ANSWERS

U9.1 The configuration space

Planning · University · about 30 min

What this lesson is about

Grow the obstacles by the robot's radius and the robot becomes a point, which is the whole reason planning is tractable.

- Grow the obstacles instead
- How much to grow by
- Seeing it
- What the free space looks like

Questions

7 marks in all

1. A robot has a 10 cm square chassis and can turn freely. By how much, in cm to two decimal places, must the obstacles be grown so the robot is safe as a point at every heading? [1 mark]

> **Answer:** 7.07 (accept within 0.01). The circle that contains the square at any heading has radius half the diagonal, 10 x sqrt(2) / 2 = 7.07 cm. For the 7 cm BugBot the same working gives 4.95 cm.

2. The simulator models the 7 cm square BugBot as a 3.5 cm circle. Why is that not safe on a real robot? [1 mark]

- ☐ **A** The circle fits inside the square, so a corner of the chassis can reach an obstacle the circle says is clear
- ☐ **B** The circle is too large, so it closes gaps the robot could fit through
- ☐ **C** A circle cannot represent the heading, so the plan cannot turn
- ☐ **D** 3.5 cm is smaller than one grid cell

> **Answer:** A. Half the side gives the inscribed circle, which is not conservative. The circle around the square, 4.95 cm, is.

3. What is the price of planning with the 4.95 cm circle around the square chassis? [1 mark]

- ☐ **A** It occasionally refuses a gap the robot could have squeezed through straight on
- ☐ **B** It occasionally plans a route that clips an obstacle
- ☐ **C** It makes the configuration space four dimensional
- ☐ **D** It makes planning time grow with the heading resolution

> **Answer:** A. A conservative model never plans a collision and sometimes rejects a feasible gap. That trade is almost always the right one.

4. For a holonomic robot with a circular footprint, why does the heading drop out of the configuration space? [1 mark]

- ☐ **A** A circle occupies the same area whichever way it is facing, so collisions do not depend on heading
- ☐ **B** A holonomic robot never changes its heading
- ☐ **C** The heading is always measured separately by the IMU
- ☐ **D** Planners cannot handle more than two dimensions

> **Answer:** A. C-space for a robot that slides and turns is (x, y, heading), but with a circular footprint the obstacles are the same at every heading.

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

```
BLOCK = (80.0, 90.0, 40.0, 16.0)
MARGIN = 10.0
bx, by, bw, bh = BLOCK
print(bw + 2 * MARGIN)
free = 0
for x in range(0, 200, 10):
    if not (bx - MARGIN <= x <= bx + bw + MARGIN):
        free += 1
print(free)
```

Answer:

```
60.0
13
```

Grown by 10 cm on both sides the block spans 70 to 130, 60 cm wide. The lanes at 70, 80, 90, 100, 110, 120 and 130 all touch it, so 13 of the 20 lanes are free.

6. The start and the goal lie in different connected components of the free space. What will a correct planner do? [1 mark]

- ☐ **A** Report that there is no route
- ☐ **B** Find a route through the narrowest passage
- ☐ **C** Return the route with the fewest collisions
- ☐ **D** Keep searching until it finds a route

Answer: A. No planner can invent a route between components. A complete planner reports that truthfully.

7. A planner mysteriously fails on a map where a route looks possible by eye. What does the page say to look for first? [1 mark]

- ☐ **A** A narrow passage only slightly wider than the inflated robot
- ☐ **B** A heuristic that underestimates
- ☐ **C** Too few obstacles in the map
- ☐ **D** A cell size that is too small

Answer: A. A grid can miss a narrow passage because no cell centre lands in it, and a sampling planner may take a very long time to throw a point into it.

The task: grow the obstacle, shrink the robot

Standing still, print `inflated:`, the width of the block in centimetres after growing it by the margin on both sides, and `free lanes:`, how many of the twenty lanes at x = 0, 10, 20 ... 190 a point robot could drive straight up from y = 30 to y = 170 without entering the grown block.

```
from bugbot import *
connect()

BLOCK = (80.0, 90.0, 40.0, 16.0)      # x, y, width, height on the mat
MARGIN = 6.0                          # the 4.95 cm around the square, plus a little
```

The hint students can ask for: The chassis radius is about 3.5 cm, so grow the block by 6 cm on every side and then treat the robot as a single point. A lane at x runs straight up the mat from y = 30 to y = 170, so it is free exactly when the grown block does not cover that x. Try x = 0, 10, 20 and so on up to 190.

A solution

```
from bugbot import *
connect()

BLOCK = (80.0, 90.0, 40.0, 16.0)      # x, y, width, height on the mat
MARGIN = 6.0                          # chassis radius 3.5 cm, plus a little

bx, by, bw, bh = BLOCK
print("inflated:", round(bw + 2 * MARGIN, 1))

# with the block grown, the robot is a point, so a lane is free unless the grown block
covers it
lo, hi = bx - MARGIN, bx + bw + MARGIN
free = 0
for x in range(0, 200, 10):
    if x < lo or x > hi:
        free += 1
print("free lanes:", free)
```

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