

# 16-350: Planning Techniques for Robotics

## Homework 2: High-DoF Arm Planning

Due: Mar 18 (Wed), 11:59pm

Professor: Maxim Likhachev

Spring 2020 TA: Navyata Sanghvi

### 1 Task

Write planner(s) for a planar arm to move from its start joint angles to the goal joint angles. Your planner must return a plan that is collision-free.

### 2 Code

Your code is within the folder `undergrad` or `grad` (we are confident that you know where to look). Your planner must output the entire plan, i.e., all sets of joint angles from start to goal. The planner should reside in `planner.cpp` file. Currently, the planner function contains an interpolation-based generation of a plan. That is, it just interpolates between start and goal angles and moves the arm along this interpolated trajectory. It doesn't avoid collisions. As mentioned before, your planner must return a plan that is collision-free. The planner function (inside `planner.cpp`) is as follows:

```
static void planner(  
    double *map,  
    int x_size,  
    int y_size,  
    double *armstart_anglesV_rad,  
    double *armgoal_anglesV_rad,  
    int numofDOFs,  
    double ***plan,  
    int *planlength  
)  
{ // Planner code }
```

### 3 Inputs

The map of size `(x_size, y_size)` contains information on what are obstacles and what are not. Value of cell `(x, y)` in the map can be accessed as:

```
(int)map[GETMAPINDEX(x,y,x_size,y_size)].
```

If it is equal to 0, then the cell `(x, y)` is free. Otherwise, it is an obstacle. `armstart_anglesV_rad` and `armgoal_anglesV_rad` describe the start and goal configurations of the arm respectively (angles are all clockwise from the x-axis, in radians). `numofDOFs` is the number of angles characterizing the arm configuration.

We expect that you will not have to worry about accessing map indices or understanding angle specifications, as we have provided a tool that verifies the validity of the arm configuration: this is all you would need for planning. In the starter code, you will see how this function: `IsValidArmConfiguration(angles, numofDOFs, map, x_size, y_size)` is being called to check for the validity of a configuration. Note: We do not check for self-collisions in this function, so do not worry about self-collisions.

## 4 Outputs

The planner function should return the entire plan and its length (see the current planner inside `planner.cpp` to understand how to interpret these variables).

When you run your code, you should be able to see the arm moving according to the plan you returned. If the arm intersects any obstacles, then it is an invalid plan. You might notice that the collision checker is not of very high quality and it might allow slightly brushing through the obstacles sometimes.

## 5 Execution

The `undergrad` and `grad` directories contain two map files (`map1.txt` and `map2.txt`). Following is an example of running the test from within MATLAB (in the same directory where all source files are placed).

To compile the cpp code:

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

**Undergraduate students:**

```
>> startQ = [pi/2 pi/4 pi/2 pi/4 pi/2];  
>> goalQ = [pi/8 3*pi/4 pi 0.9*pi 1.5*pi];  
>> runtest('map1.txt', startQ, goalQ);
```

**Graduate students:**

```
>> startQ = [pi/2 pi/4 pi/2 pi/4 pi/2];  
>> goalQ = [pi/8 3*pi/4 pi 0.9*pi 1.5*pi];  
>> planner_id = 0;      % change for different planners  
>> runtest('map1.txt', startQ, goalQ, planner_id);
```

## 6 Submission

You will submit this assignment through Gradescope (Entry Code: 9YJ642). You must upload one ZIP file named `<andrewID>.zip`. This should contain:

1. A folder `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 using `mex` and execute `runtest.m` in the parent code directory.

2. Your writeup in `<andrewID>.pdf`. This should contain a summary of your approach for solving this homework, results, and instructions for how to compile your code. Do not leave any details out because we will **not** assume any missing information. There 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 run the `mex` command to compile your code from within MATLAB under Linux!

#### **Undergraduate students:**

We highly recommend that you implement either RRT-Connect or PRM for this assignment, although you may choose any method with a sound approach. For 20 randomly generated start and goal pairs on `map2.txt`, you must report:

- (a) Planning times and their average
- (b) Path qualities and their average
- (c) Success rate for generating solutions within 5 seconds

Extra Credit: Solve the graduate students' assignment, fully or partially. Note that this means implementing at least two planners.

#### **Graduate students:**

You must implement four algorithms (check lines 305–309 in `planner.cpp` to get an idea of how to call your planners):

- (a) RRT
- (b) RRT-Connect
- (c) RRT\*
- (d) PRM

For 20 randomly generated start and goal pairs on `map2.txt`, and for each of the four planners, you must provide a table reporting the comparison of:

- (a) Average planning times
- (b) Average path qualities
- (c) Average number of vertices generated (in constructed graph/tree)
- (d) Success rates for generating solutions within 5 seconds

Note: You must randomly generate the start and goal pairs once and fix them for all planners.

Compile and explain your results. Justify your conclusions about which planner you think is the most suitable for the environment, why, what issues it still has, and how you think it could be improved.

Extra Credit: Present and compare additional statistics such as consistency of solutions (e.g., for different runs with similar start and goal, how much variance you observe in solution paths and qualities).

## 7 Grading

The grade will depend on:

1. Correctness of your implementations (optimizing data structures, e.g., using kd-trees for nearest neighbor search, is **not** required).
2. The speed of your solutions. We expect that you will achieve solutions within 5 seconds most of the time (for graduate students, this **must** be true for at least two of your planners).
3. Results and discussion.

**Note:** To grade your homework and to evaluate the performance of your planner, we may use a different map and/or different start and goal arm configurations. We will not test arm configurations with more than 7 degrees of freedom.