TD: Particle filter applied to a multi-sensor localization problem

1 Problem statement

We consider a land vehicle traveling in the plane (O, i, j). We wish to estimate its pose (x, y, ψ). Two proprioceptive sensors are available to estimate the motion of the vehicle. They measure respectively:

- The longitudinal speed at the front wheel: v with a standard deviation error of $0.05\, m.s^{-1}$
- The steering wheel angle: δ with a standard deviation error of $1°$ (attention to units).

Here we consider a front-wheel-drive vehicle, that is to say that the front wheel is driving and steering. We can model such a vehicle with two virtual wheels as illustrated below.

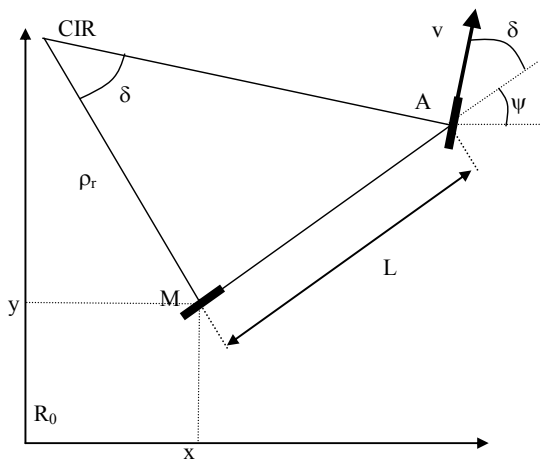

Figure 1: Vehicle modeling with two virtual wheels

It is shown that the kinematic model of a vehicle for which the speed is measured at the virtual front wheel is given by the following equations. It is called a "velocipede model."

$$\begin{cases} \dot{x} &= v \cdot \cos\delta \cdot \cos\psi \\ \dot{y} &= v \cdot \cos\delta \cdot \sin\psi \\ \dot{\psi} &= \dfrac{v}{L}. \sin\delta \end{cases} \tag{1}$$

where L is the distance between the axes of the two wheels ($L = 2m$).

We also have a set of distinguishable beacons from known positions and providing distance measurements through radio signals (cf. Figure 2). It is assumed for simplicity that all the beacons are measured at the same time and, in addition, synchronously with the proprioceptive sensors.

2 System State Space

To establish the system state space, let consider the measurements of speed and steering wheel angle as measured inputs of the system.

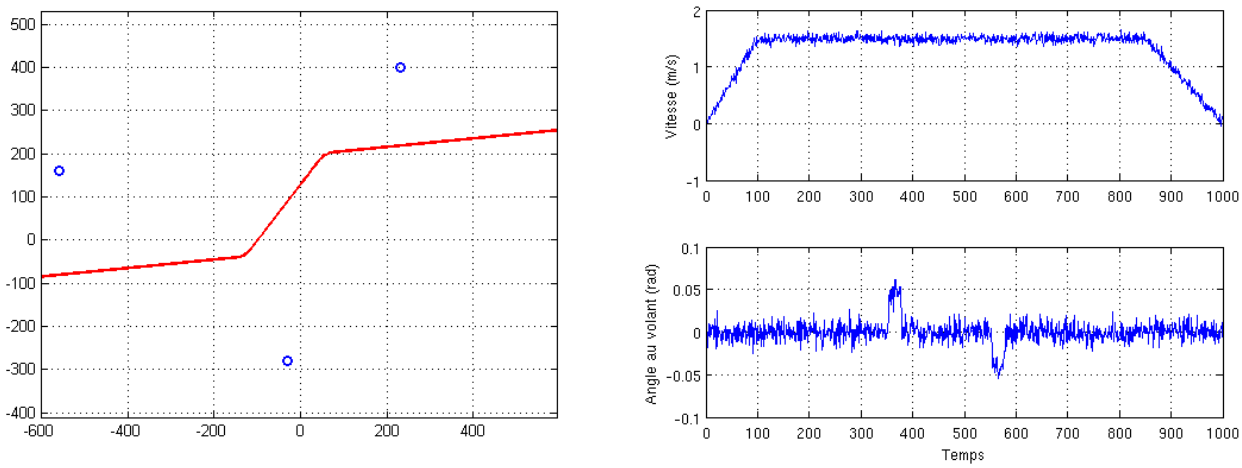

Figure 2: Dataset: reference trajectory with 3 beacons (left) and measured inputs of the system (right)

- Give the state vector to be estimated by the filter.
- Write the discrete evolution model.
- Write the observation model for distance measurements on beacons.
- Give the likelihood of distance measurement on one beacon under Gaussian assumption.
- Give the likelihood of a set of distance measurements on several beacons under Gaussian assumption.

Data to do the simulation

- A data.mat data file containing:
  - Reference Trajectory: *ref_traj (position and heading)*
  - System Inputs: *u (speed and steering wheel angle)*
  - Beacons Data: *N_balises*, *P_balises (positions)*, *y (distance measures)*
- *pf.m*: main file to edit
- *modele_evolution.m*: File describing the evolution model for a particle. To be changed.
- *displaystate.m*: Display function.
- *kitagawa_resample.m*: Resampling function.

3 Development of a well-initialized Particle filter

Run the script *pf.m*. We will modify this file to obtain a functional filter by following a step by step procedure.

3.1 Evolution of the set of particles

Each particle has to evolve according to the evolution model written previously.

1. To do this, change the function *modele_evolution.m*.
2. Initialize all particles in the correct position. Start the filter.
3. In order to ensure optimal coverage of the state space, we will apply a random drawing on inputs and that way we will not be adding model noise. Use the *randn* function that generates a Gaussian noise. Run the filter. What is happening?

3.2 Weights update

For each particle, we use the likelihood of the distance measurements made on the different beacons to update the weight of the particle. In the absence of further information on measures noise, it is assumed that the distance measurements are disturbed by a Gaussian white noise with standard deviation equal to 50 m.

1. Calculate innovations for each of the beacons and implement a bootstrap filter.
2. After the update, it is recalled that the sum of the weights of the set of particles must be equal to 1. Run the filter. What is happening?

3.3 Re-sampling of the set of particles

To avoid the divergence of the filter, we will set up the resampling procedure using the function *kitagawa_resample* each time the set of particles tends to degenerate.

Remember that the number of effective particles can be written as:

$$N_{eff} = \frac{1}{\sum_{i=1}^{N} \omega_i^2} \tag{2}$$

When your filter works, analyze the results based on the number of particles that you take.

3.4 Comparison of different estimators

What is the best estimator (MMSE or MAP) in this kind of problem?

4 Initialization of set of particles

When the initial state is unknown, usually two solutions are possible: either a random drawing of the particles to cover a large area, a distribution grid in an area where the state is very likely.

Change your previous program to make it run this way.

5 Taking into account the fact that the measurement noise is non-Gaussian

One of the major advantages of particle filtering is to allow an easy use of non-Gaussian noise models unlike the Kalman filter. We will now use this advantage.

The measurement noise on distances is actually described by a uniform probability density centered on a support interval S. We recall the expression of a uniform probability density:

$$p(x) = \begin{cases} \frac{1}{S} & \text{if } -\frac{S}{2} < x < \frac{S}{2} \\ 0 & \text{else} \end{cases} \tag{3}$$

with $S = 20\,m$.

Change the update weights of the particles to reflect this noise model, assuming again that the measurement noises are independent.

Test your filter when initialized correctly.

Compare the performance with the results obtained with the Gaussian noise model. What is happening?