ANSWERS

U7.2 A cloud of guesses

Localisation · University · about 30 min

What this lesson is about

Representing a belief as a thousand samples, and what that buys over a mean and a variance.

- What this buys
- What it costs
- How many particles

Questions

6 marks in all

1. Why did particle filters take over robot localisation from filters that assume Gaussian noise? [1 mark]

- ☐ **A** Range sensors have likelihoods full of spikes and floors that do not fit a Gaussian, and a particle filter only needs to evaluate the likelihood
- ☐ **B** Particle filters always use less computation than a Kalman filter
- ☐ **C** Particle filters give the same answer on every run, which makes them easier to debug
- ☐ **D** Particle filters need fewer samples as the state gets more dimensions

> **Answer:** A. If you can say how likely a reading is given a pose, you can weight a particle, however ugly that likelihood is.

2. Why is a particle filter hopeless for estimating the configuration of a 12 degree of freedom arm? [1 mark]

- ☐ **A** The number of particles needed grows exponentially with the number of dimensions of the state
- ☐ **B** An arm's motion model is nonlinear, which a particle filter cannot handle
- ☐ **C** The measurement model of an arm is Gaussian, so a Kalman filter is always better
- ☐ **D** Particles can only represent positions, not joint angles

> **Answer:** A. This is the curse of dimensionality. Three dimensions (x, y, heading) needs hundreds to thousands of particles; twelve needs astronomically many.

3. Particles are spread evenly along x from 0 to 200 cm. What is the standard deviation (the spread) of their x values, in cm, to one decimal place? [1 mark]

> **Answer:** 57.7 (accept within 0.1). The standard deviation of a uniform spread over a width L is L / sqrt(12), so 200 / 3.464 = 57.7 cm.

4. What does this program print? [1 mark]

```
xs = [10.0, 10.0, 10.0, 190.0, 190.0, 190.0]
N = len(xs)
mean = sum(xs) / N
spread = (sum((x - mean) ** 2 for x in xs) / N) ** 0.5
print(mean, spread)
```

> **Answer:**
>
> ```
> 100.0 90.0
> ```
>
> The mean of two equal clusters sits halfway between them, at 100, a place neither cluster believes in, and every particle is 90 cm from it. The mean of a two-cluster cloud is not a sensible estimate.

5. Which of these are genuine costs of a particle filter? [1 mark]

Tick every answer that is true.

- ☐ **A** Every particle is moved and weighted every tick
- ☐ **B** The particles needed grow exponentially with the state's dimensions
- ☐ **C** Two runs give slightly different answers, which makes debugging harder
- ☐ **D** It needs a motion model that is linear or can be differentiated
- ☐ **E** It can only represent a belief with one peak

> **Answer:** A, B, C. Compute, the curse of dimensionality and randomness are the costs. Needing a linear model and having one peak are the Kalman filter's limitations, which the particle filter removes.

6. Why must some particles start near the true pose? [1 mark]

- ☐ **A** The filter only ever reweights and copies the guesses it has, so it cannot invent a guess near the truth
- ☐ **B** Particles far from the truth make the weights sum to more than one
- ☐ **C** Distant particles slow the motion update down
- ☐ **D** The weighted mean is only defined when a particle is within one sigma of the truth

> **Answer:** A. If no particle is near the truth, weighting and resampling can only concentrate the cloud on wrong answers.

The task: a thousand guesses

Make a cloud spread evenly over the whole mat and print `particles:`, `mean x:` and `spread x:`.

```
from bugbot import *
import random
connect()

N = 600
```

The hint students can ask for: Scatter the particles evenly over the whole mat with random.uniform(0, 200) for x and y. The standard deviation of a uniform spread over a width w is w over the square root of 12.

A solution

```
from bugbot import *
import random
connect()

N = 600
particles = [(random.uniform(0, 200), random.uniform(0, 200)) for i in range(N)]
xs = [p[0] for p in particles]
mean = sum(xs) / N
spread = (sum((x - mean) ** 2 for x in xs) / N) ** 0.5
print("particles:", N)
print("mean x:", round(mean, 1))
print("spread x:", round(spread, 1))
```

Any program that meets the task's checks is marked correct in the simulator; this is one way, not the only way.