Semester 2 2016
COMP3702/7702 ARTIFICIAL INTELLIGENCE
ASSIGNMENT 2: Motion Planning

Note:

- This assignment consists of two parts: Programming and report.
- You can do this assignment in a group of at most 3 students. This means you can also do the assignment individually.
- For those who choose to work in a group:
  - All students in the group must be enrolled in the same course code, i.e., all COMP3702 students or all COMP7702 students.
  - Please register your group name in http://goo.gl/JeyLwP before **11.59pm on Monday, 5 September 2016**. If you have not registered your group by the said time, you will need to work on the assignment individually.
  - All group members are expected to work in both programming and report. In the report, you are required to write the role of each team member.
- Submission Instruction:
  - You must write your program in either Java or C/C++ or Python. Supporting code (for checking the validity of a configuration) will be given only for Java.
  - Your program should compile from command prompt (e.g., using ant or make/cmake). It should generate an executable named a2-[courseCode]-[ID] that can be run from command prompt as:

    ```
    > [OptionalAdditionalCmd] a2-[courseCode]-[ID] inputFileName outputFileName
    ```

    [courseCode] must be replaced with 3702 or 7702, depending on which class you are enrolled in. If you work individually, please replace [ID] with your student ID. Otherwise, please replace [ID] with your group name. [OptionalAdditionalCmd] must be replaced with python or java –jar or ignored.
  - The report should be in .pdf format and named a2-[courseCode]-[ID].pdf.
  - The report and all the source codes necessary to compile your program should be placed inside a folder named a2-[courseCode]-[ID]. Please submit only source codes (i.e., remove all object files).
  - The folder should be zipped under the same name, i.e., a2-[courseCode]-[ID].zip, and the zip file should be submitted via turnitin before **11.59pm on Friday, 23 September 2016**

A wheelchair (e.g., Fig. 1) that can better assist people who are paralyzed from neck down has been developed. Currently, such a wheelchair requires manual control by the user, sometimes using their head and eye movements. UQDS would like to improve such a wheelchair by enabling it to automatically perform the more mundane movement on its own, such that the user only needs to tell where he/she wants to go or what the robot arm on the wheelchair should do. This is

Fig. 1. A wheelchair with arm from Kinova (http://www.kinovarobotics.com/assistive-robotics/products/manipulation/)

where Motion Planning technology can help, and that's why you have been hired by UQDS to do the job ☺. As a first step (i.e., this assignment), you need to develop the motion planning software for a simplified version of the problem and show how it works in simulation. The rest of this specification document details the simplification.

The system

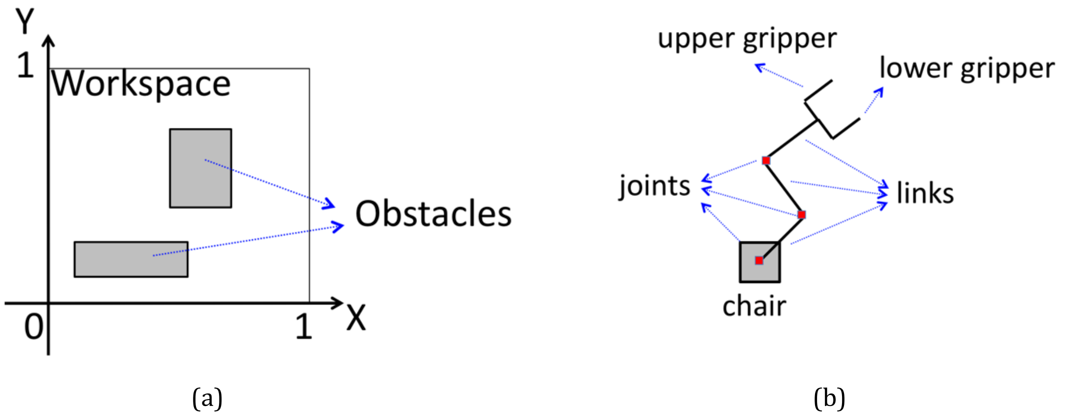

Fig. 2. (a) An example of the workspace. (b) The wheelchair and its components

The simplified wheelchair operates in a 2D workspace (rather than 3D). In particular, the 2D workspace is a plane, represented as $[0, 1]X[0, 1] \subset R^2$, and is populated by rectangular obstacles. The exact dimension and position of each obstacle in the environment is known prior to execution. Fig. 2(a) illustrates this environment.

The wheelchair may be composed of three parts: The chair (base), a robot arm, and a gripper (illustrated in Fig. 2(b)). The details of each part are:

- **The chair** can be thought of as a base for the arm (and gripper) and is a square of size 0.04X0.04 unit^2. The base can only translate. A local coordinate system is attached to this base. The origin of this coordinate system is at the center of the square, while the X and Y axis are parallel to the X and Y axis of the workspace.
- **The robot arm** forms a chain and consists of n links and n joints, where n is a non-negative integer. Each link is a line segment attached to a joint. We number the joints and links sequentially (with the index starting at 1), i.e., joint-1 is located at the center of the base, link-1 is a line-segment attached to joint-1, joint-2 lies at the other end of link-1, etc. Each joint is a rotational joint. A local coordinate system is attached to each joint. The origin of this coordinate system is the position of the joint. For joint-1, this coordinate system coincides with the coordinate system of the base. For other joints, the X axis is the line that coincides with the previous link. Each joint is a rotational joint, which means it can only rotate. We define the joint angle of joint-i as the angle between link-i and the X axis of the coordinate system attached to joint-i. The joint angle of each joint is limited to be within $[-150^0, 150^0]$. Fig. 3 illustrates the rotational joints.

Each link is of size 0.05 unit length.

- **The gripper** consists of 2 L shape segments: upper and lower gripper, as illustrated in Fig. 3. Each gripper is attached to the last link of the arm and has a fixed relative orientation with respect to the last link of the arm. However, the length of the segments can be altered and is defined as (u_1, u_2) for the upper gripper and as (l_1, l_2) for the lower gripper, where u_1, u_2, l_1, l_2 $\in$ [0.03, 0.07].

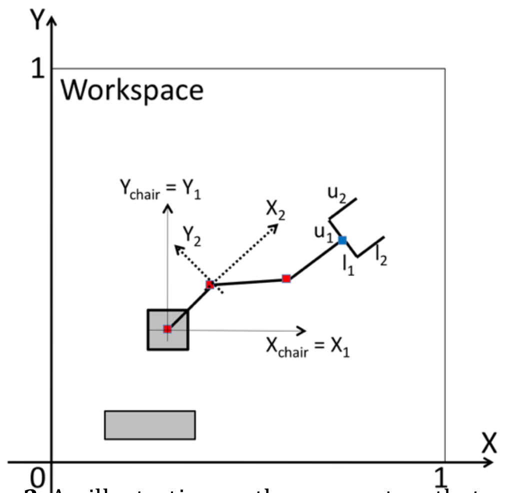

Fig. 3. An illustration on the parameters that define the robot's configuration.

What your program should do

Given the initial and goal configurations of the wheelchair, as well as a map of the environment, your program must find a valid path from the initial to the goal configurations. A valid path means that when the wheelchair executes the path, it will satisfy the following requirements:

1. The path consists of primitive steps. In each primitive step, the chair cannot move more than 0.001 unit, each joint angle cannot move more than 0.1^0, and the length of a segment in the gripper cannot move more 0.001 unit.
2. It will not collide with any of the obstacles.
3. It will not collide with itself.
4. The entire wheelchair must lie inside the workspace.
5. The joint angles (for the arm) and the segment length (for the gripper) must be inside their respective lower and upper limit (i.e. [-150^0, 150^0] as described in the previous section and [0.03, 0.07] for the segment length).
6. Since a primitive step is very small, it is sufficient to satisfy requirements 2-5 at the beginning and end of each primitive step.

Input and Output format

The input is a .txt file. To describe the format of this file, we first need to describe the format of a configuration.

Format of a configuration: A configuration is represented by n real numbers, where n is the dimension of the C-space. Each number is separated by a white space. The first two numbers are the position of the origin of the chair's coordinate system in the workspace. If the wheelchair does not have a gripper, the last n-2 numbers are the joint angles in sequential order (i.e., the third number is the joint angle of joint-1, the fourth number is the joint angle of joint-2, etc.). Each joint angle is defined in radian. If the wheelchair has a gripper, the subsequent n-6 numbers are the joint angles in sequential order, while the last 4 numbers are the values of u_1, u_2, l_1, l_2, respectively.

Input format. The program you develop should accept an input file. The file contains the type of wheelchair, the initial and goal configurations, and the obstacles position and dimension. The format of the input file is as follows.

1. The file consists of k + 4 lines, where k is the number of obstacles in the environment.
2. The first line is the type of wheelchair. There is only two possibilities, i.e., withGripper and noGripper for a wheelchair with a gripper and that without a gripper, respectively.
3. The second line is the initial configuration.
4. The third line is the goal configuration.
5. The fourth line is the number of obstacles in the environment.
6. Each line in the next k lines represents an obstacle and consists of 4 real numbers. The first two numbers represent the X and Y position of the upper-left vertex of the rectangle, while the last two represent the X and Y position of the lower-right vertex of the rectangle.

Output format. Your program should output the wheelchair's path to a file with the following format.

1. The file consists of m+2 lines, where m is the number of primitive steps in your path.
2. The first line is the number of line-segments.
3. The second line is the initial configuration.
4. The next m lines are the end configuration of each primitive step.

Examples of the input and output files are in the accompanying supporting software.

Grading for the Programming Part (total points: 60/100)

For marking, we will use 3 different classes of scenarios (and queries):

1. The wheelchair consists of only the chair. This class has 4 scenarios.
2. The wheelchair consists of the chair and the arm. This class has 4 scenarios for COMP3702 and 6 scenarios for COMP7702.
3. The wheelchair consists of all components, i.e., the chair, arm, and grippers. In this class of scenarios, for COMP3702, the arm in this type of scenario has 4 joints or more and there will be 5 scenarios, while for COMP7702, the arm has at least 6 joints and there will be 6 scenarios.

If you use sampling-based method, we will run your program 10X for each query. Your program is deemed as solving the query within the given time limit if it solves at least 5 out of 10 runs within the given time limit. Your program is deemed as solving the query within 1-2X the given time limit if it solves at least 5 out of 10 runs within 1-2X the given time limit.

The details of the grading scheme is as follows. If your situation satisfies more than one marking band, we will use the higher band.

COMP3702:

- >= 1 & < 10: The program does not compile nor run.

- >= 10 & < 20: The program runs but fails to solve any query. The program fails to solve a query when it does not find a valid path after more than 2X the given time limit.
- >= 20 & < 30: The program solves at least one of the queries within 1-2X the given time limit.
- >=30 & < 40: The program solves at least one of the queries within the given time limit when the wheelchair consists of only the chair. Each query is worth 2.5 points.
- >=40 & < 50: The program solves at least one of the queries within the given time limit when the wheelchair consists of the chair and the arm. Each query is worth 2.5 points.
- >=50 & <= 60: The program solves at least one of the queries within the given time limit when the wheelchair consists of all three components (i.e., the chair, arm, and gripper). The arm in this class of scenarios will have at least 4 joints. Each query is worth 2.5 points.

COMP7702:

- >= 1 & < 5: The program does not compile nor run.
- >= 5 & < 10: The program runs but fails to solve any query. A program fails to solve a query when it does not find a valid path after more than 2X the given time limit.
- >= 10 & < 20: The program solves at least one of the queries within 1-2X the given time limit.
- >=20 & < 30: The program solves at least one of the queries within the given time limit when the wheelchair consists of only the chair. Each query is worth 2.5 points.
- >=30 & < 45: The program solves at least one of the queries within the given time limit when the wheelchair consists of the chair and the arm. Each query is worth 2.5 points.
- >=45 & <= 60: The program solves at least one of the queries within the given time limit when the wheelchair consists of all three components (i.e., the chair, arm, and gripper). The arm in this class of scenarios will have at least 6 joints. Each query is worth 2.5 points.

Report (total points: 40/100)

Your report must contain answers to the following questions:

1. [5 points] Please define the Configuration Space of the problem and describe which method for searching in continuous space do you use.
2. [10 points] If you use sampling-based method, please describe the strategy you apply or develop for each of its four components. Otherwise, please describe the details of your discretization method.
3. [12.5 points] Which class of scenarios do you think your program will be able to solve? Please explain your answer.
4. [12.5 points] Under what situation do you think your program will fail? Please explain your answer.

Please note that in each of the above question, when explanation is requested, the explanation part is 80% of the total points. For the explanation part, you will get higher mark for stronger argument. A strong argument should at least include a logical explanation of why and include experimental results to back your explanation. Please also note that good explanation is NOT equal to long explanation!!!

oOo That's All, Folks oOo