Particle Filter for 2D localization

Lab – ENGR 509

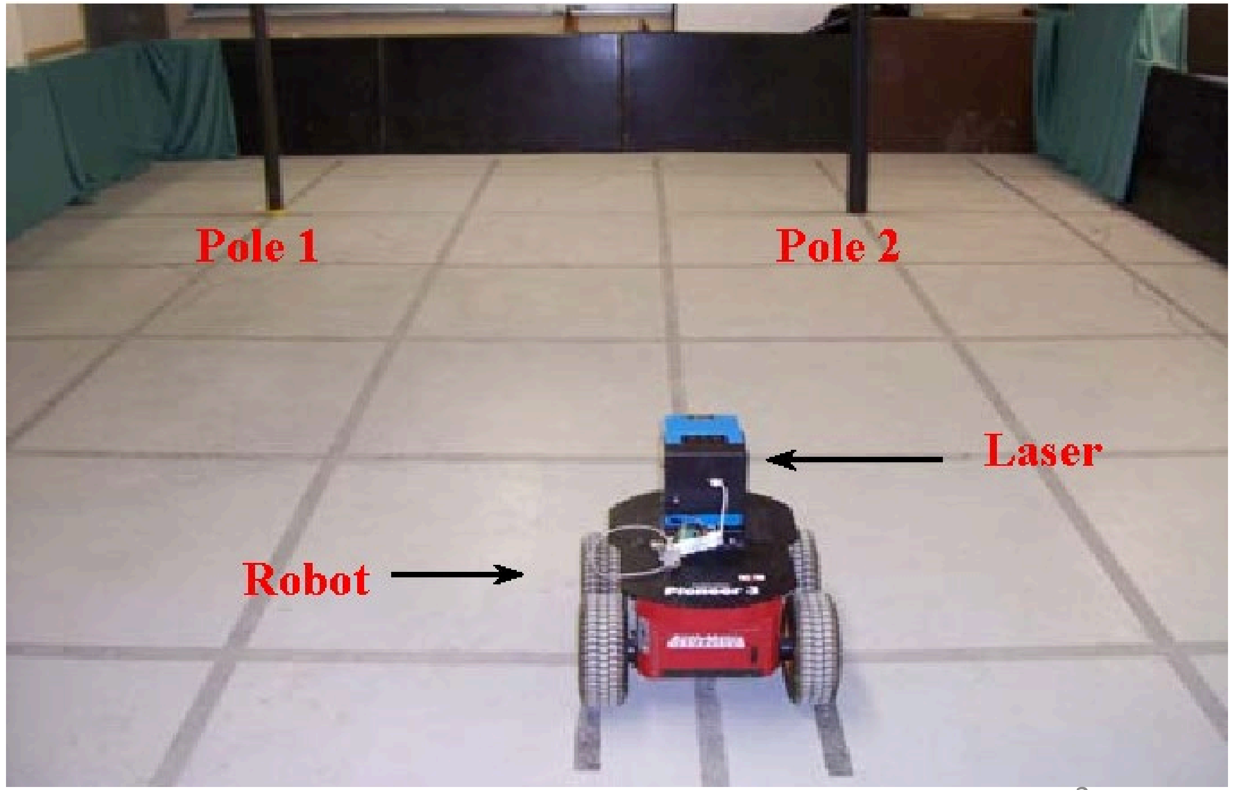
Problem Setup

- Robot localization
  - Given the map, where am I (on the map)?

$$\underbrace{Bel(x_t)}_{\text{current location estimate}} = \eta \underbrace{p(z_t \mid x_t)}_{\text{observation model}} \int \underbrace{p(x_t \mid x_{t-1}, u_t)}_{\text{motion model}} \underbrace{Bel(x_{t-1})}_{\text{previous location estimate}} \, dx_{t-1}$$

Problem setup

- Given map with landmarks
- A range sensor tells how far to a landmark
- Use odometry motion model



Extend to 2D case—Not mandatory

- The world definition and sensor readings, where the odometry motion model and a range-only sensor are used.

You need to do the following:

- Fulfill the sample motion function for particles
- Define the weight update for the particles
- Define the resampling function

A code skeleton with the particle filter framework is provided for you.

Extend to 2D case—Not mandatory

- data -This folder contains files representing the world definition and sensor readings used by the filter.
- code -This folder contains the particle filter framework.
- Run the simulation- in the terminal: *python particle_filter_st.py*. It will only work properly once you filled in the blanks in the file. The blanks include

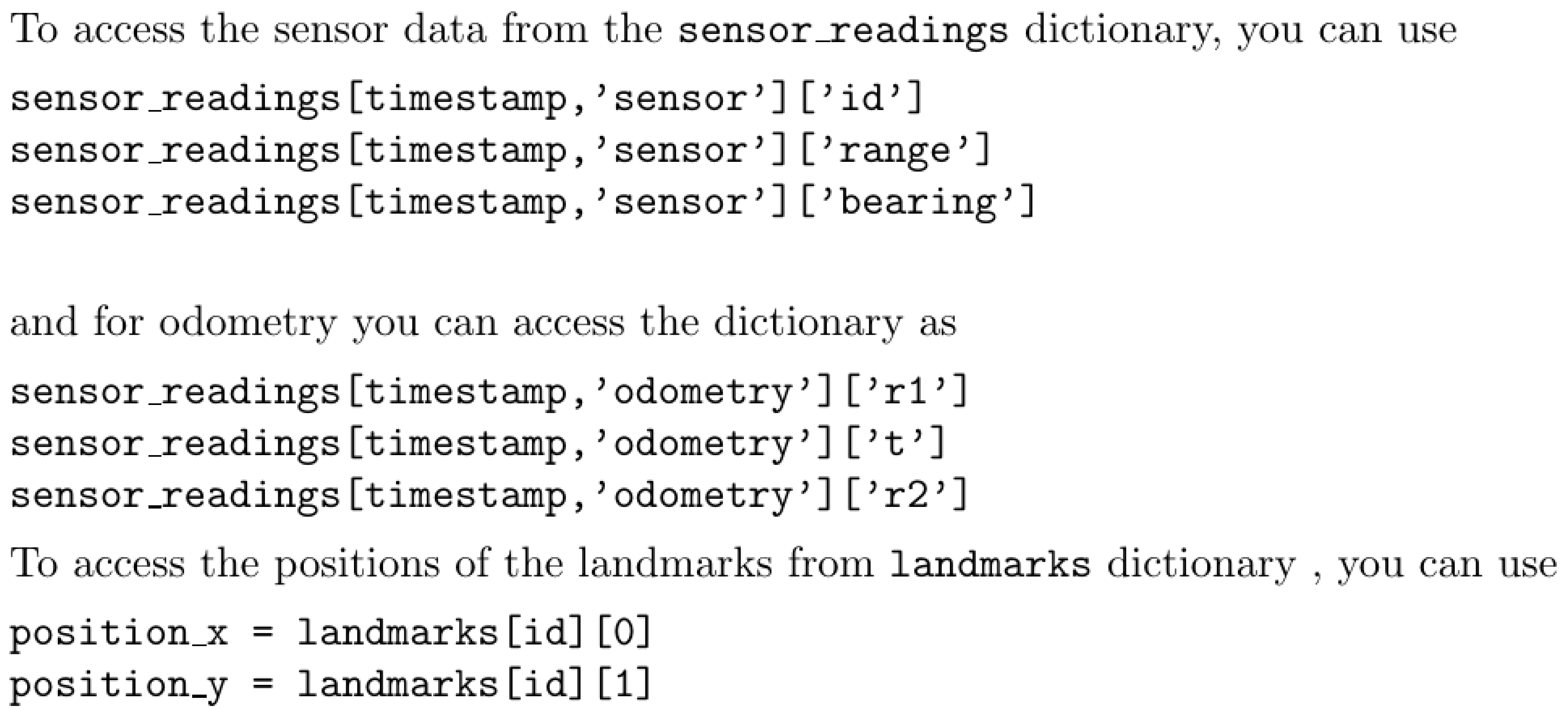
- sample_motion function: **implementing the odometry motion model and sampling from it**, then return the new set of parameters after the motion update. The function samples new particle positions based on the old positions, the odometry measurements and the motion noise. The motion noise are

$$[\alpha_1, \alpha_2, \alpha_3, \alpha_4] = [0.1, 0.1, 0.05, 0.05]$$

- weight_update function: using **a range-only sensor**, it takes the input including the landmarks positions and landmark observations (range), and returns a list of weights for the particle set. The standard deviation of the Gaussian zero-mean measurement noise is σ=0.2.
- resampling function: same as the resampling function in 1D case.

Extend to 2D case—Not mandatory

- Tips: To read in the sensor and landmark data, we have used dictionaries.



To access the sensor data from the `sensor_readings` dictionary, you can use

```
sensor_readings[timestamp,’sensor’][’id’]
sensor_readings[timestamp,’sensor’][’range’]
sensor_readings[timestamp,’sensor’][’bearing’]
```

and for odometry you can access the dictionary as

```
sensor_readings[timestamp,’odometry’][’r1’]
sensor_readings[timestamp,’odometry’][’t’]
sensor_readings[timestamp,’odometry’][’r2’]
```

To access the positions of the landmarks from `landmarks` dictionary , you can use

```
position_x = landmarks[id][0]
position_y = landmarks[id][1]
```