6.S078 Planning Algorithms, Fall 2014

September 3, 2014

Instructor

Tomás Lozano-Pérez, 32-G492, x3-7889

Organization

The course will be divided into three (roughly) equal parts, each focusing on one type of planning problem: robot motion planning, symbolic action planning and decision-theoretic planning. We will also highlight interactions between these types of planning problems as we go. Most of the examples will be drawn from robotics, but the applications to other areas, such as computer games, will be discussed.

The philosophy of the class is to learn by implementing. Thus, the assignments will involve implementation of the various algorithms we discuss. I will handout minimal support code (in Python), and I prefer that the assignments be done in Python. If you have a strong preference for another language, talk to me.

- The course has 6.006 as a pre-requisite. In particular, I will assume that you understand graph search algorithms well and the notion of orders of growth in algorithms. we will use make use of probability, primarily in the last third of the course, not much beyond the level of 6.01, but I will assume that you're familiar with basic notions such as conditional probability and expectation.

- There is (or will be) a Stellar web site where handouts will be posted. This will also be used for you to upload problem solutions.

- For the first part of the course, you can refer to *Planning Algorithms* by Steven M. LaValle. The full text is available on-line at `http://planning.cs.uiuc.edu`; this is linked from the Stellar site. You can buy a hardcopy from Amazon if you prefer. We will supplement the text with additional readings when appropriate.

- The book *Artificial Intelligence: A Modern Approach* by Russell and Norvig is a useful reference for the second and third part of the course.

- Grading will be based on homework (programming) assignments (40%) and 3 in-class quizzes (60%). The quizzes are tentatively scheduled for October 8, November 5 and December 3.

- You will be allowed three "late days" (72 hours), to be used in any combination, for your assignments. After that, assignment credit will go down by 20% per day.
- Anything that you hand in must be your own work – handing in work from your fellow students or from Web sources is **never** allowed. You are free to discuss approaches to homework with other students but what you hand in must be your own work.

Assignment 1 (due Wed. Sept 17 before class)

Implement a simple motion planner for a robot (modeled as a list of convex polygons) that can translate (not rotate) in the plane. One of the vertices of the polygon (e.g. the first vertex of the first polygon) should be designated as a reference point; positions of the polygon correspond to positions of that vertex. There will be a list of fixed obstacles, modeled as a list of convex polygons. The shape of the polygon and the obstacles will be user provided, do not make assumptions about them. The user will provide a start and goal position.

Assume a grid at some fixed resolution (user provided) and that the robot's position is confined to the grid intersections. The robot is constrained to move along the "lines" in the grid (horizontally and vertically). A valid solution is a list of positions corresponding to grid intersections, starting with the indicated start position and ending with the indicated goal position. The robot should not collide with any of the obstacles for any position along the path; note that there should not be collisions while moving **between** the grid points. Describe your approach to guaranteeing safety along the path.

- You will need to implement a "collision test" between two polygons at specified locations.
- You will need to implement a general graph search routine that can do (a) depth-first, (b) breadth-first, (c) shortest-path (uniform-cost), (d) A^* search (using a user-provided heuristic function).
- You should provide a capability for displaying the paths of the robot. There is a simple Python class (borrowed from 6.01), posted on Stellar, that allows you to create a window and draw on it. If you prefer to use some other display mechanism, that's fine, but no need to get carried away.

Please send email to ask for any clarifications.

You should uplaad a zip file with your code to the Stellar site. Also write a brief description of your approach in a PDF file and include some runs of your algorithm, with tables that show the length of the paths and the running time as you change the robot and the environment. Include images of the paths in some representative environments; make sure that you try a robot made up of multiple convex polygons. Experiment with all the graph-search methods, including two heuristic functions for A^* (one admissible and one not admissible but which leads to faster running time).