CIS 32.5 Fall 2009, Project 6

1 Description

This project builds directly on the last one. Now that we can get the robot to localize, we can look at navigation. This project involves both following a plan and building a map from a plan. Since these concepts are related, I suggest that you read these instructions to the end before you start.

2 Start up

1. Prof. Parsons will give you a folder called `project6` which contains a lot of familiar-looking files.
2. The control program is in the file `navigate.cc`, so first build that:

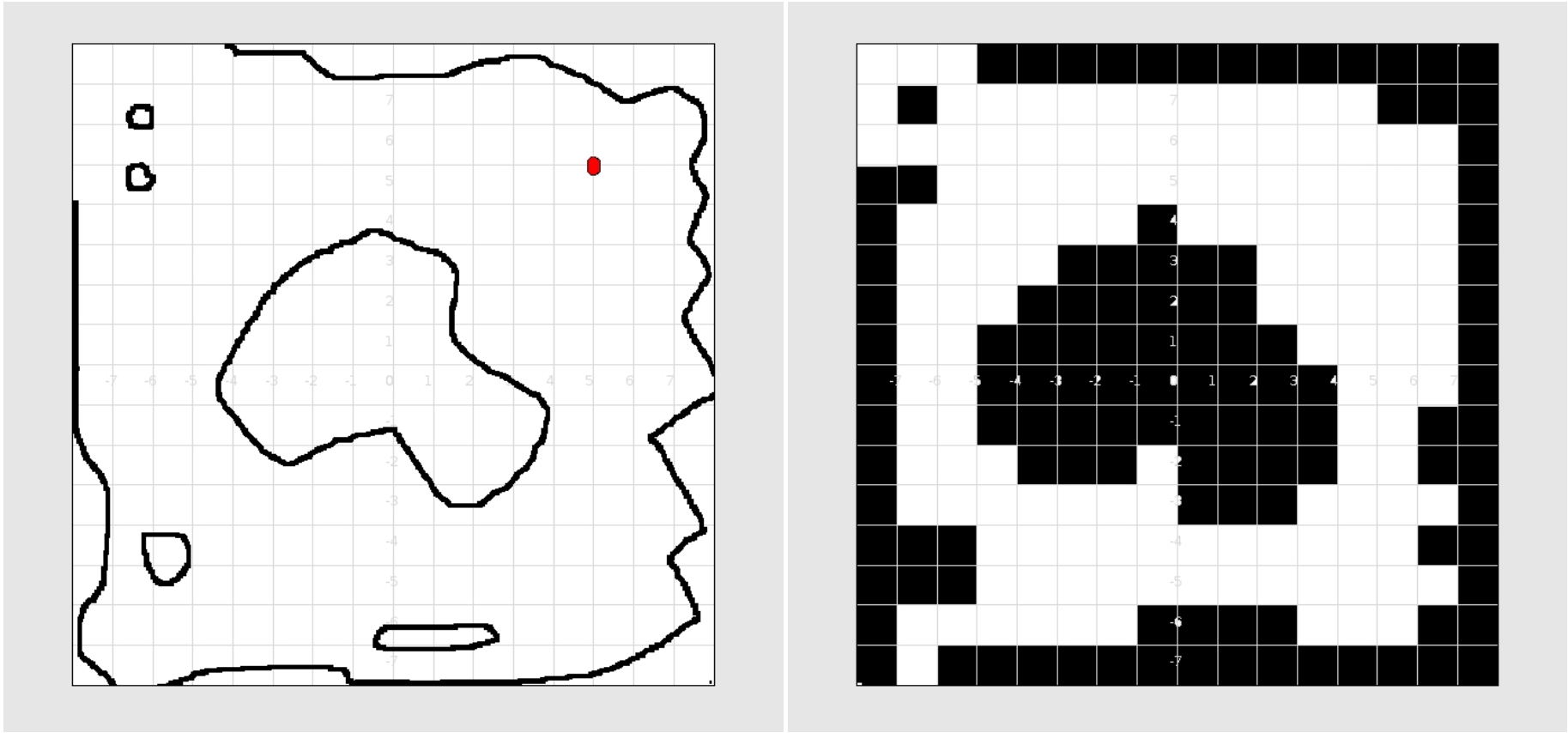
   `./build navigate`

   then run player on the config file `world6.cfg`:

   `player world6.cfg`
3. The robot, as it has at the start of previous projects, will trundle off to the south-east.
4. However, before it starts to move, it will display two things.
5. The first of these is an occupancy grid with a resolution of 0.5 meters (for the format of the grid, see below). For the purposes of the project, this is a *map*.
6. The second of these is what we'll call a *plan*

3 Map format

1. An occupancy grid map is an array of integers.
2. A 1 indicates that a square is occupied, a 0 indicates that it is not occupied.
3. The code in `navigate.cc` assumes that the map is square, so there is only one parameter that controls the size of the map.
4. In the example in the folder `project6`, each grid square is half a meter on each side.
5. This example map is for a map that is 16 meters on each side.
6. An example map is given in Figure 1(b), while Figure 1(a) shows the corresponding simulated world from Stage.
7. Note that this is NOT the world you'll be using in the project.
8. In addition, the occupancy grid in Figure 1(b) is lower resolution than the map in `navigate` — each square is one meter on each side.



(a) An example world in which the robot operates.

(b) An occupancy grid map.

Figure 1: A simulated world and the corresponding occupancy grid map

.

4 Follow a plan

1. Your first task is to take a plan that gives a sequence of locations for the robot and make the robot move through this sequence of locations.
2. The precise plan format is described in the next section.
3. The robot should stop when it gets close to the final location.
4. While the plan that you are following should have been created in such a way that the robot can move in a straight line from one point to the next without hitting an obstacle, your code will have to include code for obstacle avoidance. (You never know when a faulty map or path-planning algorithm might have been used in creating the plan that your robot is following).
5. The plan in the file `plan.txt` should be suitable for testing the robot.
6. Once your controller follows the plan, save your working program as **proj6-part1.cc**.

5 Plan format

1. A plan is an integer followed by $2n$ doubles.
2. The integer indicates how many coordinates there are in the plan (so it always has to be $2n$).
3. Each pair of doubles is a set of coordinates, and x value and a y value, in that order..
4. The coordinates in my example follow the coordinate system used by Stage. This has the origin in the center, and has coordinates ranging from -8 to $+8$.

6 Use the map to plan

1. The second task is to write a robot controller, that can use an occupancy grid to plan a route through the world represented by the occupancy grid.
2. The task assumes that you start with an occupancy grid, a set of coordinates that the robot starts at, and a set of coordinates that the robot has to end up at.
3. The task is then to generate a plan that is a sequence of sets of coordinates. The first of these sets of coordinates is the initial location of the robot, the last set of coordinates is the final location of the robot, and each intermediate point is a point that the robot should pass through on its way from the initial location to its final location.
4. You need to choose the intermediate locations carefully so that the robot can move in a straight line from one to the next without encountering an obstacle.
5. A suitable way to create the plan is to use the wildfire planner we discussed in class.
6. An example plan, that works for a robot in the world illustrated in Figure 1(b) is given in Figure 2.
7. You can use the map that came in `navigate.cc`
8. Your final program should be able to create a plan from the map and then follow the map.
9. Once your controller can do this, save your working program as **proj6-part2.cc**.

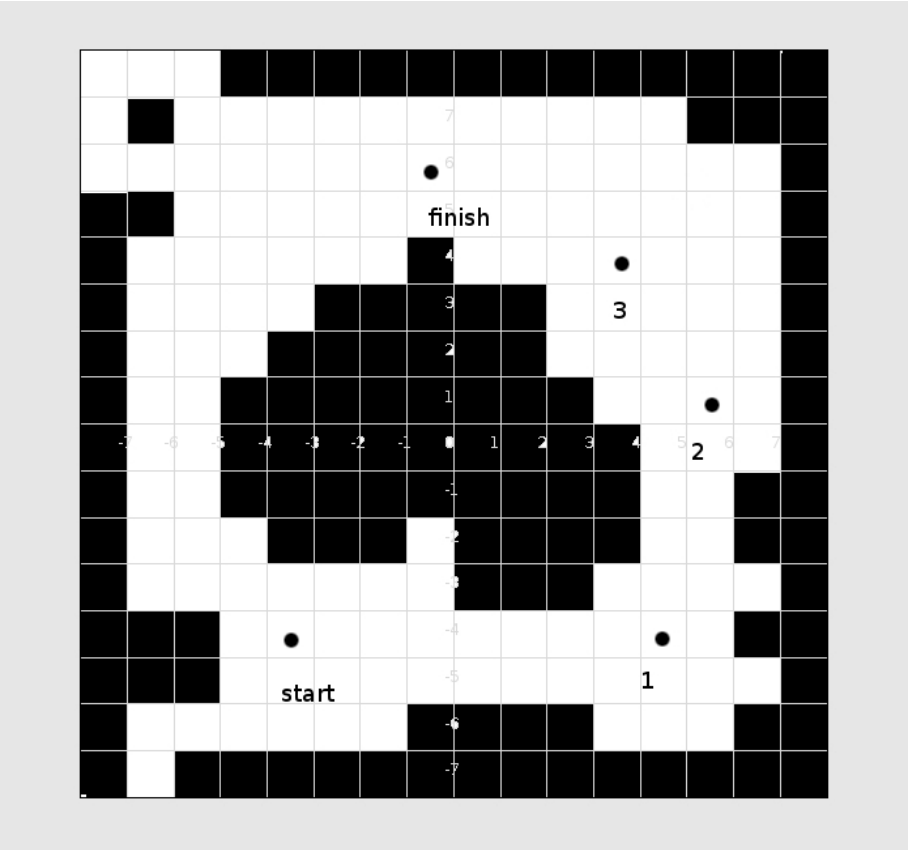


Figure 2: A plan for a robot