Simulate Movements of a Robotic Lawnmower

Jonas Lundberg, jonas.lundberg@lnu.se
Department of Computer Science, Linnaeus University

Abstract— **A robotic lawnmower moves across a lawn in order to cut the grass. In this programming project we will try to simulate the movement of such a robot and try to estimate how much of the lawn is cut after a certain amount of time.**

I. Notations and Recommended Approach

In this section we introduce various notations and present a suggested approach for how to simulate the movements of a robotic lawnmower.

The Ground Map

The robot moves across a *ground* described by a *grund map*. A ground map is a simple csv file consisting of three different symbols: L is the lawn to be cut, O is an obstacle (e.g. a house or a tree) that should not be cut, and S is the robot starting point. Any position outside the map is considered an obstacle and should not be cut. Each symbol represents a square area of 1 m^2. The actual starting point is the lower left hand corner of the start square S. The start square S is a part of the lawn that should be cut.

Simple.csv

```
L,L,O,L,L
L,L,O,L,L
L,L,L,L,L
L,L,L,S,L
```

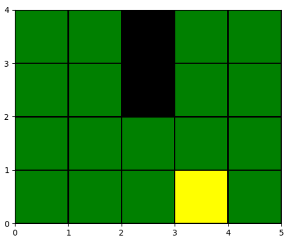

Fig. 1
Simple ground map (left) visualized using `matplotlib` (right).

Figure 1 shows a simple example of a ground map `simple.csv`. On the left-hand side we see the csv markup and on the right we see a `matplotlib` visualization of the same map. This map represents a very small ground ($4m \times 5m$) with $18m^2$ of lawn and $2m^2$ of obstacles.

Coordinates and Speed

A position on the ground is given by a pair of coordinates (x, y). The origo of the coordinate system is in the lower left corner of the map. The start position (x^0, y^0) for the robot (lower left corner of the start square) in ground map `simple.csv` is then given by the coordinates $(3.0, 0.0)$.

The robot speed v is always $0.3m/s$ and the robot velocity at any moment has two components v_x, v_y such that $\sqrt{v_x^2 + v_y^2} = v$. Given an arbitrary robot position (x^i, y^i) we can then compute a new position (x^{i+1}, y^{i+1}) after a time Δt as:

$$
\begin{aligned}
x^{i+1} &= x^i + v_x \Delta t \\
y^{i+1} &= y^i + v_y \Delta t.
\end{aligned} \tag{1}
$$

Using the above procedure we can iteratively simulate the movements of a robot with velocity (v_x, v_y) from a start position (x^0, y^0). The movement will be a straight line until we hit an obstacle and change the robot direction

Random Movement and Bouncing

The robot will move in a certain direction until it hits an obstacle. A new random direction is generated by generating a new random velocity (v_x, v_y). This can be accomplished by generating a random angle $\alpha \in [0, 2\pi]$ and compute the new velocity as

$$
\begin{aligned}
v_x &= v\cos(\alpha) \\
v_y &= v\sin(\alpha).
\end{aligned} \tag{2}
$$

Approach (2) to generate random velocities guarantees that the robot always moves at the same speed v.

The robot should *bounce* when it hits an obstacle. We recognize hitting an obstacle when we in a given position (x^i, y^i) compute a new position (x^{i+1}, y^{i+1}) (using (1)) that is inside an obstacle or outside the map. In that case we should *not* move to the new position but instead generate a new random velocity (using (2)) and try to take a step in that direction. We repeat this process (generate new random velocity and take a new step) until we find a new position (x^{i+1}, y^{i+1}) that is outside any obstacle. That is, the robot will move in a straight line until it hits an obstacle, then it will stop and start moving in a new random direction starting from the point where the obstacle was hit.

The Robot Trace

Each step the robot takes across the lawn gives a new position (x^i, y^i). All steps form a *trace*, a sequence of all steps, that starts in position (x^0, y^0). Figure 2 shows two such traces. The trace on the left shows the robot movements after 15 minutes, the right hand figure shows the trace after 30 minutes (in a separate simulation). Each run of the simulation will give a separate trace due to our random choice of velocities.

Coverage

A robot trace, like in Figure 2, give a hint about which part of the lawn that has been cut or not. In order to get a qualitative estimate of the *coverage* (i.e., percentage of

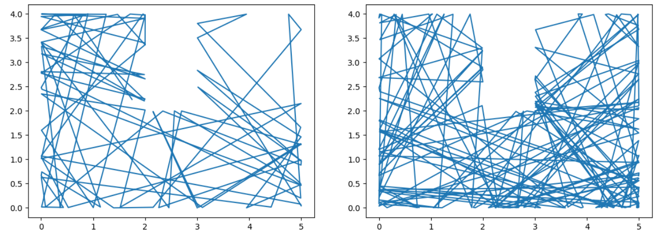

Fig. 2
ROBOT TRACE AFTER 15 MINUTES (LEFT) AND 30 MINUTES (RIGHT).

the lawn that is cut) we divide each square meter in the ground map into a smaller $N \times N$ grid. We refer to these smaller squares as *pixels* and the entire map of pixels as the *coverage map*.

For example, if we divide all 18 lawn squares in Figure 1 into a 5×5 grid we get a coverage map with 450 pixels.

We then, during trace construction or afterwards, take each position (x^i, y^i) in the trace, compute the corresponding pixel (X^n, Y^m), and mark this pixel as *visited*. After the simulation we can then compare the number of visited pixels with the total number of pixels and compute a coverage.

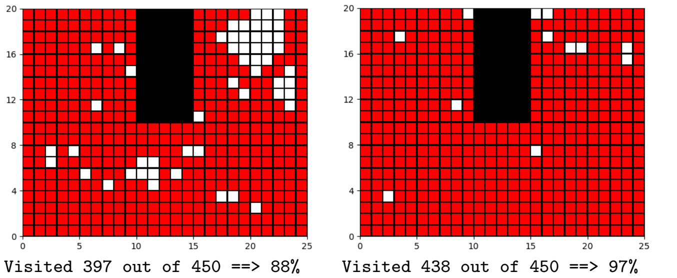

Fig. 3
COVERAGE AFTER 15 MINUTES (LEFT) AND 30 MINUTES (RIGHT).

Figure 3 shows two coverage maps (using a 5×5 grid) based on the two traces in Figure 2. Once we know the number of visited pixels we can easily compute the coverage (88% and 97% respectively).

II. Exercises and Grade Requirements

As a starting point, for grades E and C, we expect each project to implement the approach (compute trace, bouncing mechanism, coverage computation) that was outlined in the previous section. We also expect a written report that (at a minimum) handles the more complex ground map `small_house.csv` as well as your own map `my_map.csv`.

All grades: Create your own ground map `my_map.csv`.

Grade E

Implement support for a single robot simulation that 1) reads a csv ground map, 2) shows a `matplotlib` visualization of the ground map (similar to right-hand side of Figure 1), 3) computes and visualizes a trace (similar to Figure 2), and 4) computes coverage (e.g. 93.2%). Questions: What is the coverage after two hours when using the ground map `small_house.csv`? How do the two parameters Δt and N influence the coverage?

Grade C

In addition to the grade E requirements: 1) make a `matplotlib` visualization of the coverage map similar to Figure 3, 2) add support for multiple simulations in a single run of the program that makes it possible to compute the average coverage for a given map after a given time. The trace and coverage plots can in this case be based on the final simulation. 3) Create a new interesting ground map. Questions: What is the effective cutting width of the lawnmower and what input parameters (v, Δt and N) influence the width? Also, what is a reasonable workload (hours of work per day) for the robot to handle ground map `small_house.csv`? Motivate your answers and your assumptions.

Grade A

In addition to the grade C requirements: Improve the approach outlined in the previous section. Try to make it more realistic and/or suggest improvement in the Robot AI that improves the coverage. Present experiments and visualizations that motivates and/or evaluates your improvements.