

Assignment 2

CSCI-4290/6290: Robot Motion Planning

Due: Wednesday, October 1, 2003, 11:59:59pm

Assignments are due before 11:59:59pm on October 1, and are to be done individually. Assignments will be graded on the basis of correctness, clarity, and code documentation. See the course syllabus for the late submission policy and academic integrity guidelines.

In this programming assignment, you are to implement the visibility graph method to find shortest paths in a planar environment with polygonal obstacles. This assignment is intended to familiarize you with writing programs that involve geometric computations.

1. The input to your program will be a set of polygons representing the obstacles, and the initial configuration and goal configuration of a point robot. The program should return the shortest path as a sequence of vertices from the initial configuration to the goal configuration.
2. Implement the construction of the visibility graph for a point robot translating in the plane. You will need to create appropriate data structures to represent the initial and goal configurations, the polygons, the visibility edges, and the visibility graph.
3. Each polygon will be specified by its vertices in counterclockwise order. You may assume all polygons are convex. The vertex coordinates of the polygons will be in an input file. Each vertex will be specified by its x and y coordinates.

Assume the initial and goal configurations are not in the interior or on the boundary of any obstacle polygon.

4. You may implement the straightforward $O(n^3)$ algorithm to determine the visibility edges of the graph, or use the rotational sweep algorithm for a more efficient $O(n^2 \log n)$ implementation.
5. Use Dijkstra's algorithm (or the A* search algorithm) to find the shortest path from the initial configuration to the goal configuration. Your program should print the length of the shortest path, and then print the shortest path as a series of (vertex) points, each point on a separate line, starting at the initial configuration and ending at the goal configuration.
6. Use the drawableObject Library (dolt) for displaying the visibility graph and the shortest path. Viewing your results graphically can greatly aid in debugging. Test your program on example cases. Source code for dolt and additional information on dolt usage will be provided on the course web page.
7. It is strongly recommended that you use C++ to implement your program. We will provide support code in C++ to read workspace and robot descriptions. Your program will be tested

on the Solaris (UNIX) workstations in the computer labs in Amos Eaton 117 and Amos Eaton 217. You should provide a README file that describes how to compile and run your program. Your code should be commented.

8. Next implement functions to generate the reduced visibility graph and display it.
9. Provide an interesting example workspace environment to demonstrate the capabilities of your planner. You can extend the capabilities of your planner here. For example, it may handle nonconvex obstacles, or handle a translating convex polygonal robot. You may use the Computational Geometry Algorithms Library (CGAL) for more involved geometric computations if required.

Submission

The code must be submitted no later than 11:59:59 pm on October 1, 2003. **You are responsible for ensuring that your code compiles and runs on the Sun Ultra10s in the OOT Lab (Amos Eaton 117) or in the Sparc Lab (Amos Eaton 217).**

You must hand in your source code (source and header files) along with a Makefile to compile it, as well files that specify your example environment. Also include a README file with the following information: your name, email address, instructions on how to compile the code and run it, an overview of your implementation and search algorithm, additional planner capabilities, known bugs or limitations, and any other relevant information.

Details of the submission procedure for your assignment will be posted on the course webpage. **Please check the course web page frequently for announcements and additional information on the assignment.**