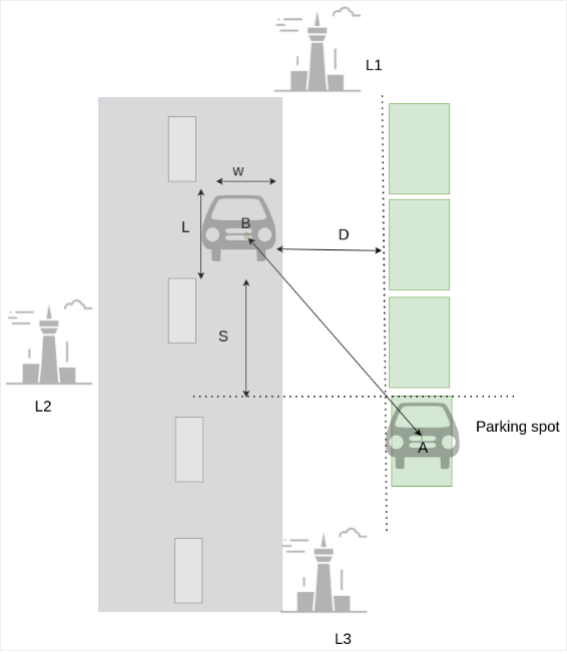
Assignment 4

Ivan Efremov

November 2022

1 Task description

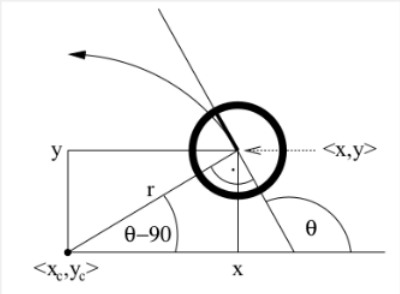


So in our task we have to move robot from point B to A and localize it using sensors data and motion model.

2 Theory

2.1 Motion model

Velocity Motion Model (Noise-free)



- The center of the circle is at

$$\begin{bmatrix} x_c \\ y_c \end{bmatrix} = \begin{bmatrix} x - \frac{v}{\omega} sin(\theta) \\ y + \frac{v}{\omega} cos(\theta) \end{bmatrix}$$

- After δt time, ideal robot will be at $\mathbf{x}_{t+1} = \begin{bmatrix} x_{t+1} & y_{t+1} & \theta_{t+1} \end{bmatrix}$

$$= \begin{bmatrix} x_c + \frac{v}{\omega} sin(\theta_t + \omega\delta t) \\ y_c - \frac{v}{\omega} cos(\theta_t + \omega\delta t) \\ \theta_t + \omega\delta t \end{bmatrix} = \begin{bmatrix} x_t \\ y_t \\ \theta_t \end{bmatrix} + \begin{bmatrix} -\frac{v}{\omega} sin(\theta_t) + \frac{v}{\omega} sin(\theta_t + \omega\delta t) \\ \frac{v}{\omega} cos(\theta_t) - \frac{v}{\omega} cos(\theta_t + \omega\delta t) \\ \omega\delta t \end{bmatrix}$$

Figure 1: Motion model in case of rotation with angular velocity.

```
# In case angular velocity equal to 0
f = sympy.Matrix([
    [x + dt*vt*sympy.cos(theta)],
    [y + dt*vt*sympy.sin(theta)],
    [theta]
])
```

Figure 2: Movement model in case of movement along a straight line.

I decided to use Dubins path planning to design my control. For this task I used an LSR Dubins path.

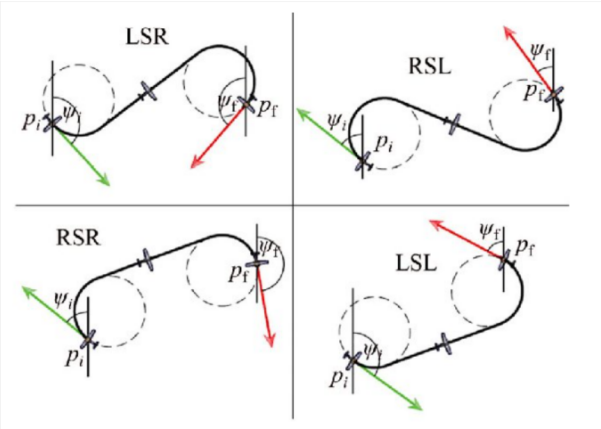


Figure 3: Examples of Dubins pathes.

2.2 Measurement model

- Landmark locations: L1 (5, 30), L2 (5, -30), and L3 (-5, 0), which can be seen by the sensor attached to car. Sensor readings are obtained in the following way

$$\underbrace{\begin{bmatrix} r_t^i \\ \theta_t^i \end{bmatrix}}_{z_t^i} = \underbrace{\begin{pmatrix} \sqrt{(m_{j,x}-x)^2+(m_{j,y}-y)^2} \\ atan2(m_{j,y}-y, m_{j,x}-x)-\theta \end{pmatrix}}_{h(x_{t,j,m})} + N(0,R), \quad (1)$$

where $m_{j,x}, m_{j,y}$ denotes the coordinate of the jth landmark that detected at time t. The optimal robot current location estimation, the vehicle heading angle, and each sensor white noise reading are given by $\mathbf{x}_{t,x}^{-} = x, \mathbf{x}_{t,y}^{-} = y, \theta$, and $R = \begin{bmatrix} \sigma_r^2 & 0 \\ 0 & \sigma_r^2 \end{bmatrix}$, respectively.

Figure 4: Measurement model description.

3 Results

The code for this assignment you can find in my GitHub repository.

Algorithm is implemented according to the theory. Results you can see in figure 5 and 6.

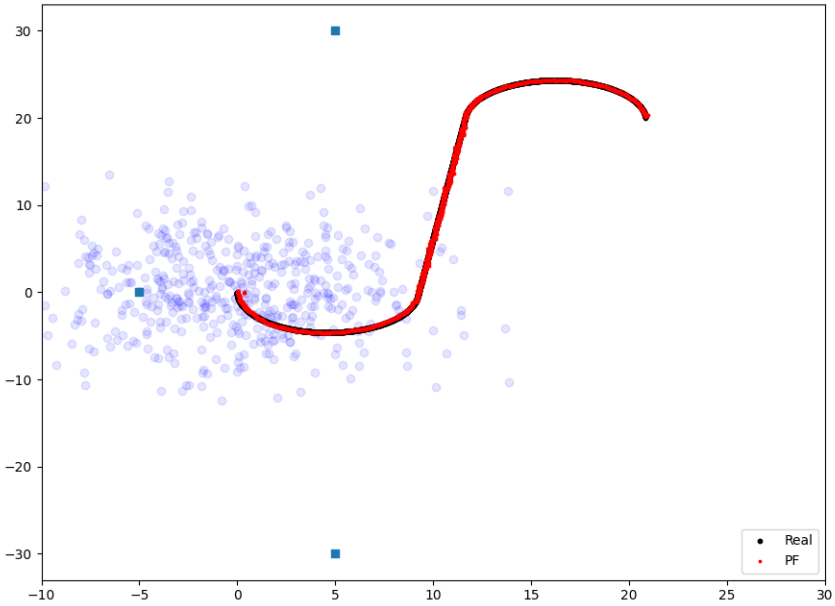


Figure 5: Monte Carlo robot localization.

Blue squares are landmarks. The black line is the ideal simulated Dubins path and the red line is the Monte Carlo trajectory. As we can see Particle filter works quite good.

Also particle filter is quite sensitive to discretization time and to number of particles at the beginning. So if discretization time is low (about 0.01 second) and number of particles is high, so it's more likely that Monte Carlo localization will work properly.

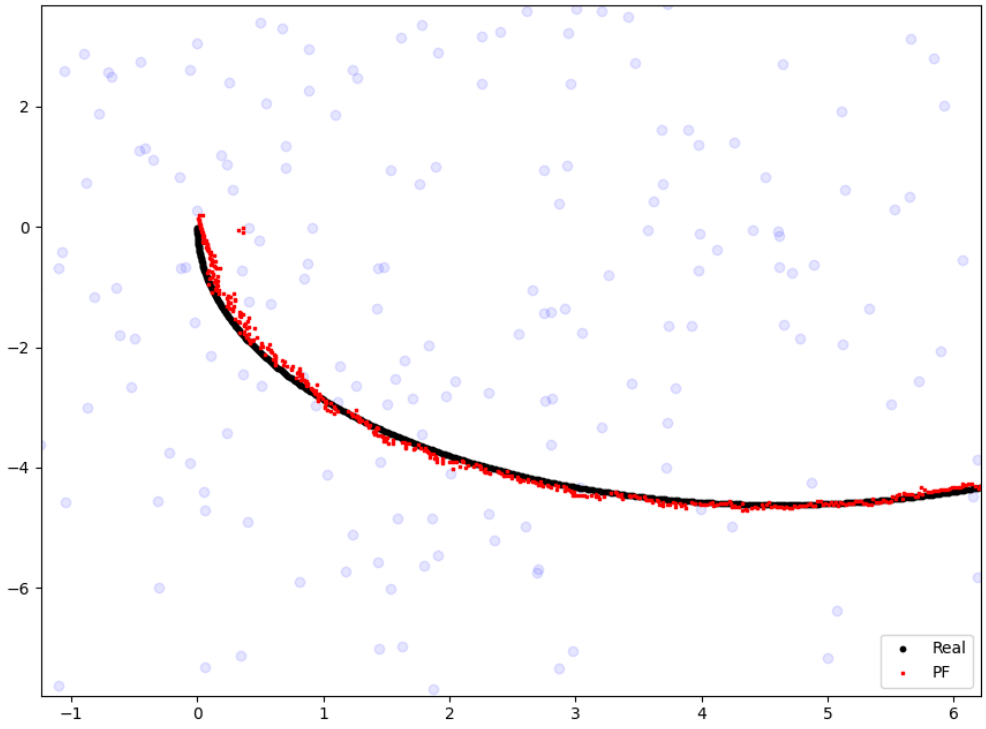


Figure 6: Monte Carlo robot localization zoomed at the beginning.