Software Engineering Intern Project
Carbon Robotics
San Francisco, California

Anirban Sinha

February 19, 2019

1 Task Overview

The task is to simulate autonomous non-holonomic point robots moving around a $2D$ grid world with obstacles in places. In order to accomplish this task I have implemented three classes, namely **World**, **Robot** and **pt2d** respectively. Package has been built successfully and tested as functional. While testing the $2D$ grid world size has been considered to be 7×7 dimensional which includes a 2×2 size obstacle in it. Apart from the obstacles, the world is surrounded with non-penetrable boundaries from all sides. Further, two autonomous robots have been spawned in the world at two different locations to move autonomously and tested for 100 iterations to run in the environment.

2 roboagents Package

2.1 Class Descriptions

This package consists of two directories, namely **include** and **src**. The **include** directory consists of the header files of three classes (**World**, **Robot** and **pt2d**) that has been implemented here and complete definitions of each class methods has been provided in *src* folder as *.cpp* files. The *main.cpp* file can be found in **src** directory which implements the complete simulation.

2.1.1 robot class

The **robot** class creates a point robot and simulates simple movement behaviors (up, down, left, right). Given a moving direction, robot tries to move in that direction by one step using **robot::move(int search_dir)** method. If it does not find any obstacle by the last taken movement, the robot updates its states using **robot::update_state** method.
Robot movement strategy: A very simple robot movement strategy has been employed in this project. A robot randomly chooses one moving direction

out of four possible directions (up, down, left, right) and then keeps moving in that direction until it is obstructed by the obstacles in the environment or by the other robots in the environment. If obstructed it randomly chooses another move direction and tries to move in that direction if no obstacles found.

2.1.2 world class

The **world** class creates the environment and consists of free-space, obstacles and point robots. The constructor accepts a two dimensional grid in the form of $vector < vector < int >>$. The **world::addRobot(int, int, char)** method adds a robot to the environment by accepting information about spawning coordinate of the robot and a name of the robot. If multiple robots have been added to the map (environment), each of them can be run autonomously using different threads by calling **world::run** method. This method takes in name of the robot and number of iterations as input argument. The **world** class has the mechanism to always keep track of the map states and can be visualized by calling **world::display** method in a separate thread. The **world::display** method takes in number of iteration as input argument.

2.1.3 pt2d class

This class creates a simple data type that can hold $2D$ coordinate (x, y). There are two methods available to access member variables, named as **pt2d::getpt_x** and **pt2d::getpt_y**. This class objects are used in conjunction with **robot::move** and **robot::state_update** methods.

3 Building and Running the Simulation

The project builds with *Cmake* and *Make*. To build the project, please follow the commands as below,

```
$    cd < path to package >
$    mkdir build  &&  cd build  &&  cmake ..  &&  make
$    . \ roboagents
```

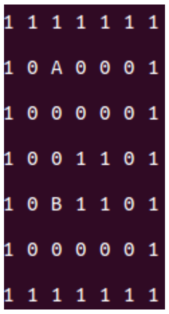
The project is tested with the following machinery,

- CXX compiler GNU 5.5.0
- C compiler GNU 5.5.0
- Linux OS : Ubuntu 16.04 LTS
- System: Intel i5 processor, 8 GB memory

The simulation is tested with 2 robots moving in a 7×7 grid world for 100 iterations. In order to properly visualize, there is a wait time of 500 miliseconds.

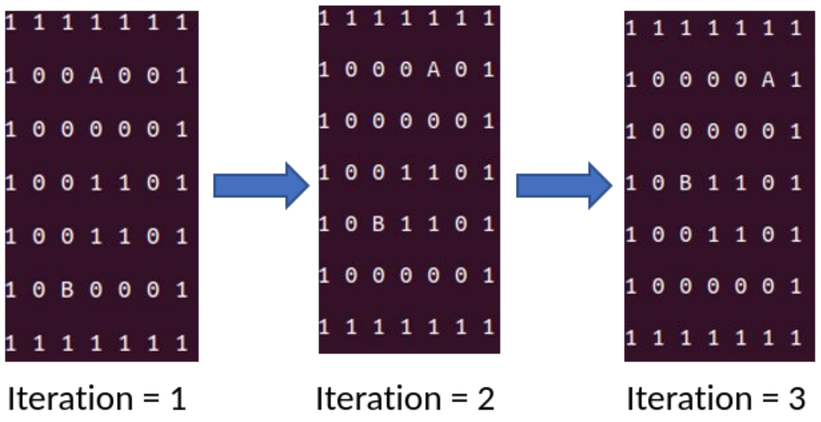
4 Expected output

The figure 1 shows initial robot location in the grid world. In figure 2 first 3 instants of map states has been presented.



```
1 1 1 1 1 1 1
1 0 A 0 0 0 1
1 0 0 0 0 0 1
1 0 0 1 1 0 1
1 0 B 1 1 0 1
1 0 0 0 0 0 1
1 1 1 1 1 1 1
```

Figure 1: Simulation console output right after robot A and B has been spawned into the world at location $(1, 2)$ and $(4, 2)$ respectively. The 1s and 0s represent obstacles and free space in the grid world



```
1 1 1 1 1 1 1
1 0 0 A 0 0 1
1 0 0 0 0 0 1
1 0 0 1 1 0 1
1 0 0 1 1 0 1
1 0 B 0 0 0 1
1 1 1 1 1 1 1
```

```
1 1 1 1 1 1 1
1 0 0 0 A 0 1
1 0 0 0 0 0 1
1 0 0 1 1 0 1
1 0 B 1 1 0 1
1 0 0 0 0 0 1
1 1 1 1 1 1 1
```

```
1 1 1 1 1 1 1
1 0 0 0 0 A 1
1 0 0 0 0 0 1
1 0 B 1 1 0 1
1 0 0 1 1 0 1
1 0 0 0 0 0 1
1 1 1 1 1 1 1
```

Figure 2: (From left) Simulation console output after iterations $1, 2, 3$ respectively. Notice that robot A moves towards right whereas robot B moves upward