

Homework I:

Planning for a robot trying to catch a target

DUE: March 1st (Monday) at 11:59PM

Task

Write a planner for the robot to catch a target in a 2D grid world. At every time step, the target moves randomly in one of four directions. At every time step, the robot is allowed to make a move on an 8-connected grid.

Code

The planner function must output a single robot move. The planner should reside in *planner.cpp* file (or *robotplanner.m* file if you prefer to write it in MATLAB). Currently, the file contains a greedy planner that always moves the robot in the direction that decreases the distance in between the robot and the target. The planner function (inside *planner.cpp*) is as follows:

```
static void planner(  
    double* map,  
    int x_size,  
    int y_size,  
    int robotposeX,  
    int robotposeY,  
    int goalposeX,  
    int goalposeY,  
    char *p_actionX,  
    char *p_actionY  
)  
{
```

Inputs

The map of size x_size , y_size contains information on what are obstacles and what are not. It should be accessed as:

```
(int)map[GETMAPINDEX(x,y,x size,y size)]
```

If it is equal to 0, then the cell $\langle x,y \rangle$ is free. Otherwise, it is an obstacle. Note that all coordinates start with 1. In other words, x can range from 1 to x_size . The current robot pose is given by $\langle robotposeX, robotposeY \rangle$ and the current target pose is given by $\langle goalposeX, goalposeY \rangle$.

Outputs

The planner function should return what the robot should do next using the pointers *p_actionX* and *p_actionY*. Specifically, **p_actionX* should be set to the desired change in X (-1, 0, or 1) and **p_actionY* should be set to the desired change in Y (also, -1, 0, or 1). The robot moves on an 8-connected grid. All the moves should be valid with respect to obstacles and map boundaries (see the current planner inside *planner.cpp* for how it tests the validity of the next robot pose).

Frequency of Moves

The planner is supposed to produce the next move within 200 milliseconds. Within that 200 milliseconds, the target also makes one move. But the target can only move in four directions. If the planner takes longer than 200 milliseconds to plan, then the target will move by longer distance. In other words, if the planner takes $N \times 200$ milliseconds (where N is rounded up) to plan the next move of the robot, then the target will move by N steps in the meantime.

Execution

The directory contains few map files (map1.txt and map3.txt). Here are few examples of running the tests from within the MATLAB (in the same directory where all source files are placed):

To compile the C++ code:

```
>> mex planner.cpp
```

To run the planner

```
>> robotstart = [10 10];  
>> targetstart = [200 100];  
>> runtest('map3.txt', robotstart, targetstart);
```

Same map but more difficult to catch the target

```
>> robotstart = [250 250];  
>> targetstart = [400 400];  
>> runtest('map3.txt', robotstart, targetstart);
```

Much larger map and more difficult to catch the target

```
>> robotstart = [700 800];  
>> targetstart = [850 1700];  
>> runtest('map1.txt', robotstart, targetstart);
```

Executing *runtest* command multiple times will show that sometimes the robot does catch the target, and sometimes it does not. The letters R and T indicate the current positions of the robot and target respectively. (Sometimes, they may appear as if they are on top of a boundary of an

obstacle, but in reality they are not. The letters are just much bigger than the actual discretization of the map.)

Submission

You will submit this assignment through Gradescope. You need to upload one ZIP file named <Andrew ID>.zip. This should contain:

1. A folder named code that contains all code files, including but not limited to, the ones in the homework packet. If there are subfolders, your code should handle relative paths. We will only compile your MEX code (according to instructions in the writeup) and execute the *runtest.m* script.
2. A PDF file named <Andrew ID>.pdf. This should contain a summary of your approach for solving this homework, the results for both maps (whether the object was caught, the time taken to run the test, number of moves made by the robot, and the cost of the path traversed by the robot), and most importantly instructions for how to compile your code. This should be one line for us to execute in MATLAB of the form “mex <file 1> <file 2> ... <file N>”. We should be able to just run *mex planner.c* to compile your code from within MATLAB under Linux!
 - We want results for the above examples of *robotstart*, *targetstart*. You are free to experiment with and include results from other start positions for the robot and target.
 - For your planner summary, we want details about the algorithm you implemented, data structures used, heuristics used, any efficiency tricks, memory management details etc. Basically any information you think would help us understand what you have done and gauge the quality of your homework submission.
 - Include plots of the maps overlaid with the object and solved robot trajectories. Please **do not** include the map text files in your submission.

Grading:

The grade will depend on two things:

1. How well-founded the approach is. In other words, can it guarantee completeness (to catch a target), can it provide sub-optimality or optimality guarantees on the paths it produces, can it scale to large environments, can it plan within 200 milliseconds?
2. How fast (and how consistently) it catches the target

NOTE: To grade your homework and to evaluate the performance of your planner, we may use different and larger maps than the ones we provided in the directory, and a different strategy for how the target moves than the one in *targetplanner.m*. The only promise we make is that the target will only move in four directions and the size of the map will not be larger than 5000 by 5000 cells.