Artificial Intelligence in Control Engineering exercise

Lecturer: Dr.Pham Viet Cuong

October 08th, 2018

Group 9:

Nguyen Chinh Thuy	1513372
Nguyen Tan Phu	1512489
Le Van Hoang Phuong	1512579
Do Tieu Thien	1513172
Nguyen Tan Sy	1512872
Nguyen Van Qui	1512702

1 Problem

2 Configuration

3 Implementation

3.1 Particle Filter

To solve the Paticle Filter problem, we implement follow below steps:

- Prediction
  - Implementing a loop with M step which is numbers of particles
  - Using process model and control signals u_t in **VG** which are affected by thermal noise to calculate coordinate $x_t^{[m]}$ of robot.
  - Combining coordinate of robot from above step and coordinate of landmarks in **lm** to calculate range r_t and bearing angle b_t.
  - Calculating importance factor $w_t^{[m]}$ depend on probability density function fomula with μ is matrix of expected range and bearing which are from **XODO**

$$f_x(x_1, x_2, ..., x_N) = \frac{1}{(2\pi)^{\frac{N}{2}} \|\Sigma\|^{\frac{1}{2}}} exp\left(\frac{-1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)\right) \tag{1}$$

- Selection
  - Implementing a loop with M step.
  - Choosing a index in range $[1, M]$ for $x_t^{[m]}$ with probabilities $w_t^{[m]}$.

3.1.1 Python code

```
#-------------------------------------------------------------------------------
#    Libraries
#-------------------------------------------------------------------------------
import numpy as np
from scipy.io import loadmat
from utils.particle_filter import ParticleFilter


#-------------------------------------------------------------------------------
#    Parameters
#-------------------------------------------------------------------------------
```

```
n_particles = 100
sigma_v, sigma_g = 0.5, 3/180*np.pi
sigma_r, sigma_b = 0.2, 2/180*np.pi
wb = 4
time_step = 0.025
particle = "max"


#------------------------------------------------------------------------------
#   Main execution
#------------------------------------------------------------------------------
# Load data
data = loadmat("data20171107.mat")
landmarks, X_gt, Z, U = data["lm"], data["XTRUE"], data["Z"], data["VG"]
n_steps = X_gt.shape[1]


# Create a Particle Filter instance
particle_filt = ParticleFilter(
    n_particles=n_particles,
    n_steps=n_steps,
    landmarks=landmarks,
    sigma_v=sigma_v,
    sigma_g=sigma_g,
    sigma_r=sigma_r,
    sigma_b=sigma_b,
    wb=wb,
    time_step=time_step,
)


# Perform loops
x_start = X_gt[:, 0, np.newaxis]
X_record, W_record = particle_filt.loop_over_steps(x_start, U, Z)


# Visualize the result
mse = particle_filt.compute_MSE(X_gt, X_record, W_record, particle)
print("MSE: %.6f" % (mse))
particle_filt.visualize(X_gt, X_record, W_record, particle)
```

3.1.2 Result

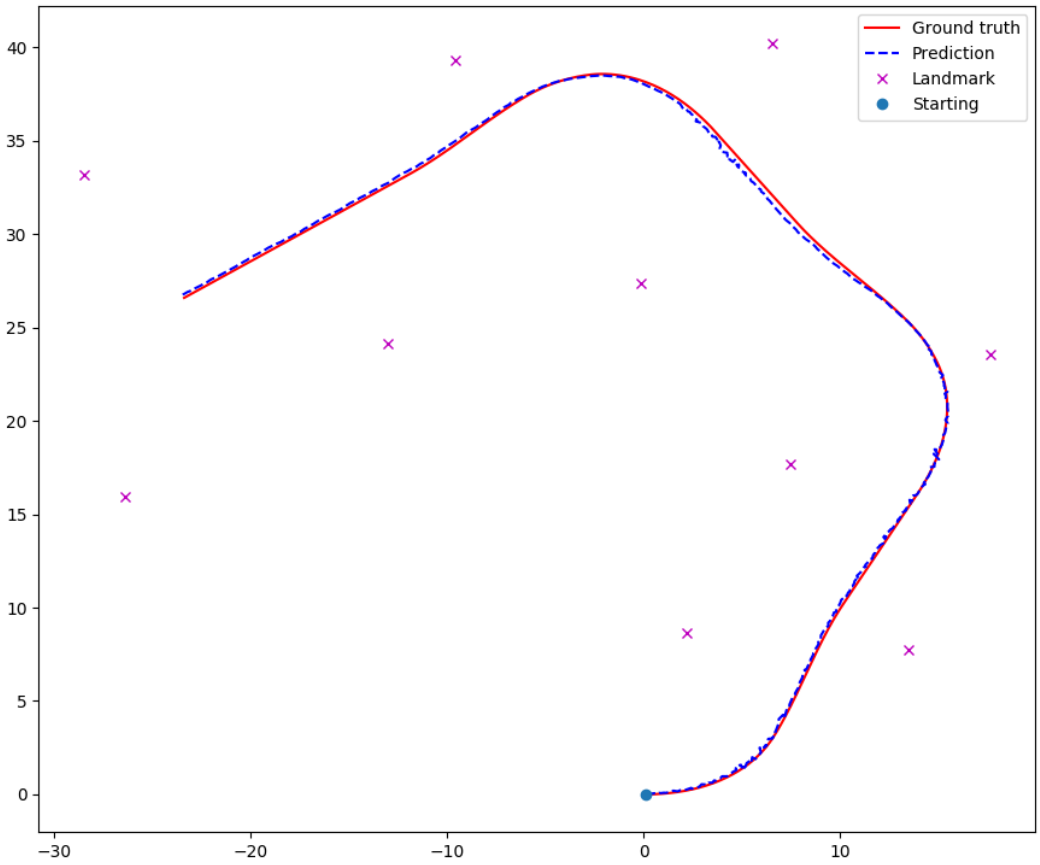

Figure 1: Trajectory