Particle Filter Implementation and Analysis

Joseph Jackson - JJ820

July 19, 2024

Overview

This project implements and analyzes a Particle Filter for robot localization. The implementation is divided into four main parts:

- **Part 1: Control Sequences**
- **Part 2: Landmark Sensor**
- **Part 3: Particle Filter**
- **Part 4: Analysis of Noise and Particle Count**

Part 1: Control Sequences

Control sequences are generated by initializing a random state and adding 20 random controls, each executed for 10 time steps. If a sequence results in the robot approaching the boundary closer than 0.2 units, it is discarded. Valid sequences are visualized by integrating each control forward with a time step of 0.1 seconds according to the motion model equations.

Part 2: Landmark Sensor

The landmark sensor calculates the true Euclidean distance and angle between the robot and landmarks using the arctan2 function. Gaussian noise (mean 0, standard deviation 0.02) is added to the distance and angle readings. Visualizations show the effect of accumulated noise on the sensor readings over time.

Part 3: Particle Filter

A Particle Filter is implemented using the Sampling Importance Resampling (SIR) algorithm. Each particle is weighted based on the likelihood of the measurements given its pose. Particles are resampled according to their weights.

Numerical stability issues were mitigated by adding a small positive value to all weights to prevent particle deprivation.

Part 4: Analysis of Noise and Particle Count

Level of Noise in Readings

- **Low Noise:** Enhances the accuracy of the particle filter, with particles converging closely around the robot's true position. This often reduces the number of iterations required for convergence and slightly decreases runtime.
- **High Noise:** Compromises the accuracy of the particle filter, leading to incorrect convergence or longer convergence times, thereby increasing the runtime.

Number of Particles

- **Fewer Particles:** Leads to underrepresentation of the robot's state distribution, resulting in lower accuracy but quicker computation per iteration.
- **More Particles:** Improves accuracy and resilience against noise but increases computational workload and runtime per iteration.

Comparison with Dead Reckoning

Particle filters continually adjust the position estimate based on sensor readings, providing an advantage over dead reckoning, which is highly susceptible to cumulative errors.

Graphs

The project includes 10 graphs representing errors. The first 5 graphs are from the known location particle filter, and the next 5 are from the kidnapped implementation.

Installation and Usage

To run the project, clone the repository and execute the main script:

```
git clone https://github.com/yourusername/yourrepository.git
cd yourrepository
```

Launch Codes

Simulation and Dead Reckoning

```
python simulate.py --plan controls/controls_1_1.npy --map maps/landmark_1.npy --execution g

python dead_reckoning.py --map maps/landmark_1.npy --execution gts/gt_1_1.npy --sensing rea

python dead_reckoning.py --map maps/landmark_1.npy --execution gts/gt_1_2.npy --sensing rea

python simulate.py --plan controls/controls_4_2.npy --map maps/landmark_4.npy --execution g

python dead_reckoning.py --map maps/landmark_4.npy --execution gts/gt_4_2.npy --sensing rea
```

Particle Filter

```
python particle_filter.py --map maps/landmark_0.npy --sensing readings/readings_0_1_H.npy -

python particle_filter.py --map maps/landmark_3.npy --sensing readings/readings_3_2_L.npy -

python particle_filter.py --map maps/landmark_3.npy --sensing readings/readings_3_2_L.npy -

python particle_filter.py --map maps/landmark_3.npy --sensing readings/readings_3_2_L.npy -

python particle_filter.py --map maps/landmark_3.npy --sensing readings/readings_3_2_L.npy -

python particle_filter.py --map maps/landmark_4.npy --sensing readings/readings_4_2_L.npy -

python particle_filter.py --map maps/landmark_1.npy --sensing readings/readings_1_2_L.npy -
```

Particle Filter (Kidnapped Robot)

```
python particle_filter_kidnapped.py --map maps/landmark_0.npy --sensing readings/readings_0
```

Evaluation

```
python evaluate.py --map maps/landmark_1.npy --execution gts/gt_1_1.npy --estimates estim1/

python evaluate.py --map maps/landmark_0.npy --execution gts/gt_0_1.npy --estimates estim2/
```

Contact

For any questions or issues, please contact joemjackson90@gmail.com.