WORKSHEET

U7.6 Monte Carlo localisation

Localisation · University · about 35 min

Name _____ Class _____ Date _____

What this lesson is about

The three steps in a loop on a real mat, with tags as the measurement.

- Mat coordinates, not travel
- Only while it faces the wall
- Using tags as well
- When it goes wrong

Questions

7 marks in all

1. Put one tick of Monte Carlo localisation in order. [1 mark]

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

___ `Weigh every particle against the sensor reading`
___ `Move every particle by the odometry, plus noise`
___ `Report the weighted mean as the estimate`
___ `If neff is low, resample`

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

```
depth = [0.9, 0.3, 0.05]
tag = [0.05, 0.3, 0.9]
both = [a * b for a, b in zip(depth, tag)]
total = sum(both)
print([round(w / total, 2) for w in both])
added = [a + b for a, b in zip(depth, tag)]
print([round(w, 2) for w in added])
```

3. The filter has a weight from the depth sensor and a weight from a tag seen by the camera for each particle. How should they be combined? [1 mark]

- ☐ **A** Multiply them, which is Bayes' rule for independent measurements
- ☐ **B** Add them, so that each sensor contributes its share
- ☐ **C** Take whichever is larger
- ☐ **D** Average them, so neither sensor dominates

4. The cloud collapses onto the wrong answer. Which of these could cause it? [1 mark]

Tick every answer that is true.

- ☐ **A** Not enough jitter when resampling
- ☐ **B** A measurement sigma that is too small
- ☐ **C** A motion model that is too confident
- ☐ **D** Readings taken while the robot turns, weighed with a model that assumes it faces the wall square on
- ☐ **E** A measurement that is not informative enough to rule places out

5. The spread of the cloud stays large however long the robot stands still. What does the page recommend? [1 mark]

- ☐ **A** Move the robot, so one ambiguous reading becomes several that disagree about the wrong places
- ☐ **B** Inject more scattered particles each tick
- ☐ **C** Reduce the jitter to zero
- ☐ **D** Increase the number of particles until it collapses

6. The filter tracks well for a minute and then loses the robot for good. What is the likely cause and fix? [1 mark]

- ☐ **A** Particle deprivation; inject a few scattered particles each tick
- ☐ **B** Too many particles; reduce N
- ☐ **C** Sigma too large; reduce it
- ☐ **D** The odometry noise is too small; set it to zero

7. What does MCL give the robot that dead reckoning does not? [1 mark]

- ☐ **A** A position in the mat's own frame, which other things are described in, rather than a distance from wherever it started
- ☐ **B** A smoother velocity estimate
- ☐ **C** An estimate with no noise in it
- ☐ **D** The map of the mat

The task: Monte Carlo localisation

Run the full filter while driving at least 50 cm. Plot `spread`, and print `my y:`, the robot's position on the mat as your filter has it. No `position()`. Drive straight, and expect the tag card to pull your answer a few centimetres ahead once the robot is near it.

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

DT, N, SIGMA = 0.1, 300, 3.5
particles = [random.uniform(0, 200) for i in range(N)]
```

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

Do it on the robot

www.bugbotlab.com/learn/u7-6-monte-carlo-localisation/

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

Challenges

1. Start with the cloud in the wrong half of the mat only. Does it recover, and how?
2. Add the tag on the far wall as a second measurement and compare how fast the spread collapses.
3. Drop to 30 particles. At what point does it stop working?
4. Part way along, stop, turn 30 degrees left and straight back, and keep weighing all the while. Where does the estimate go, and does it come back? Then skip the readings until the robot faces the far wall again.