PROJECT 4: PROBABILISTIC ACTIONS

Due: Wednesday, March 12 11:59pm EST

The objective of Project 4 is to implement probabilistic actions and belief state computation using Markov chains. In this lab, you will work entirely in simulation to validate your algorithm.

This project consists of these two main steps:

- A state update step, in which the new state of the robot is determined based on the action and state transition probabilities at the current state
- A belief update step, in which the new belief state of the robot is computed based on the action and conditional probability table

A. Overview

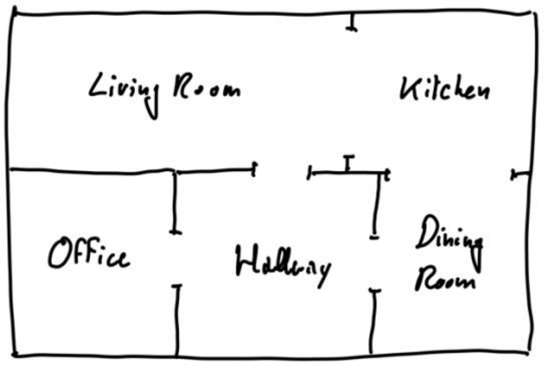


Figure 1: An example map

In this project, a simulated robot moves in a map (Figure 1) according to a pre-defined sequence of actions. The state of the robot is given by the type of the space the robot is currently located. At each state, the robot executes one of the actions, (‘Left’, ‘Right’, ‘Up’, ‘Down’), and each action results in a new robot state with known probabilities. For example, if the robot is in the office and moves ‘Right,’ the robot will end up in the hallway with a probability *p,* and it will remain in the office with the probability of *1-p*. Your task is to implement the probabilistic actions and belief state computation for the robot.

Please see the lecture slides for more details.

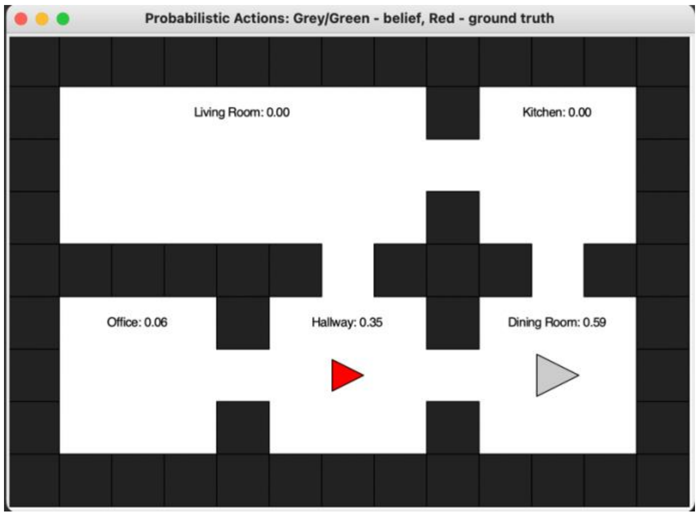


Figure 2: The simulation GUI window

Simulator: After launching the local simulator, you will see a GUI window that shows the map, actual state and the maximum of the belief state of the robot (Figure 2). The ground truth robot state will be shown as red triangle, and the gray triangle represents the robot's maximum belief state. The belief probability for each state is shown in each room. **Note that the red and grey triangles may not always align.**

B. Project Files

- **grid.py:** Handles the grid environment used for the GUI Graphical User Interface
- **gui.py:** Contains helper functions for the GUI.
- **pa_gui.py:** Creates a GUI for visualizing the probabilistic actions.
- **utils.py:** Contains utility functions commonly used in the project.
- **robot.py:** Handles the probabilistic actions and belief state computations of the robot
- **setting.py:** Contains various configuration settings used throughout the project.
- **test_robot.py:** Contains unit tests for robot.py
- **map_house.json:** Contains the geometry of a house map
- **map_office.json:** Contains the geometry of an office map

C. Instructions

In this lab, we provide skeleton code in **robot.py**, and your job is to fill in the missing pieces. To install the necessary libraries to complete the project and run the GUI, call **pip install -r requirements.txt**.

Implement all the missing functions in **robot.py** to build a robot that executes probabilistic actions and tracks its belief state. You may look at any of the other provided files, but please do not modify them, as the autograder expects those files to remain unchanged.

To run the local sanity checks for robots.py, call **python3 test_robot.py** in your terminal. This will provide sanity checks for all functions except update, which is expected to be debugged through the local simulator.

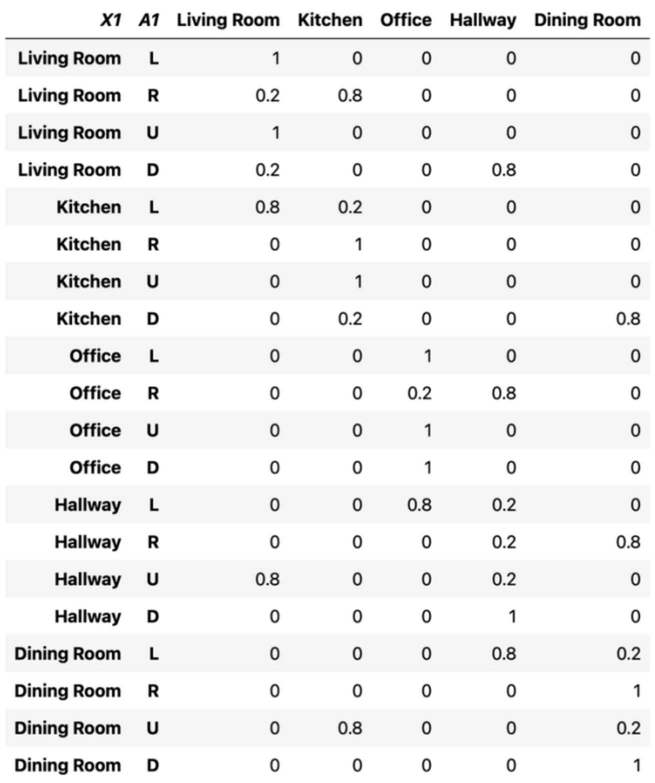
To run the local simulator, call **python3 pa_gui.py** in your terminal. (Note this will not work until you implement the required functions or temporary workarounds, the method docstrings will call out any such functions/workarounds).

- To test on the house map (smaller): **python3 pa_gui.py**
- To test on the office map (larger): **python3 pa_gui.py --map map_office**

Below, we provide some helpful tips for the project and additional details on grading and submission.

D. Hints for Code Completion

- **robot.py** is the only file that needs to be completed
- The conditional probability table is given in **setting.py** in the form of dictionary.



X1	***A1***	**Living Room**	**Kitchen**	**Office**	**Hallway**	**Dining Room**
Living Room	**L**	1	0	0	0	0
Living Room	**R**	0.2	0.8	0	0	0
Living Room	**U**	1	0	0	0	0
Living Room	**D**	0.2	0	0	0.8	0
Kitchen	**L**	0.8	0.2	0	0	0
Kitchen	**R**	0	1	0	0	0
Kitchen	**U**	0	1	0	0	0
Kitchen	**D**	0	0.2	0	0	0.8
Office	**L**	0	0	1	0	0
Office	**R**	0	0	0.2	0.8	0
Office	**U**	0	0	1	0	0
Office	**D**	0	0	1	0	0
Hallway	**L**	0	0	0.8	0.2	0
Hallway	**R**	0	0	0	0.2	0.8
Hallway	**U**	0.8	0	0	0.2	0
Hallway	**D**	0	0	0	1	0
Dining Room	**L**	0	0	0	0.8	0.2
Dining Room	**R**	0	0	0	0	1
Dining Room	**U**	0	0.8	0	0	0.2
Dining Room	**D**	0	0	0	0	1

Figure 2: An example of a conditional probability table

E. Submission

- Submit the updated **robot.py** to Gradescope by the deadline.
- Include your name in a comment at the top of each file.

F. Grading

- Robot.py (100 pts)
  - *compute_state()* – 30 points
  - *compute belief()* – 70 points

G. Additional Notes

- Ensure your implementations adhere to the specified requirements and maximize your score potential by following the instructions carefully.
- To install the necessary libraries to complete the project and run the gui, call pip install -r requirements.txt.
- You may look at any of the other provided files, but please do not modify them, as the autograder expects those files to remain unchanged.
- **Passing the sanity checks does not guarantee a successful implementation as these tests are minimal, but they should indicate that you are on the right track!**
- **The test_robot.py is just designed for you to debug. We only consider your score on Gradescope when calculating your final score for this project.**