WORKSHEET

U8.4 Building a grid

BugBotLab

Mapping · University · about 35 min

Name _____ Class _____ Date _____

What this lesson is about

Eight beams, a known pose, and a map that fills in while the robot drives.

- Storing the grid
- The pose goes in first
- Watching it fill
- The known fraction
- Reading an answer off the map

Questions

6 marks in all

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

```
W = 40
row, col = 3, 7
i = row * W + col
print(i)
print(divmod(1234, W))
```

2. The robot drives at 30 cm/s and the mapper uses the pose from 150 ms after the scan was taken. How far, in cm, is every wall smeared? [1 mark]

3. Every wall in a map comes out shifted and some land off the edge of the grid, but their shapes are right. What is the most likely bug? [1 mark]

- ☐ **A** position() is relative to where the robot started, and the start position was not added
- ☐ **B** heading() was passed to math.sin in degrees
- ☐ **C** The free marking does not stop one cell short
- ☐ **D** L_MAX is too small

4. Put the steps of integrating one tick of scans into the grid in order. [1 mark]

Number the lines 1 to 5 to put them in the right order.

___ `If the range is a real return, add L_OCC at the hit cell`

___ `Add L_FREE once to each cell in the set`

___ `Step along the beam in half cells to one cell short of the range, collecting the cells in a set`

___ `For each beam, work out th = radians(heading + beam angle)`

___ `Fetch the pose and add the start position`

5. The plot of the known fraction has gone flat while the robot keeps driving in the same direction. What does that tell you? [1 mark]

- ☐ **A** Driving further this way is buying almost no new information, so it is time to go somewhere else
- ☐ **B** The map has become wrong and needs resetting
- ☐ **C** The clamp has been reached in every cell
- ☐ **D** The robot has stopped moving

6. Why is it normal to require several occupied cells in a row before believing a wall is there? [1 mark]

- ☐ **A** A single occupied cell can be a stray reflection
- ☐ **B** Walls are always wider than one cell
- ☐ **C** The ray caster cannot see a single cell
- ☐ **D** It removes the half-cell bias

The task: build a grid while you drive

The robot starts at (100, 40). Drive at least 40 cm up the mat, integrating every scan into a log odds grid. Plot `known`, then print `known:`, the fraction of the grid your map has an opinion about, and `wall y:`, where your map puts the near face of the wall.

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

CELL, W = 5.0, 40
L_OCC, L_FREE, L_MAX, FAR = 0.85, -0.4, 8.0, 170.0
START_X, START_Y = 100.0, 40.0
grid = [0.0] * (W * W)
```

Plan your program here, then type it in and press Run.

Do it on the robot

www.bugbotlab.com/learn/u8-4-building-a-grid/

The simulator checks it and tells you when it passes. Nothing to install, no account.

Challenges

1. Corrupt the pose on purpose by adding 10 degrees to the heading. What does the wall look like now?
2. Count how many cells ever change from free to occupied or back. What does a large number mean?
3. Print the map with one character per cell every 2 seconds and watch the wedge grow.