresponding points of $X$ and $Y$ in the least-squares sense (1).

**function** ICP$(X, Y, t_0, R_0, d_{\max})$
    Initialization: $\hat{t} \leftarrow t_0$, $\hat{R} \leftarrow R_0$.
    **repeat**
        Initialize list of empty correspondences: $C \leftarrow \varnothing$, $K \leftarrow 0$.
        **for** $i = 1, \ldots, n_X$ **do**: ▷ Estimate point correspondences
            Find closest point $y_j$ in $Y$ to image of $x_i$ under transformation $(\hat{t}, \hat{R})$:

$$y_j = \operatorname*{argmin}_{y \in Y} \|y - (\hat{R}x_i + \hat{t})\|_2^2. \tag{2}$$

            **if** $\|y_j - (\hat{R}x_i + \hat{t})\|_2 < d_{\max}$ **then**
                Add $x_i$ and $y_j$ to the list of point correspondences: $C \leftarrow C \cup \{(i, j)\}$.
            **end if**
        **end for**
        Compute optimal registration given the point correspondences $C$:

$$(\hat{t}, \hat{R}) \leftarrow \textsc{ComputeOptimalRigidRegistration}(X, Y, C) \tag{3}$$

    **until** convergence
    **return** $(\hat{t}, \hat{R}, C)$
**end function**

- Set the initial estimate for the optimal registration to the identity: $(t_0, R_0) = (0, I_3) \in \mathrm{SE}(3)$.
- Set $d_{\max} = .25$
- Set the number of ICP iterations to 30: `num_ICP_iters` $= 30$.

Check your results by plotting the pointcloud $Y$ and the image of $X$ under the estimated transformation $(\hat{t}, \hat{R}) \in \mathrm{SE}(3)$ on the same axes, using different colors for the two pointclouds, and compute the root-mean-squared error for the associated points under this registration:

$$\mathrm{RMSE} = \sqrt{\frac{1}{K} \sum_{k=1}^{K} \|y_{j_k} - (\hat{R}x_{i_k} + \hat{t})\|_2^2}. \tag{8}$$

Then report and submit the following:

- The parameters $(\hat{t}, \hat{R}) \in \mathrm{SE}(3)$ for the estimated rigid transformation,
- The RMSE (8) for the estimated point correspondences,
- The plot showing the co-registered pointclouds,
- Your code.

**Algorithm 2** Optimally aligning pointclouds with *known* point correspondences

**Input:** Pointclouds $X = \{x_i\}_{i=1}^{n_X} \subset \mathbb{R}^d$ and $Y = \{y_j\}_{j=1}^{n_Y} \subset \mathbb{R}^d$, list of point correspondences $C = \{(i_k, j_k)\}_{k=1}^{K}$.

**Output:** Rigid transformation $T = (\hat{t}, \hat{R}) \in \mathrm{SE}(d)$ that optimally aligns corresponding points of $X$ and $Y$ in the least-squares sense (1).

**function** ComputeOptimalRigidRegistration$(X, Y, C)$

Calculate pointcloud centroids:

$$\bar{x} \triangleq \frac{1}{K}\sum_{k=1}^{K} x_{i_k}, \qquad \bar{y} \triangleq \frac{1}{K}\sum_{k=1}^{K} y_{j_k}. \tag{4}$$

Calculate deviations of each point from the centroid of its pointcloud:

$$x'_{i_k} \triangleq x_{i_k} - \bar{x}, \qquad y'_{j_k} \triangleq y_{j_k} - \bar{y}. \tag{5}$$

Compute cross-covariance matrix $W$:

$$W \triangleq \frac{1}{K}\sum_{k=1}^{K} y'_{j_k}(x'_{i_k})^{\mathsf{T}}. \tag{6}$$

Compute singular value decomposition: $W = U\Sigma V^{\mathsf{T}}$.

Construct optimal rotation:

$$\hat{R} \triangleq U\,\mathrm{Diag}(1, \ldots, 1, \det(UV)))V^{\mathsf{T}}. \tag{7}$$

Recover optimal translation: $\hat{t} \triangleq \bar{y} - \hat{R}\bar{x}$.

**return** $(\hat{t}, \hat{R})$.

**end function**

## Problem 3: State estimation by particle filtering on a Lie group

We saw in class that the Particle Filter provides a simple yet highly versatile algorithm for performing recursive Bayesian estimation, and is especially well-suited to state estimation problems with nonlinear state transition or measurement functions or non-Gaussian models of uncertainty.

In this exercise, you will apply particle filtering to perform state estimation over a Lie group: specifically, you will design and implement a particle filter to track the pose of a differential-drive ground robot.

(a) The state propagation step of the particle filter (Algorithm 3) requires a procedure for sampling from the motion model $p(x_{t+1}|x_t, u_t)$ (cf. line 4). Recall from our earlier discussion that differential drive robots are actuated by controlling the velocities $(\dot{\varphi}_l, \dot{\varphi}_r)$ of their left and right wheel speeds.

Suppose that we can only control the wheel speeds of our robot *imprecisely*; that is, the *true* left and right wheel speeds $(\tilde{\varphi}_l, \tilde{\varphi}_r)$ are related to the *commanded* wheel speeds $u \triangleq (\dot{\varphi}_l, \dot{\varphi}_r)$ according to:

$$\begin{aligned} \tilde{\varphi}_l &= \dot{\varphi}_l + \epsilon_l, \quad \epsilon_l \sim \mathcal{N}(0, \sigma_l^2), \\ \tilde{\varphi}_r &= \dot{\varphi}_r + \epsilon_r, \quad \epsilon_r \sim \mathcal{N}(0, \sigma_r^2). \end{aligned} \tag{13}$$

Using (13), derive a generative description for (i.e., a list of steps to *draw samples from*) the motion model $p(x_{t_2}|x_{t_1}, \dot{\varphi}_l, \dot{\varphi}_r, r, w, \sigma_l, \sigma_r)$ that parameterizes the distribution of the

**Algorithm 3** Particle filter propagation

**Input:** Particle set $X_t = \{x_t^{[i]}\}_{i=1}^n$ sampled from belief $p(x_t)$ over initial state $x_t$, control $u_t$.

**Output:** Particle set $X_{t+1} = \{x_{t+1}^{[i]}\}_{i=1}^n$ sampled from belief $p(x_{t+1}|u_t)$ over subsequent state $x_{t+1}$ after applying control $u_t$.

**function** PARTICLEFILTERPROPAGATE($X_t, u_t$)
  Initialize empty particle set: $X_{t+1} \leftarrow \varnothing$.
  **for** $i = 1, \ldots, n$ **do**
    Draw sample $x_{t+1}^{[i]}$ from the motion model $p(x_{t+1}|x_t, u_t)$:

$$x_{t+1}^{[i]} \sim p(x_t|x_t^{[i]}, u_t). \tag{9}$$

    Add sample $x_{t+1}^{[i]}$ to the output particle set: $X_{t+1} \leftarrow X_{t+1} \cup \{x_{t+1}^{[i]}\}$.
  **end for**
  **return** $X_{t+1}$.
**end function**

**Algorithm 4** Particle filter update

**Input:** Particle set $X_t = \{x_t^{[i]}\}_{i=1}^n$ sampled from prior belief $p(x_t)$ over $x_t$, measurement $z_t$.

**Output:** Particle set $\bar{X}_t = \{\bar{x}_t^{[i]}\}_{i=1}^n$ sampled from posterior belief $p(x_t|z_t)$ over $x_t$ after incorporating measurement $z_t$.

**function** PARTICLEFILTERUPDATE($X_t, z_t$)
  **for** $i = 1, \ldots, n$ **do** ▷ Calculate importance weights
    Calculate importance weight for $x_t^{[i]}$ using measurement likelihood function:

$$w_i \triangleq p(z_t|x_t^{[i]}). \tag{10}$$

  **end for**
  Initialize empty particle set: $\bar{X}_t \leftarrow \varnothing$.
  **for** $i = 1, \ldots, n$ **do** ▷ Importance-weighted resampling
    Sample $i$th particle $\bar{x}_t^{[i]}$ from prior particle set $X_t$ with replacement:

$$\bar{x}_t^{[i]} \sim p\left(\bar{x}_t|X_t, \{w_k\}_{k=1}^n\right), \tag{11}$$

    with sampling probabilities proportional to importance weights:

$$p\left(\bar{x}_t = x_t^{[k]} \mid X_t, \{w_k\}_{k=1}^n\right) \propto w_k. \tag{12}$$

    Add sample $\bar{x}_t^{[i]}$ to the posterior particle set: $\bar{X}_t \leftarrow \bar{X}_t \cup \{\bar{x}_t^{[i]}\}$.
  **end for**
  **return** $\bar{X}_t$.
**end function**

pose of the robot $x_{t_2} \in \mathrm{SE}(2)$ at time $t_2$ as a function of its pose $x_{t_1} \in \mathrm{SE}(2)$ at time $t_1$ given the *commanded* wheel speeds $(\dot{\varphi}_l, \dot{\varphi}_r)$, its wheel radius $r$ and track width $w$, and the variances $\sigma_l^2$ and $\sigma_r^2$ for the true wheel speeds.

Note that, *for the noiseless case*, we can describe the motion of a differential drive robot as follows. As we saw in class, we can describe *velocities* of states taking values in a Lie group $G$ using an element $\dot{\Omega} \in \mathrm{Lie}(G)$ of the Lie algebra of $G$. As you showed on the previous assignment, the trajectory starting at state $X_0 \in G$ at time $t = 0$ and moving at constant velocity $\dot{\Omega}$ is then given by:

$$\gamma(t) = X_0 \exp\left(t\dot{\Omega}\right). \tag{14}$$

For the case of a 2D differential drive robot, the relation between the wheel speeds $(\dot{\varphi}_l, \dot{\varphi}_r)$ and the Lie group element $\dot{\Omega}(\dot{\varphi}_l, \dot{\varphi}_r)$ is:

$$\begin{gathered}\dot{\Omega}\colon \mathbb{R}^2 \to \mathrm{Lie}(\mathrm{SE}(2)) \\ \dot{\Omega}(\dot{\varphi}_l, \dot{\varphi}_r) = \begin{pmatrix} 0 & -\frac{r}{w}(\dot{\varphi}_r - \dot{\varphi}_l) & \frac{r}{2}(\dot{\varphi}_r + \dot{\varphi}_l) \\ \frac{r}{w}(\dot{\varphi}_r - \dot{\varphi}_l) & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix},\end{gathered} \tag{15}$$

In brief: your goal is to *extend* the *deterministic* motion model (14)–(15) by accounting for the effect of the noise (13) in the commanded wheel speeds.

(b) Similarly, the measurement update step of the particle filter (Algorithm 4) requires a procedure for evaluating the measurement likelihood function $p(z_t|x_t)$ (cf. line 3).

Suppose that our robot is equipped with a sensor that is capable of producing a noisy measurements of its position in the plane (for example, a GPS receiver). Specifically, the measurement $z_t \in \mathbb{R}^2$ at time $t$ is related to the robot's pose $x_t \triangleq (l_t, R_t) \in \mathrm{SE}(2)$ at time $t$ according to:

$$z_t = l_t + \epsilon_p, \quad \epsilon_p \sim \mathcal{N}(0, \sigma_p^2 I_2). \tag{16}$$

Derive a closed-form expression (i.e., PDF) for the measurement likelihood function $p(z_t|x_t)$ under the measurement model (16).

(c) Write (in code) a function that implements the particle filter propagation step (Algorithm 3) for a differential-drive ground robot. Your function should accept as input the following data:

- The current time $t_1$
- The particle set $X_{t_1} = \{x_{t_1}^{[i]}\}_{i=1}^n \subset \mathrm{SE}(2)$ describing the robot's belief over its position at time $t_1$
- The commanded wheel speeds $(\dot{\varphi}_l, \dot{\varphi}_r)$
- The time $t_2$ at which to predict its next pose $x_{t_2} \in \mathrm{SE}(2)$
- The parameters $r$, $w$, $\sigma_l$, and $\sigma_r$ of the robot

and return a particle set $X_{t_2} = \{x_{t_2}^{[i]}\}_{i=1}^n \subset \mathrm{SE}(2)$ describing the robot's belief over its pose at time $t_2$, sampled according to the generative model you derived in part (a).

(d) Write (in code) a function that implements the particle filter update step (Algorithm 4). Your function should accept as input:

- The particle set $X_t = \{x_t^{[i]}\}_{i=1}^n \subset \mathrm{SE}(2)$ representing its prior belief over its pose
- A noisy position $z_t$ measurement sampled according to the generative model (16),
- The magnitude $\sigma_p$ of the measurement noise,

and return the particle set $\bar{X}_t = \{x_t^{[i]}\}_{i=1}^n \subset \mathrm{SE}(2)$ modeling the robot's *posterior* belief after incorporating the measurement $z_t$.

In the next series of experiments, we will assume the following (fixed) parameter values: $\dot{\varphi}_l = 1.5$, $\dot{\varphi}_r = 2$, $r = .25$, $w = .5$, $\sigma_l = .05$, $\sigma_r = .05$, and $\sigma_p = .10$.

(e) Using the particle filter propagation function that you implemented in part (c), generate $N = 1000$ realizations of $x_{10}$ (the pose of the robot at time $t = 10$) from $p(x_{10}|x_0, \dot{\varphi}_l, \dot{\varphi}_r, r, w, \sigma_l, \sigma_r)$, assuming that $x_0 = (0, I_2) \in \mathrm{SE}(2)$ (i.e. that the robot starts at the origin at time $t = 0$). Calculate the empirical mean and covariance of the positions of the points in this particle set, and plot the positions of the sampled particles in the plane.

(f) In this part of the exercise, you will simulate the evolution of our differential drive robot's belief over its pose while navigating using *dead reckoning*.

Starting with an initial particle set $X_0$ consisting of $N = 1000$ copies of $I = (0, I_2) \in \mathrm{SE}(2)$ (indicating absolute certainty that the robot's initial pose $x_0$ is the origin), apply your particle filter propagation function from part (c) to recursively generate sample-based approximations to the robot's belief over its pose $x_t \in \mathrm{SE}(2)$ at times $t \in \{5, 10, 15, 20\}$.

For each of these sample sets, report the empirical mean and covariance of the particles' positions. Finally, plot the positions of the particles in each of these sample sets in a *single* plot, using a different color for each sample set.

(g) Now let us consider the effect of incorporating noisy measurements $z_t$ from the model (16) on the robot's uncertainty over its position.

Staring with an initial particle set $X_0$ containing $N = 1000$ copies of $I = (0, I_2) \in \mathrm{SE}(2)$ (indicating *absolute certainty* that the robot's initial pose $x_0$ is the origin), apply your particle filter propagation and update functions from parts (c) and (d) to recursively generate sample-based approximations of the robot's *posterior* beliefs over its pose $x_t \in \mathrm{SE}(2)$ at times $t \in \{5, 10, 15, 20\}$ obtained *after* incorporating the sequence of measurements:

$$z_5 = \begin{pmatrix} 1.6561 \\ 1.2847 \end{pmatrix}, \quad z_{10} = \begin{pmatrix} 1.0505 \\ 3.1059 \end{pmatrix}, \quad z_{15} = \begin{pmatrix} -0.9875 \\ 3.2118 \end{pmatrix}, \quad z_{20} = \begin{pmatrix} -1.6450 \\ 1.1978 \end{pmatrix}. \tag{17}$$

Report the empirical mean and covariance of the particles' positions in each of the posterior sample sets, and plot the positions of all of these particles in a *single* plot, using a different color for each sample set.