ANSWERS

U8.1 A map of cells

Mapping · University · about 25 min

What this lesson is about

What a map has to do for a robot, why a grid is the usual answer, and what a cell costs.

- What a map is for
- Two families
- Resolution
- Indexing
- The pose is assumed known here

Questions

6 marks in all

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

```
CELL = 5.0
x, y = 102.4, 187.6
col = int(x / CELL)
row = int(y / CELL)
print(col, row)
print(col * CELL + CELL / 2, row * CELL + CELL / 2)
```

Answer:

```
20 37
102.5 187.5
```

Rounding down gives col 20 and row 37, and the centre of that cell is 5 x 20 + 2.5 = 102.5 and 5 x 37 + 2.5 = 187.5.

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

```
CELL = 5.0
grid = [0] * 40
x = -7.0
grid[int(x / CELL)] = 1
print(grid.index(1))
```

Answer:

```
39
```

int(-1.4) is -1, and a negative index in Python counts from the end, so a reading 7 cm off the mat marks cell 39 on the far side. That is why the bounds are checked every time.

3. Why do occupancy grids beat feature maps for planning, despite using far more memory? [1 mark]

- ☐ **A** They represent free space explicitly, and "nothing is there" is what a planner needs most
- ☐ **B** They store each feature with a covariance
- ☐ **C** They need fewer sensor readings to become confident
- ☐ **D** They suit a Kalman filter better

Answer: A. A feature map only contains what the detector recognised and says nothing about the space between features.

4. A 200 by 200 cm mat is mapped with 1 cm cells, each stored as a 4 byte float. How many bytes does the grid need? [1 mark]

> **Answer:** 160000. 200 x 200 = 40,000 cells, and 40,000 x 4 = 160,000 bytes, about 160 kB. At 5 cm it would be 1,600 cells and 6,400 bytes.

5. Which cell size best follows the page's rule of thumb for a robot that must fit through 20 cm gaps, with a sensor whose noise is about 2 cm? [1 mark]

- ☐ **A** 5 cm
- ☐ **B** 1 cm
- ☐ **C** 20 cm
- ☐ **D** 40 cm

> **Answer:** A. The cell should be rather smaller than the smallest gap and rather larger than the sensor noise. At 20 cm the doorway can vanish between two cells; at 1 cm each cell needs many readings.

6. Every mapping task in U8 gives the robot its true pose from the overhead camera. Why? [1 mark]

- ☐ **A** Mapping with known poses is a tidy, solved problem, whereas building a map while localising against it lets the two errors feed each other
- ☐ **B** The robot's sensors cannot measure distance without the camera
- ☐ **C** Occupancy grids cannot be built from odometry
- ☐ **D** The camera is more accurate than any map could ever be

> **Answer:** A. Doing both at once is SLAM. Learning each half separately first is the deliberate choice.

The task: which cell did that reading land in?

The robot stands at (100, 50) on the mat facing straight up it. Print `cells:`, how many cells a 5 cm grid needs for this mat, and `hit col:` and `hit row:`, the cell the reading ahead lands in.

```
from bugbot import *
connect()

CELL = 5.0
W = 40
```

The hint students can ask for: The robot stands at (100, 50) on the mat facing straight up it, so the thing it can see ahead is at y = 50 plus the reading. A cell index is a coordinate divided by the cell size, rounded down.

A solution

```
from bugbot import *
connect()

CELL = 5.0
W = 40                        # 200 cm of mat at 5 cm a cell
print("cells:", W * W)

readings = []
for i in range(12):
    readings.append(distance())
    wait(0.1)
d = sum(readings) / len(readings)

# the robot stands here and faces straight up the mat, so the hit is d further up
X0, Y0 = 100.0, 50.0
hx, hy = X0, Y0 + d
print("hit col:", int(hx / CELL))
print("hit row:", int(hy / CELL))
```

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