WORKSHEET

U6.7 Project: navigate on the estimate

State estimation · University · about 45 min

Name _____ Class _____ Date _____

What this lesson is about

Reach a target the robot cannot see, using nothing but the filter you wrote.

- What the program contains
- The structure
- What comes next

Questions

6 marks in all

1. The robot has moved sideways to x = 30 cm at y = 60 cm, and tag 9 is at (0, 155), straight up the mat from the start. The code treats the tag's range as if it were the y distance. What does it print? [1 mark]

```
import math
x, y = 30.0, 60.0
TAG_Y = 155.0
rng = math.hypot(0 - x, TAG_Y - y)
y_from_fix = TAG_Y - rng
print(round(rng, 1), round(y_from_fix, 1), round(y - y_from_fix, 1))
```

2. How does the lesson suggest avoiding that trap? [1 mark]

- ☐ **A** Gate the fix to when the estimated x is small, or drive up the tag's line first
- ☐ **B** Raise R_TAG to a very large value
- ☐ **C** Use the tag range as the x coordinate instead
- ☐ **D** Take the fix only when the variance is small

3. After a missed run, a plot shows the estimate tracked the truth closely, yet the robot still missed the target. Where is the fault? [1 mark]

- ☐ **A** The controller
- ☐ **B** The filter
- ☐ **C** The tag detector
- ☐ **D** The gyro calibration

4. The controller uses speed = min(13.0, 0.6 * gap). What speed, in cm/s, does it ask for when the gap is 10 cm? [1 mark]

5. The prediction step uses the heading h + 0.5 * rate * DT to rotate the flow into the world frame. Why the half step? [1 mark]

- ☐ **A** It uses the average heading over the tick, since the robot turns during it
- ☐ **B** It halves the gyro noise
- ☐ **C** It corrects for gyro bias
- ☐ **D** It keeps the heading within half a turn

6. Put the parts of the navigation program in the order the lesson lists them. [1 mark]

Number the lines 1 to 4 to put them in the right order.

___ `Plot, to tell estimator faults from controller faults`
___ `Calibrate the gyro`
___ `Steer by the estimate through the inverse kinematics`
___ `Run a filter every tick, predicting with flow and correcting from tag 9`

The task: navigate on the estimate

Reach the green target, which is 105 across and 105 up from the start, using only your own estimate. Print `my x:` and `my y:` at the end, within 12 cm of the truth. `position()` is not allowed anywhere.

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

DT = 0.1
TX, TY = 105.0, 105.0
```

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

Do it on the robot

www.bugbotlab.com/learn/u6-7-project-blind-navigation/

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

Challenges

1. Run it with the tag fix removed. How much worse is it?
2. Plot your estimate and `odometry()` together. Is your filter beating the built-in dead reckoning?
3. Make the robot drive a detour past the tag before turning for the target. Is a longer route with a fix better than a short route without one?