WORKSHEET

U8.6 Frontiers

BugBotLab

Mapping · University · about 30 min

Name _____ Class _____ Date _____

What this lesson is about

The edge between what is known and what is not, and why a map with holes is still useful.

- The definition
- From cells to targets
- Watching exploration saturate
- Shadows
- When it goes wrong

Questions

6 marks in all

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

```
rows = ["FFFU",
        "FOFU",
        "FFFF",
        "UUFF"]
H, W = len(rows), len(rows[0])
count = 0
for r in range(H):
    for c in range(W):
        if rows[r][c] == "F":
            for dr, dc in ((1, 0), (-1, 0), (0, 1), (0, -1)):
                rr, cc = r + dr, c + dc
                if 0 <= rr < H and 0 <= cc < W and rows[rr][cc] == "U":
                    count += 1
                    break
print(count)
```

2. Why is driving to a frontier cell guaranteed to produce new information? [1 mark]

- ☐ **A** It is free, so it can be reached, and it borders a cell no beam has ever reached
- ☐ **B** It is always the cell furthest from the robot
- ☐ **C** It is next to an occupied cell, so the robot will see a wall
- ☐ **D** Its log odds are exactly zero

3. Why can a feature map not support frontier exploration? [1 mark]

- ☐ **A** Absence from its list means both "not there" and "never looked", so it cannot say "unknown"
- ☐ **B** Feature maps cannot store positions accurately enough
- ☐ **C** Feature maps use too much memory for large rooms
- ☐ **D** Feature maps have no free cells to drive to

4. Put the frontier exploration pipeline in order. [1 mark]

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

___ `Group adjacent frontier cells into regions with a flood fill`

___ `Discard regions smaller than the robot`

___ `Score each remaining region`

___ `Find every free cell with an unknown neighbour`

___ `Drive to the best one`

5. The robot alternates for ever between two frontiers of equal value on opposite sides of the room. What is the fix? [1 mark]

- ☐ **A** Hysteresis: keep the current target until it is reached or disappears
- ☐ **B** Discard frontier regions smaller than the robot
- ☐ **C** Use eight neighbours instead of four
- ☐ **D** Lower L_MAX so the map changes its mind faster

6. A small post leaves a wedge of unknown behind it. Why does moving a short distance sideways often beat driving a long way forward? [1 mark]

- ☐ **A** A second viewpoint a little apart sees round the post and resolves nearly all of the shadow
- ☐ **B** Sideways motion gives less odometry error
- ☐ **C** The sensor has a longer range sideways
- ☐ **D** Driving forward would make the robot collide with the post

The task: what have I not seen?

Turn on the spot, building a grid as you go. Plot `known`, then print `explored:`, the fraction of cells that are no longer unknown, and `frontier:`, how many free cells have an unknown neighbour.

```
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, 100.0
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/u8-6-frontiers/

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

Challenges

1. Group your frontier cells into connected regions with a flood fill and print how many regions there are.
2. Print the centre of the nearest region bigger than three cells.
3. Work out how wide the shadow of the post is at the far wall, and check it against your map.