ANSWERS

U7.1 Where am I?

Localisation · University · about 25 min

What this lesson is about

Tracking against global localisation, and why one Gaussian is not enough for the second.

- One reading, many places
- Why symmetry is the hard case
- What the belief has to be able to say

Questions

6 marks in all

1. Why can a Kalman filter not represent the belief of a robot that has just woken up somewhere unknown on the mat? [1 mark]

- ☐ **A** Its belief is a single Gaussian, which has one peak, and the true belief can have several separate peaks
- ☐ **B** It needs a linear motion model, and a robot that has been picked up has no motion model
- ☐ **C** It needs far more computation than a particle filter for the same state
- ☐ **D** It cannot use range sensors, only odometry

> **Answer:** A. A Gaussian has one peak by construction. "I am either here or over there" has two, and global localisation can have dozens.

2. The robot roughly knows where it is and only wants to keep knowing, with a small error and a single blob of belief. What does the page call this localisation problem? [1 mark]

> **Answer:** tracking. Tracking is the easy case, and U6's Kalman filter is exactly right for it. Global localisation is the case where the robot has no idea.

3. The robot faces the far wall and reads its distance. Which set of places on the mat is consistent with that single reading? [1 mark]

- ☐ **A** A stripe across the mat at one value of y, covering every x
- ☐ **B** A stripe up the mat at one value of x, covering every y
- ☐ **C** A single point, since the reading has a definite value
- ☐ **D** A circle of that radius around the middle of the far wall

> **Answer:** A. Facing the far wall the reading is 200 - y, so it pins down y and says nothing at all about x.

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

```
measured = 130.0
possible = 0
for gy in range(0, 200, 10):
    for gx in range(0, 200, 10):
        if abs((200 - gy) - measured) < 8:
            possible += 1
print(possible)
```

Answer:

```
20
```

Only gy = 70 gives a predicted reading within 8 cm of 130 (60 and 80 are 10 cm out), and every one of the 20 values of gx at that row fits, so 20 places.

5. A robot sits in one corner of a square room with four identical corners. What resolves which corner it is in? [1 mark]

- ☐ **A** Moving some distance and taking further readings, so most candidates become inconsistent with the set
- ☐ **B** Averaging many readings from the same spot to reduce the noise
- ☐ **C** Using a range sensor with a smaller standard deviation
- ☐ **D** Using more particles so that every corner is covered

Answer: A. From one spot the information is simply not there, whatever the noise. Motion turns one ambiguous reading into several that disagree about the wrong corners.

6. Which of these beliefs can a single Gaussian represent without behaving badly? [1 mark]

Tick every answer that is true.

- ☐ **A** "I am at (100, 60), give or take 3 cm"
- ☐ **B** "I am on a line somewhere"
- ☐ **C** "I am in one of four corners"
- ☐ **D** "I have no idea"

Answer: A. Only the first. A line and four corners are not single blobs, and "no idea" needs an enormous variance, which behaves badly in practice. A particle filter can say all four.

The task: which places are possible?

Standing still, print `reading:`, the distance the robot measures ahead, and `possible:`, how many positions on a 10 cm grid across the mat would give that same reading to within a few centimetres.

```
from bugbot import *
connect()

readings = []
```

The hint students can ask for: The mat is 200 by 200 and the robot faces along +y, so the wall ahead is at y = 200 and the reading tells you y and nothing about x. Step a grid of candidate positions and count the ones whose predicted reading is within a few centimetres of what you measured.

A solution

```
from bugbot import *
connect()

readings = []
for i in range(20):
    readings.append(distance())
    wait(0.1)
measured = sum(readings) / len(readings)
print("reading:", round(measured, 1))

# every candidate (x, y) on a 10 cm grid, facing the same way
possible = 0
for gx in range(0, 200, 10):
    for gy in range(0, 200, 10):
        predicted = 200 - gy
        if abs(predicted - measured) < 6:
            possible += 1
print("possible:", possible)
```

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