WORKSHEET

U9.7 Project: plan a route and drive it

BugBotLab

Planning · University · about 55 min

Name _____ Class _____ Date _____

What this lesson is about

A plan computed on board, shortened, and then followed across a mat with two walls in it.

- The shape of the program
- Why shortcut at all
- Following waypoints
- When it goes wrong
- What comes next

Questions

6 marks in all

1. Put the stages of the plan and drive program in order. [1 mark]

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

___ `Inflate the walls and the mat edges to build the configuration space`

___ `Print the plan length`

___ `Run A* on the 40 by 40 grid with eight neighbours and the octile heuristic`

___ `Shortcut the grid path down to its corners`

___ `Follow the waypoints with the inverse kinematics`

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

```
import math

def free(x, y):
    return not (40.0 <= x <= 60.0 and y <= 45.0)

def clear(a, b):
    d = math.hypot(b[0] - a[0], b[1] - a[1])
    n = max(2, int(d / 1.5))
    return all(free(a[0] + (b[0] - a[0]) * k / n, a[1] + (b[1] - a[1]) * k / n) for k in
range(n + 1))

path = [(0.0, 0.0), (0.0, 50.0), (50.0, 50.0), (100.0, 50.0), (100.0, 0.0)]
out, i = [path[0]], 0
while i < len(path) - 1:
    j = len(path) - 1
    while j > i + 1 and not clear(path[i], path[j]):
        j -= 1
    out.append(path[j])
    i = j
length = sum(math.hypot(q[0] - p[0], q[1] - p[1]) for p, q in zip(out, out[1:]))
print(len(out), round(length, 1))
```

3. Why must the shortcut's line of sight test use the same inflated map as the search? [1 mark]

- ☐ **A** Tested against the raw obstacles, a shortcut can cut a corner the search was careful to avoid
- ☐ **B** The raw obstacles are not stored after inflation
- ☐ **C** The inflated map has fewer cells, so the test is faster
- ☐ **D** The search and the shortcut must give the same number of waypoints

4. The plan is inflated by 10 cm, although the chassis only needs 4.95 cm. Why? [1 mark]

- ☐ **A** The grid, the shortcut and the waypoint follower all introduce error, and the plan has to absorb it
- ☐ **B** The mat edges are further away than the walls
- ☐ **C** A* cannot plan with margins below one cell
- ☐ **D** The walls are 10 cm thick

5. The printed plan length is right, but the robot does not end up where it should. Where is the fault? [1 mark]

- ☐ **A** In the follower, not the planner, which is exactly why the plan is printed before driving
- ☐ **B** In the A* heuristic
- ☐ **C** In the inflation margin
- ☐ **D** In the grid cell size

6. The robot drives straight past the final waypoint. What is the likely cause? [1 mark]

- ☐ **A** It is fast enough to pass through the arrival tolerance between two ticks, so the speed should ease down as the gap closes
- ☐ **B** A* returned a suboptimal path
- ☐ **C** The shortcut removed the final waypoint
- ☐ **D** The inflation closed the corridor

The task: plan a route and drive it

Plan a route from (30, 30) to the green corner at (170, 170), print `plan:`, its length in centimetres, and then drive it. Do not touch either wall or the edge of the mat.

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

DT = 0.1
CELL = 5.0
N = 40
INFLATE = 10.0
V_MAX, V_LAT = 20.0, 15.0
START = (30.0, 30.0)
GOAL = (170.0, 170.0)
WALLS = [(70.0, 0.0, 8.0, 115.0), (125.0, 85.0, 8.0, 115.0)]
```

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

Do it on the robot

www.bugbotlab.com/learn/u9-7-project-plan-and-drive/

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

Challenges

1. Plan with the cost map from U9.3 instead of plain distance and compare how close the robot comes to a wall on the way.
2. Plan, then move a wall in your program's map by 15 cm without telling the planner, and watch what a wrong map does to a robot that trusts it.
3. Replan every two seconds from where the robot actually is, rather than once at the start. What does that fix, and what does it cost?