ANSWERS

U3.5 A fix from a landmark

Odometry and drift · University · about 30 min

What this lesson is about

Dead reckoning between landmarks, and a reset whenever something known comes into view.

- The oldest idea in navigation
- A fix from a tag
- Replacing and blending
- When not to trust a fix

Questions

6 marks in all

1. Tag 7 is 165 cm up the mat from the start, and the robot, facing it, reads it at 40 cm. What y does the fix give, in cm? [1 mark]

> **Answer:** 125. fixed = TAG_Y - distance to the tag = 165 - 40 = 125 cm.

2. The estimate is y = 120 and three fixes in a row all say 130. The estimate is blended as in the lesson. What does this print? [1 mark]

```
y = 120.0
for fixed in (130.0, 130.0, 130.0):
    y = 0.8 * y + 0.2 * fixed
    print(round(y, 2))
```

> **Answer:**
>
> ```
> 122.0
> 123.6
> 124.88
> ```
>
> Each blend moves y a fifth of the way to the fix: 120 to 122, then 123.6, then 124.88. The gap shrinks by a factor of 0.8 each time instead of vanishing at once.

3. Dead reckoning with a fix every so often. What does a plot of the position error against time look like? [1 mark]

- ☐ **A** A sawtooth: growing between fixes and dropping to nearly nothing at each one
- ☐ **B** A straight line that grows all run, only more slowly
- ☐ **C** A square root curve that levels off without any fixes
- ☐ **D** Flat at zero, because each fix removes the error for good

> **Answer:** A. Between fixes the error grows as before, and each fix throws the accumulated error away. Only information from outside stops it growing without limit.

4. When is replacing the estimate with the fix, rather than blending, the right choice? [1 mark]

- ☐ **A** When the fix is far better than the estimate
- ☐ **B** Always, because a fix comes from outside the robot
- ☐ **C** When the fix is noisy, so that its noise is removed at once
- ☐ **D** When the tag is far away, since a distant tag covers more of the mat

> **Answer:** A. Replacing throws away the dead reckoning. That is right for a much better fix and wrong for a noisy one, which makes the estimate jump about. The Kalman filter in U6 sets the weight from the uncertainties.

5. Which of these are reasons in the lesson to distrust a fix? [1 mark]

Tick every answer that is true.

- ☐ **A** The tag is far away
- ☐ **B** The tag is seen at a sharp angle
- ☐ **C** It is a single detection
- ☐ **D** The id does not match the landmark you think it is
- ☐ **E** The same id has been detected on two ticks in a row

> **Answer:** A, B, C, D. Range error grows roughly with the square of distance, edge-on tags give worse fixes, and one reading can be a mistake. Two detections in a row from the right id is the cheap safeguard, not a warning sign.

6. Watching the printout, each fix makes a bigger jump in the estimate than the last. What does that tell you? [1 mark]

- ☐ **A** The dead reckoning error between fixes is growing, which points to a bias such as an uncalibrated gyro
- ☐ **B** The tag is moving
- ☐ **C** The fixes are getting noisier as the robot gets closer
- ☐ **D** The blend weight is too high

> **Answer:** A. The jump is the error accumulated since the last fix. If it grows run after run, something one-sided is driving it.

The task: a fix from a tag

Tag 7 is on the far wall, 165 cm up the mat from the start. Drive at least a metre towards it, dead reckoning as you go, take a fix when the tag is close enough to be worth having, and print `my y:` at the end.

```
from bugbot import *
connect()

set_cv("apriltag")
TAG_Y = 165.0
```

The hint students can ask for: Tag 7 sits on the far wall, 165 cm up the mat from where the robot starts. Dead reckon towards it, and when the camera reads it, the fourth number in the tag is how far away it is. That gives you your y directly, so replace the estimate rather than adding to it.

A solution

```
from bugbot import *
connect()

DT = 0.1
TAG_Y = 165.0          # how far up the mat the tag is, from where the robot starts

set_cv("apriltag")
y = 0.0
forward(75)
for i in range(85):
    y += flow()[1] * DT
    seen = [t for t in apriltags() if t[0] == 7]
    if seen and seen[0][3] < 120:
        y = TAG_Y - seen[0][3]        # a fix: replace the estimate, do not add to it
    wait(DT)
stop()
print("my y:", round(y, 1))
print("truth", position())
```

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