WORKSHEET

U3.6 An error budget

Odometry and drift · University · about 25 min

Name _____ Class _____ Date _____

What this lesson is about

Predicting how far out you will be before you drive, which is what covariance is for.

- A budget for a 2 m straight run
- Adding the pieces up
- This is covariance, informally

Questions

6 marks in all

1. This adds up the lesson's error budget for a 2 m run with a calibrated gyro, then again with the noise term removed. What does it print? [1 mark]

```
import math
pieces = {"heading": 0.4, "scale": 6.0, "noise": 0.7, "slip": 2.0}
total = math.sqrt(sum(v * v for v in pieces.values()))
print("total:", round(total, 1), "cm")
print("without noise:", round(math.sqrt(total ** 2 - 0.7 ** 2), 1), "cm")
```

2. Two independent errors of 3 cm each. What is the total, in cm to one decimal place? [1 mark]

3. An uncalibrated gyro bias of 0.19 deg/s acts for 10 s while the robot drives 200 cm at a steady speed, so the heading error grows steadily from 0 to 1.9 degrees. Using d × e with the average heading error, how far to the side does it end up, to the nearest cm? [1 mark]

4. Flow noise contributes 0.7 cm over a run of 100 steps. Following the random walk, what does it contribute over 400 steps, in cm? [1 mark]

5. With the gyro calibrated, the budget reads heading 0.4 cm, scale 6 cm, noise 0.7 cm and slip 2 cm. What should be worked on next? [1 mark]

- ☐ **A** The flow scale error
- ☐ **B** The noise, by averaging more readings
- ☐ **C** The slip
- ☐ **D** The gyro bias again

6. For that calibrated budget after a long straight run, what shape is the region the robot is likely to end up in? [1 mark]

- ☐ **A** An ellipse, longest along the direction of travel, because the scale error dominates
- ☐ **B** A circle, because the errors are added in quadrature
- ☐ **C** An ellipse, longest across the direction of travel, because heading error dominates
- ☐ **D** A line along the direction of travel, because heading error is zero after calibration

The task: an error budget

Print what you expect the dead reckoning error to be over your run, as `predicted error:`, before you drive. Then drive at least 90 cm and print `actual error:`, the real gap between `odometry()` and `position()`.

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

DISTANCE = 100.0
```

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

Do it on the robot

www.bugbotlab.com/learn/u3-6-an-error-budget/

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

Challenges

1. Run the same thing at `set_noise(3)` and see which line of the budget you should have tripled.
2. Budget a run with four corners in it. What changes?
3. Repeat the run ten times and compare the spread of the actual errors with your predicted number.