ANSWERS

U2.7 Project: a shape in the world

Kinematics and frames · University · about 40 min

What this lesson is about

Drive a square in world coordinates while the robot spins, which only a holonomic drive can do.

- The structure
- What comes next

Questions

6 marks in all

1. Put the structure of the spinning square in order. [1 mark]

Number the lines 1 to 6 to put them in the right order.

___ `apply the inverse kinematics with the heading you have now`
___ `set the speed in proportion to the gap, capped`
___ `add the rotation term, all the way round`
___ `for each corner of the square:`
___ `until close to that corner:`
___ `work out the vector to the corner, in the world`

Answer:

```
for each corner of the square:
until close to that corner:
work out the vector to the corner, in the world
set the speed in proportion to the gap, capped
apply the inverse kinematics with the heading you have now
add the rotation term, all the way round
```

The outer loop walks the corners and the inner loop is a go-to-point controller with a rotation term added. The translation is recomputed every tick against the new heading.

2. world_drive is asked for 10 cm/s due east at three headings the spinning robot passes through. With V_MAX = 20 and V_LAT = 15, what does this print (heading, fwd, lat)? [1 mark]

```
import math
V_MAX, V_LAT = 20.0, 15.0
for h_deg in (0, 90, 180):
    a = math.radians(h_deg)
    wx, wy = 10.0, 0.0
    vx = wx * math.cos(a) - wy * math.sin(a)
    vy = wx * math.sin(a) + wy * math.cos(a)
    print(h_deg, round(100 * vy / V_MAX), round(100 * vx / V_LAT))
```

Answer:

```
0 0 67
90 50 0
180 0 -67
```

East is the robot's right at heading 0 (lat 67), its forward at heading 90 (fwd 50) and its left at heading 180 (lat -67). The same world request becomes a different body command every tick.

3. The robot spins as it should, but instead of heading east its path curls round in circles. What is the likely bug? [1 mark]

- ☐ **A** The heading is read once at the top instead of inside the loop on every tick
- ☐ **B** The spin rate is too fast for the motors
- ☐ **C** The rotation matrix has the wrong sign
- ☐ **D** V_MAX and V_LAT are the wrong way round

Answer: A. The body frame keeps turning, so a rotation worked out with an old heading points the translation the wrong way by more and more. A small steady slant is something else: the drive's lag behind the spin.

4. Along one side of the spinning square, which of these change from tick to tick? [1 mark]

Tick every answer that is true.

- ☐ **A** The heading
- ☐ **B** The body frame command sent to drive()
- ☐ **C** The world direction of travel being asked for
- ☐ **D** The coordinates of the corner being aimed at

Answer: A, B. The world request points at the same corner all along the side. Only the heading changes, and so the body command that delivers that request changes with it.

5. world_drive(10, 0, 25) is called every 0.1 s for 40 ticks. Ideally, how far east does the robot travel, in cm? [1 mark]

Answer: 40. 10 cm/s for 40 × 0.1 = 4 s is 40 cm. The lesson's run ends at (36.0, -6.8): the speed-up at the start costs a few centimetres, and the drive's lag behind the spin slants the path about 10 degrees south.

6. The whole square is driven with odometry() instead of position(). Why is the fourth corner the least square? [1 mark]

- ☐ **A** Dead reckoning error accumulates, and heading error rotates every step after it, so the estimate is worst at the end
- ☐ **B** The inverse kinematics is less accurate for the last side
- ☐ **C** odometry() resets at every corner
- ☐ **D** The fourth side is driven westwards, where the sideways full scale is smaller

> **Answer:** A. Each side adds its error to the ones before. U3 is about how fast that happens and what it costs.

The task: a square while spinning

Visit the four corner zones in order, east then north then west then home, with a rotation term running all the way round.

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

V_MAX, V_LAT = 20.0, 15.0
# the corners, in cm from where the robot started
CORNERS = [(40, 0), (40, 60), (-20, 60), (-20, 0)]
```

The hint students can ask for: Four corners of a square in world coordinates, and the robot rotates the whole way round. Every tick, work out the world vector to the next corner, rotate it into the body frame with the heading you have now, and keep a rotation term on as well.

A solution

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

V_MAX, V_LAT = 20.0, 15.0
CORNERS = [(40, 0), (40, 60), (-20, 60), (-20, 0)]

for tx, ty in CORNERS:
    for tick in range(200):
        x, y = position()
        dx, dy = tx - x, ty - y
        gap = math.hypot(dx, dy)
        if gap < 7:
            break
        speed = min(13.0, 0.6 * gap)
        wx, wy = speed * dx / gap, speed * dy / gap
        a = math.radians(heading())
        bx = wx * math.cos(a) - wy * math.sin(a)
        by = wx * math.sin(a) + wy * math.cos(a)
        drive(100 * by / V_MAX, 100 * bx / V_LAT, 25)
        wait(0.1)
stop()
```

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