ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Institute for Dynamic Systems and Control
IDSC
Institut für Dynamische Systeme und Regelungstechnik

151-0566-00 **Recursive Estimation** (Spring 2025)

Programming Exercise Topic: Recursive Filtering

Issued: April 16, 2025 Due: June 20, 2025

Extended Kalman Filter and Particle Filter for Tracking an Autonomous Vehicle

Sensor fusion is a typical application of the Kalman Filter and the Particle Filter. Typically, autonomous cars have a number of sensors on board, allowing for the overlaying of their information. The adoption of several sensors is motivated by concerns for robustness as well as accuracy. When a sensor has a flaw or receives inaccurate data, other sensors can still provide some information.

You are tasked with designing an Extended Kalman Filter (EKF) and a Particle Filter (PF) for an autonomous vehicle equipped with GPS, a compass, and a tachometer. The goal is to estimate the relative position and orientation of the vehicle with respect to the coordinate system origin, while accounting for the unavailability of GPS signals in indoor areas and the large uncertainty of compass data. Additionally, the velocity and a vehicle parameter are also considered as estimation targets. A schematic diagram of the vehicle is shown in Figure 1.

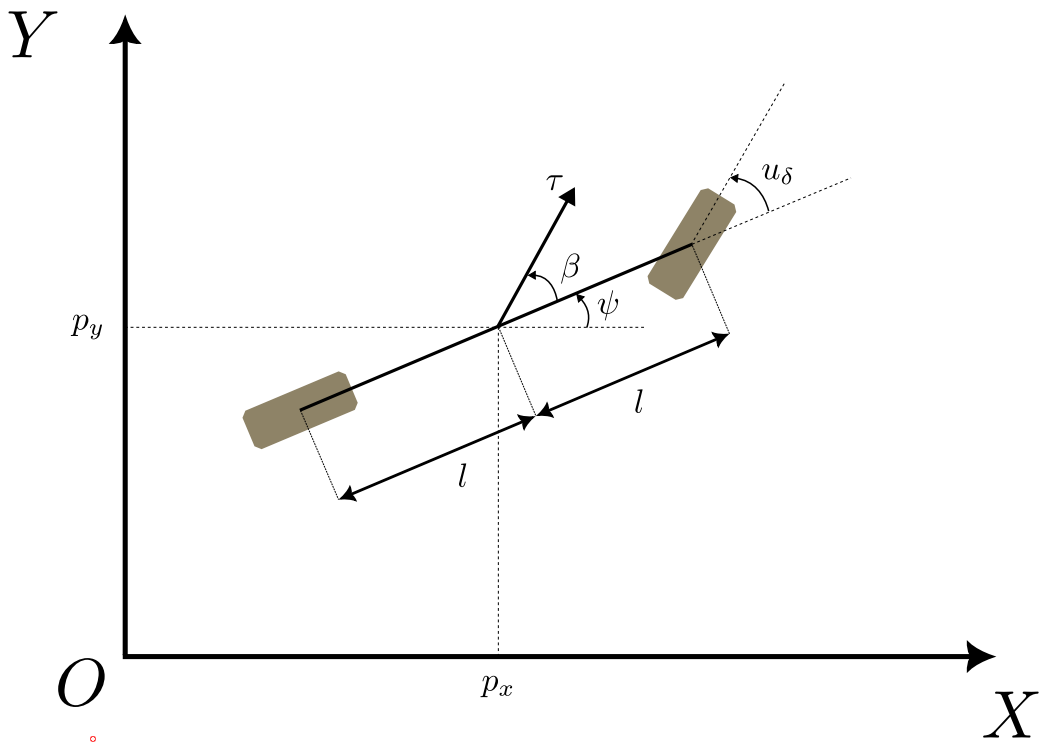

Figure 1: Schematic diagram of the vehicle and the fixed coordinate system (X, Y) with origin O. The position of the vehicle center is denoted by (p_x, p_y), the orientation with respect to the x-axis by ψ and the steering angle by u_δ.

The vehicle is modeled using a simple kinematic bicycle model with two control inputs and can be described by the following equations with discrete time step T_s:

$$p_x(k) = p_x(k-1) + \tau(k-1)\cos(\psi(k-1) + \beta(k-1))T_s, \tag{1}$$
$$p_y(k) = p_y(k-1) + \tau(k-1)\sin(\psi(k-1) + \beta(k-1))T_s, \tag{2}$$
$$\psi(k) = \psi(k-1) + \left(\frac{\tau(k-1)}{l}\sin(\beta(k-1)) + v_\beta(k-1)\right)T_s, \tag{3}$$
$$\tau(k) = \tau(k-1) + (u_c(k-1) + v_{u_c}(k-1))T_s, \tag{4}$$

where p_x and p_y are the world frame coordinates of the center of the vehicle, ψ is the heading angle, τ is the total speed of the vehicle, and β is the slip angle at the center, defined as $\beta(k) = \arctan\left(\frac{1}{2}\tan(u_\delta(k))\right)$. The parameter l represents the distances from the center of the vehicle to the front axle or the rear axle.[1] The model input $u = [u_\delta, u_c]$ consists of the steering angle u_δ and the velocity control term u_c. The process noises $v_\beta(k)$ and $v_{u_c}(k)$ take model uncertainties into account.

The vehicle receives the localization information through GPS. The measurement model for the GPS sensor is given by:

$$z_{p_x}(k) = p_x(k) + w_{p_x}(k), \tag{5}$$
$$z_{p_y}(k) = p_y(k) + w_{p_y}(k). \tag{6}$$

Due to the occlusion of buildings, GPS might sometimes not provide any data. In the Python code, the missing data are represented by `np.nan` in the provided sensor measurements.

Similarly, the compass[2] and the tachometer provide the following data:

$$z_\psi(k) = \psi(k) + w_\psi(k), \tag{7}$$
$$z_\tau(k) = \tau(k) + w_\tau(k). \tag{8}$$

At the initial time $k = 0$, the vehicle is located at $(p_x(0), p_y(0)) = (p_{x,0}, p_{y,0})$ with orientation $\psi(0) = \psi_0$ and speed $\tau(0) = \tau_0$. The initial position $(p_{x,0}, p_{y,0})$ is equally likely to be anywhere inside a circle of radius R, centered at the origin. The probability density functions of ψ_0 and τ_0 are uniformly distributed with $\psi_0 \in [-\bar{\psi}, \bar{\psi}]$ and $\tau_0 \in [0, \bar{\tau}]$. Furthermore, the initial estimation of l is uniformly distributed with $l \in [l_{lb}, l_{ub}]$.

All random variables $(p_{x,0}, p_{y,0})$, ψ_0, τ_0 and noises $\{v_\psi(\cdot)\}$, $\{v_{u_c}(\cdot)\}$, $\{w_{p_x}(\cdot)\}$, $\{w_{p_y}(\cdot)\}$, $\{w_\psi(\cdot)\}$, $\{w_\tau(\cdot)\}$ are mutually independent, where $p_{x,0}$ and $p_{y,0}$ are implicitly dependent with the given initial condition.

Noise Distribution

In the lecture, we typically assume Gaussian noise distributions for Kalman Filtering. But in reality, this may not always be the case. In this programming exercise, we provide two different sets of noise distributions. You may compare the performance of both the EKF and the PF on the two different noise models. The noise models are designed such that the mean and variance are identical for both cases.

[1] In the classic kinematic bicycle model, the parameters l_f and l_r represent the distances from the center of the vehicle to the front axle and rear axle respectively. For simplicity, we define that l_f and l_r share the same length l in our problem.

[2] This compass measurement has previous revolutions encoded and is thus not wrapped to $[-\pi, \pi)$.

Gaussian Noise

The process noise is defined as $v_\beta(k) \sim \mathcal{N}\left(0, \sigma_\beta^2\right)$ and $v_{u_c}(k) \sim \mathcal{N}\left(0, \sigma_{u_c}^2\right)$.

The GPS sensor shares the same model in both directions, where $w_{p_x}(k) \sim \mathcal{N}\left(0, \sigma_{\text{GPS}}^2\right)$ and $w_{p_y}(k) \sim \mathcal{N}\left(0, \sigma_{\text{GPS}}^2\right)$.

The noises in the compass and the tachometer are described as $w_\psi(k) \sim \mathcal{N}\left(0, \sigma_\psi^2\right)$ and $w_\tau(k) \sim \mathcal{N}\left(0, \sigma_\tau^2\right)$.

Non-Gaussian Noise

The process noise is defined as $v_\beta(k) \sim \mathcal{U}\left(-\sqrt{3}\sigma_\beta, \sqrt{3}\sigma_\beta\right)$ and $v_{u_c}(k) \sim \mathcal{U}\left(-\sqrt{3}\sigma_{u_c}, \sqrt{3}\sigma_{u_c}\right)$.

The GPS measurement noises $w_{p_x}(k)$ and $w_{p_y}(k)$ have the same distribution which is a mixture of two Gaussian distributions. For ease of notation we will drop the time index k. Let

$$a \sim \mathcal{N}\left(\frac{\sqrt{3}}{2}\sigma_{\text{GPS}}, \frac{1}{4}\sigma_{\text{GPS}}^2\right) \text{ and } b \sim \mathcal{N}\left(-\frac{\sqrt{3}}{2}\sigma_{\text{GPS}}, \frac{1}{4}\sigma_{\text{GPS}}^2\right)$$

be two GRVs. The PDF of $w_{p(\cdot)}$ is then given by

$$p_{w_{p(\cdot)}}\left(\bar{w}_{p(\cdot)}\right) = 0.5p_a\left(\bar{w}_{p(\cdot)}\right) + 0.5p_b\left(\bar{w}_{p(\cdot)}\right) . \tag{9}$$

Equivalently, we can write the PDF as

$$p\left(w_{p(\cdot)}\right) = \frac{1}{\sigma_{\text{GPS}}\sqrt{2\pi}}\left(\exp\left(-\frac{1}{2}\left(\frac{w_{p(\cdot)} - \frac{\sqrt{3}\sigma_{\text{GPS}}}{2}}{\frac{\sigma_{\text{GPS}}}{2}}\right)^2\right) + \exp\left(-\frac{1}{2}\left(\frac{w_{p(\cdot)} + \frac{\sqrt{3}\sigma_{\text{GPS}}}{2}}{\frac{\sigma_{\text{GPS}}}{2}}\right)^2\right)\right) . \tag{10}$$

The resulting PDF is illustrated in Figure 2.

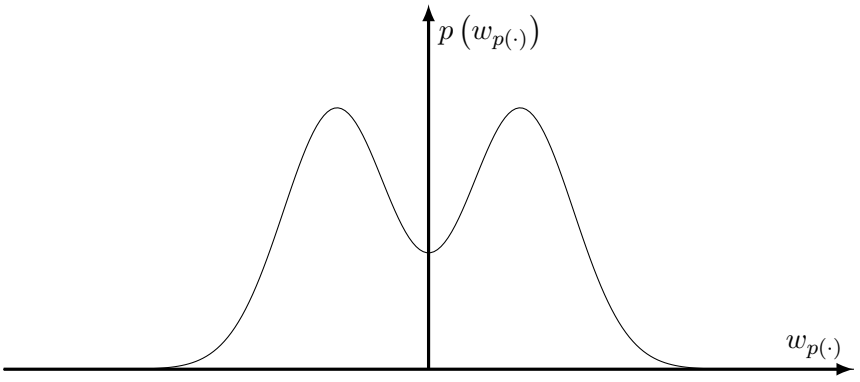

Figure 2: Probability density function of the GPS measurement noise.

The noises in the compass and the tachometer are described as $w_\psi(k) \sim \mathcal{U}\left(-\sqrt{3}\sigma_\psi, \sqrt{3}\sigma_\psi\right)$ and $w_\tau(k) \sim \mathcal{U}\left(-\sqrt{3}\sigma_\tau, \sqrt{3}\sigma_\tau\right)$.

Provided Python Files

A set of Python files is provided on the class website. Please use them to solve the exercise.

`run.py`	Python script that is used to execute a simulation of the true system, run the estimator, plot the results, and report the average run time and root-mean squared tracking error.
`estimator.py`	Python class template to be used for your implementation of the EKF and the PF.
`simulator.pyc`	Python class used to simulate the motion of the vehicle and measurements. This class is called by `run.py`, and is obfuscated (i.e., its source code is not readable).
`const.py`	Constants known to the estimator.

Task

You should design an EKF and a PF to estimate p_x, p_y, ψ, τ and l. Implement your solutions for the estimator in the file `estimator.py`. Your code has to run with the Python script `run.py`. You *must* use *exactly* the definition as given in the template `estimator.py` for the implementation of your estimator.

In order to import the simulator class from `.pyc` file without any problem, please use `Python 3.10` to execute the files. You are only allowed to use the Python packages with the exact version listed in `requirements.txt`, as well as the Python standard library. You may use the command `pip install -r requirements.txt` to install all packages.

While the style of your code is not relevant for evaluation, points will be deduced for severe violation of common sense programming techniques, which result for example in considerably increased computation times.

Evaluation

To evaluate your solution, we will test your EKF and PF on the given problem setup. The `Gaussian` noise will be used to evaluate the EKF, while the `Non-Gaussian` noise will be used for the PF. Moreover, we will make suitable modifications to the parameters in `const.py` and test the robustness of your estimator for different values of the constants. Specifically, the values of the constants R, σ_β, σ_{u_c}, and σ_{GPS} will be modified, where the modification range is given in `const.py`. The constants l_{lb}, l_{ub}, σ_τ, σ_ψ, $\bar{\psi}$, and $\bar{\tau}$ are fixed and will not be modified during evaluation.

The correctness, efficiency and robustness of your method will be key for the evaluation. Specifically, we will evaluate the following requirements:

- Correctness of all the tasks;
- Weighted score of the computational time and estimation accuracy;
- In the case of a draw after the previous point calculation, the submission received first will be rewarded with more points.

Submission

Your submission must follow the instructions here, as grading is automated. Submissions that do not conform will not be graded.

Hand in a single zip-file, where the filename contains your name according to this template (note that there are no spaces in the filename):
`RE25_FirstnameSurname.zip`

The zip-file contains the `RE25_FirstnameSurname` folder (with the correct name as in the zip filename). This folder only contains one file:

- Your implementation of the EKF and the PF in the single file `estimator.py`, which has the exact same function definition as in the provided template.

Send your zip-file in a single email:

- Send the email to `recursiveestimation@ethz.ch`.
- Use this *exact* subject: `Programming Exercise Submission - Firstname Surname`

You will receive an automatic reply confirming receipt of the email. The teaching assistants will not look at the email before the deadline expires, and you are ultimately responsible that we correctly receive your solution in time. Late submissions will not be considered, nor will submissions where the attachment was forgotten. You may submit the solutions multiple times but please follow the deliverable requirements strictly. Only the last submission will be evaluated.

No teamwork is allowed. Submissions with identical codes will not be graded.