Robotics Motion Planning (CSCI-UA.0465-001); Spring 2017

Professor Chee Yap

Out: Apr 4, 2017 Due: Apr 11, 2017

Homework 4

INSTRUCTIONS: *please use the same submission instructions as Homework 1.*

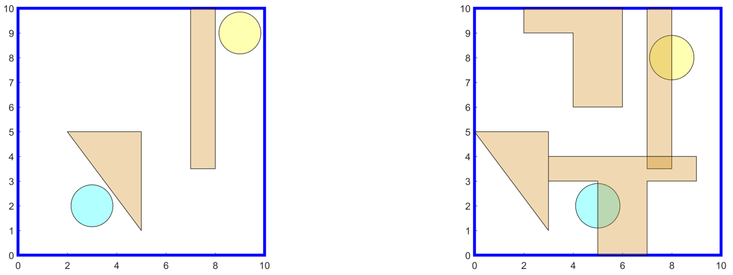
PLEASE READ THESE CODING GUIDELINES CAREFULLY

- In this homework, we want to construct the remaining elements needed for path planning of a disc robot. We will build upon some of the classes from hw2 (`Box.m, Subdiv.m, UnionFind.m`).
- Since this is a long assignment, I will allow late hw. But you will lose 5% per day. But the absolutely last day to submit is April 18.
- Please follow the following coding criteria. Up to 25% of the grades may be affected by these issues. This is very important since we are building a large project,
- Please comment every file, and most important methods. We do not suggest leaving this to the last minute because these comments can guide your programming. If we cannot understand your code, we will deduct points.
- Always pass around "objects" (points, edges, boxes) and not their individual components. In other words, use Object Oriented Programming.
- Try to break down a large method into small (re-usable) pieces that are easier to understand and debug.
- Think carefully about your choice of names for variables and methods and classes. Otherwise, your code might be incomprehensible.
- EVERY class must have a static method called `test(...)` with optional arguments. You may write more than one test if needed.

- It is a good idea to provide default arguments for methods.
- Naming conventions: Capitalize the name of classes, but use small letters for names of methods and members. Try to use "camelCase" for names (e.g., `showEnv()` instead of `show_env()`).

Q 1: (30 Points) An instance of the Path Planning Problem consists of (i) the robot description (radius $r > 0$), (ii) the resolution of search ($\epsilon > 0$), (iii) the bounding box (BB), (iv) the start and goal configurations (α, β), (v) the list of polygonal obstacles contained in BB $(P_1, P_2, \cdots)$. For simplicity, assume that the P_i are not nested, and are in CCW order.

All this information is stored in "env" files. We provide 3 samples files `env0.txt, env1.txt, env2.txt` (see Figure 1).



(a) env0.ext

(b) env2.ext

Figure 1

Please submit a file `Environment.m` which is the expansion of the provided `Environment_.m` which provides the outline of you need to implement.

We suggest using `mapshape` to represent polygons. Please make sure that when you run `Environment.test()`, you reproduce our Figure **??**. Please use this color scheme: start footprint `[0,1,1]`, goal footprint `[1,1,0]`, obstacles `[0.8, 0.5, 0]`. Set transparency (or alpha channel) using `alpha(0.3)`.

Q 2: (0 Points) We will provide Matlab files `Box1.m` and `Subdiv.m` which are just the corresponding classes from hw2, except that we now exploit Matlab's handles (read up on handles). In other words, we no longer use "registers" and "box indices". This is much cleaner and simpler.

Q 3: (50 Points) Now write new classs `Box2` and `Subdiv2` that extend the previous `Box1` and `Subdiv` classes. What is new here?

(1) First of all, we now want to maintain adjacencies and be able to obtain the neigbors of boxes. We provide `Box2_.m` and `Subdiv2_.m` with the skeleton of what you must implement.

(2) Second, we need to be able to classify boxes. Thus, the boxes must have feature sets, and each box has a `Type` which is `FREE/STUCK/MIXED`. This classification depends on the environment, and therefore, the Subdiv2 class will have an environment property.

(3) Third, we must be able to decide if two `FREE` boxes are in the same connecte component. Thus, `Subdiv2` has a UnionFind data structure.

(4) Finally, we must be able to find a path from start box to the goal box using neighboring `FREE` boxes. Thus `Subdiv2` has a `findPath` method.

We also provide the class `Indicators.m` that has special code for the compass directions (N,S,E,W) and the children indicators (NE,NW,SW,SE). YOU MUST USE THESE IN YOUR METHODS FOR ADJACENCY and NEIGHBOR. Please read up about enumeration type in Matlab. We provide two methods `dirs()` and `children()` that returns an array or directions or children indicators. Use them to do iterations (e.g., to find neighbors in each direction). In other words, you must NOT have 4 versions of your code to do each case! These ideas becomes essential when you go to (say) 3D boxes.

Q 4: (35 Points) We need a class called `SSS.m` that has the main algorithm for "Soft Subdivision Search". Fill in the various methods provided in this file.

Note that this class is the main entry point to our algorithm! You should be able to run any `env.txt` file by calling `SSS.test('env.txt')` (or use the default argument).

Q 5: (20 Points) We want to have a class called `Geom2d` that knows about points, edges, polygons. Its main methods are `sep(obj1,obj2)` to compute separation of two features, and `leftOf(p,q,r)` to compute orientation of three points. Add other methods here as needed.

It is therefore essential that `Subdiv2` is a subclass of `Geom2d` to access these functions. This means `Subdiv2` has multiple superclasses. The constructor for `Subdiv2` must call the constructor for each of its superclasses. To see how to do this, consult https://www.mathworks.com/help/matlab/matlab_oop/subclass-constructors.html