COMP3702/COMP7702 Artificial Intelligence (Semester 2, 2020)
Assignment 2: Continuous motion planning in CANADARM

Key information:

- **Due: 5pm, Friday 25 September**
- This assignment will assess your skills in developing continuous motion planning problems.
- Assignment 2 contributes 15% to your final grade.
- This assignment consists of two parts: (1) programming and (2) a report.
- This is an individual assignment.
- Both code and report are to be submitted via Gradescope (`https://www.gradescope.com/`). You can find instructions on how to register for the COMP3702/COMP7702 Gradescope site on Blackboard.
- Your program (Part 1, 60%) will be graded using the Gradescope code autograder, using the testcases in the support code provided at `https://gitlab.com/3702-2020/assignment-2-support-code`.
- Your report (Part 2, 40%) should fit the template provided, be in .pdf format and named according to the format `a2-[courseCode]-[SID].pdf`. Reports will be graded by the teaching team.

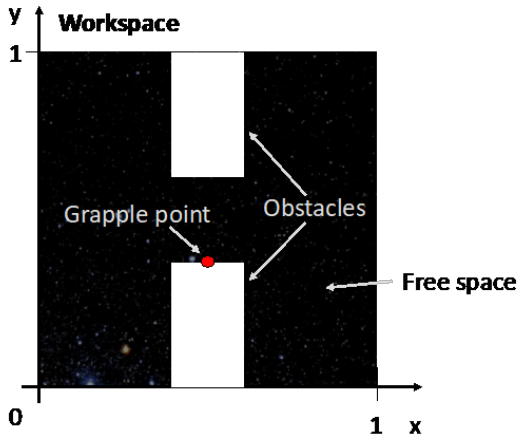
The CANADARM2 motion planning environment

CANADARM2 is a remote-controlled mechanical arm aboard the International Space Station (ISS), see Figure 1. The robotic arm is used to deploy, capture and repair satellites, position astronauts, maintain equipment, and move cargo. The rest of this specification document details the system and task.

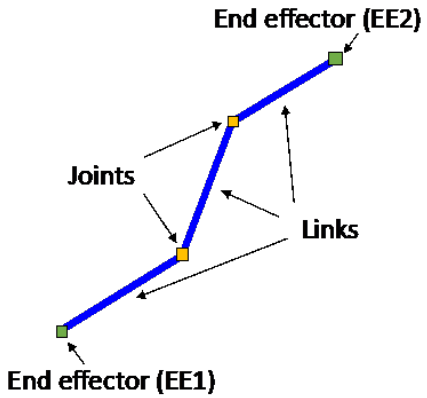
The system: The simplified Canadarm operates in a 2D workspace (rather than 3D). In particular, the 2D workspace is a plane, represented as $[0, 1] \times [0, 1] \subset \mathbb{R}^2$, and is populated by rectangular obstacles. In addition, there are grapple point(s) which the end effectors of the robotic arm can attach to. One of the end effectors must be grappled to a grapple point at any time. The exact dimensions and positions of each obstacle and the number and position of the grapple points in the environment are known prior to execution. Figure 2a illustrates this environment.

The robotic arm is composed of x links and x joints, where x is a non-negative integer, with two end effectors EE1 and EE2, which can attach onto grapple points. An example robotic arm with 3 links is shown in Figure 2b. Each link of the robot is a line segment attached to a joint. The link connected to the grappled

Figure 1: Canadarm2 with astronaut Stephen K. Robinson. Source: NASA



(a) An example of the workspace



(b) The robot arm and its components

Figure 2

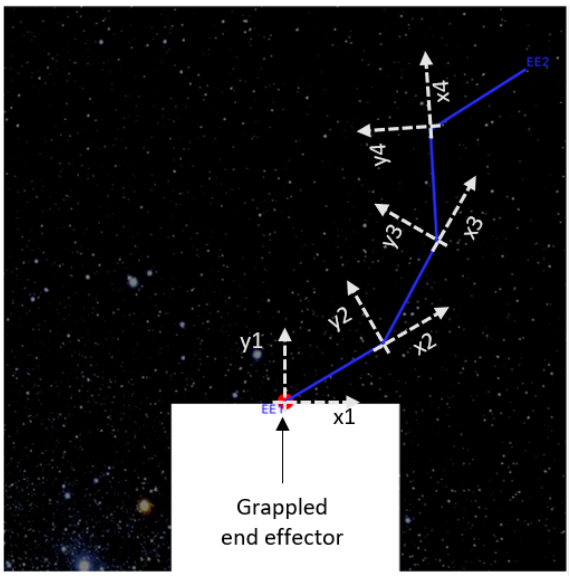


Figure 3: An illustration of the co-ordinate scheme of each joint on a 4-segment robotic arm.

end effector acts as a joint. Each joint is a rotational joint which means it can only rotate. A local coordinate system is attached to each joint. The co-ordinate system of the joint at the location of the grappled end effector, coincides with the coordinate system of the end effector. For the remaining joints, the x-axis is the line that coincides with the previous link. We define the angle of segment-i as the angle between link-i and the X axis of the coordinate system attached to joint-i. The joints are numbered relative to the grappled end-effector. Figure 3 illustrates the rotational joints. In some tasks, the links are telescopic and can change length (i.e. when the specified min and max segment lengths differ). This allows the robotic arm to more easily reach the grapple points.

Motion planning for CANADARM

This section describes what your program should do. Given the initial and goal configurations of the Canadarm robotic arm, 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 Canadarm executes the path, it will satisfy the following requirements:

1. The path consists of primitive steps. In each primitive step, each joint of the robot arm cannot move more than 0.001 units (i.e. radians or arm length).

2. It will not collide with any of the obstacles
3. It will not collide with itself
4. The entire Canadarm robotic arm must lie inside the workspace
5. The angle between adjacent arm segments cannot be tighter than 15 degrees (i.e. angles 2, 3, 4... must be between -165° and +165°).
6. The segment lengths must be within the bounds specified in the input file (i.e. within min and max lengths)
7. Since a primitive step is very small, it is sufficient to satisfy requirements 2-4 at the beginning and end of each primitive step.

What is provided to you

We will provide supporting code in Python only, in the form of:

1. Support code to represent the problem (`problem_spec.py`, `robot_config.py`, `obstacle.py`, `angle.py`)
2. A visualiser
3. A tester
4. Test cases to test and evaluate your solution

The support code can be found at: `https://gitlab.com/3702-2020/assignment-2-support-code`. Autograding of code will be done through Gradescope, so that you can test your submission and continue to improve it based on this feedback — you are strongly encouraged to make use of this feedback.

Your assignment task

You have been hired to develop Motion Planning software for a simplified version of the robotic arm. You will be graded on both your submitted **program (Part 1, 60%)** and the **report (Part 2, 40%)**. These percentages will be scaled to the 15% course weighting for this assessment item.

Part 1 — The programming task

Your program will be graded using the Gradescope autograder, using the COMP3702 testcases in the support code provided at `https://gitlab.com/3702-2020/assignment-2-support-code`.

Interaction with the testcases and autograder

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. See `example_output.txt` in the support code for an example of how this is formatted. Each number is separated by a space, and each group of numbers is separated by a semicolon:

- The first two numbers are the position of the origin of the grappled end effector in the workspace.
- The next x numbers are the joint angles (ordered relative to the grappled end effector), with each joint angle defined in degrees.
- The last x numbers are the lengths of each link (line segment) of the robot ordered relative to end effector 1.

Input format: The program you develop should accept an input file. The file contains information on the set-up of the robotic arm, the initial configuration, the goal configuration, and the environment. This input file uses comments (marked with '#') to label which values correspond to which parameters. The input parameters are as follows:

```
----------
# Robotic arm details
number of segments
min lengths for each segment (order from ee1)
max lengths for each segment (order from ee1)

# Initial Configuration
initial grappled ee (either 1 or 2, representing ee1 or ee2)
initial grappled eex, eey
initial angles from grappled ee +ve axis (degrees)
initial segment lengths from ee1

# Goal Configuration
goal grappled ee (either 1 or 2, representing ee1 or ee2)
goal grappled eex, eey
goal angles from grappled ee +ve axis (degrees)
goal segment lengths from ee1

# Environment
number of grapple points
x, y for each grapple point

number of obstacles
x1, y1, x2, y2 for each obstacle
----------
```

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

1. The file consists of m+1 lines, where m is the number of primitive steps in your path.
2. The first line is the initial configuration.
3. The next m lines are the end configuration of each primitive step, where the final line should correspond to the goal configuration.

Examples of the input and output files are in the accompanying support code.

Grading rubric for the programming component (total marks: 60/100)

For marking, we will use testcase scenarios and queries of varying difficulty levels to evaluate your solution:

Level 1: at most 3 arm segments, all with fixed length, 1 grapple point, wide gaps between obstacles

Level 2: at most 4 arm segments, some with telescoping length, 1 grapple point, narrower gaps between obstacles

Level 3: at most 4 arm segments, all with telescoping length, 2-3 grapple points, narrower gaps between obstacles

Level 4: at most 5 arm segments, all with telescoping length, 3+ grapple points, tight gaps between obstacles

Marks will be allocated according to the following rules:

- The program fails to solve a query when it does not find a valid path in the given time limit.
- If you use a sampling-based method, we will run your program up to $3\times$ for each query. Your program is deemed as solving the query if it solves at least 1 out of 3 runs within the given time limit.
- The time limit is 1 minute per query for levels 1 and 2, and 2 minutes for the higher-level test cases.

COMP3702	
Marks	*Performance threshold*
	The following are marks for submission that do not solve any testcases within the time limit
<10	The program does not run (marks at manual grader's discretion).
15	The program runs but fails to solve any testcase within $2\times$ the given time limit.
20	The program solves one of the testcases within 1-2$\times$ the given time limit.
25	The program solves more than one of the testcases within 1-2$\times$ the time limit.
30	The program solves one testcase.
35	The program solves 2 testcases.
40	The program solves 3 testcases.
45	The program solves 4 testcases.
50	The program solves 5 testcases.
55	The program solves 6 testcases.
60	The program solves at least 7 of the 8 testcases.

COMP7702	
Marks	*Performance threshold*
<10	The program does not run (marks at manual grader's discretion).
10	The program runs but fails to approximately solve any testcases within $2\times$ the given time limit.
15	The program solves one of the testcases within 1-2$\times$ the time limit.
20	The program solves more than one of the testcases within 1-2$\times$ the time limit.
25	The program solves one testcase.
30	The program solves 2 testcases.
35	The program solves 3 testcases.
40	The program solves 4 testcases.
45	The program solves 5 testcases.
50	The program solves 6 testcases.
55	The program solves 7 testcases.
60	The program solves all 8 testcases.

Part 2 — The report

The report tests your understanding of the methods you have used in your code, and contributes 40/100 of your assignment marks. **Please make use of the report templates provided on Blackboard**, because Gradescope makes use of a predefined assignment template. Submit your report via Gradescope, in .pdf format (i.e. use "save as pdf" or "print-to-file" functionality), and named according to the format `a2-[courseCode]-[SID].pdf`. Reports will be graded by the teaching team.

Your report task is to answer the questions below:

Question 1. **(5 marks)**
Please define the Configuration Space of the problem and describe which method you use for searching in continuous space.

Question 2. **(20 marks)**
If you used a sampling-based method, describe the strategy you applied to develop each of its components (sampling strategy, connection strategy, checking if a configuration is valid or not, checking if a line segment in C-space is valid or not). Otherwise, please describe the equivalent details of your discretization method.

Question 3. **(15 marks)**
Which class of scenarios do you think your program will be able to solve and under what situation(s) do you think your program will fail? Please explain your answers.

Academic Misconduct

The University defines Academic Misconduct as involving "a range of unethical behaviours that are designed to give a student an unfair and unearned advantage over their peers." UQ takes Academic Misconduct very seriously and any suspected cases will be investigated through the University's standard policy (`https://ppl.app.uq.edu.au/content/3.60.04-student-integrity-and-misconduct`). If you are found guilty, you may be expelled from the University with no award.

It is the responsibility of the student to ensure that you understand what constitutes Academic Misconduct and to ensure that you do not break the rules. If you are unclear about what is required, please ask.

It is also the responsibility of the student to take reasonable precautions to guard against unauthorised access by others to his/her work, however stored in whatever format, both before and after assessment.

In the coding part of this assignment, you are allowed to draw on publicly-accessible resources, but you must make reference or attribution to its source, by doing the following:

- All blocks of code that you take from public sources must be referenced in adjacent comments in your code.
- Please also include a list of references indicating code you have drawn on in your `solution.py` docstring.

However, you must not show your code to, or share your code with, any other student under any circumstances. You must not post your code to public discussion forums (including Piazza) or save your code in publicly accessible repositories (check your security settings). You must not look at or copy code from any other student.

All submitted files (code and report) will be subject to electronic plagiarism detection and misconduct proceedings will be instituted against students where plagiarism or collusion is suspected. The electronic plagiarism detection can detect similarities in code structure even if comments, variable names, formatting etc. are modified. If you collude to develop your code or answer your report questions, you will be caught.

For more information, please consult the following University web pages:

- Information regarding Academic Integrity and Misconduct:
  - https://my.uq.edu.au/information-and-services/manage-my-program/student-integrity-and-conduct/academic-integrity-and-student-conduct
  - http://ppl.app.uq.edu.au/content/3.60.04-student-integrity-and-misconduct
- Information on Student Services:
  - https://www.uq.edu.au/student-services/

Late submission

Students should not leave assignment preparation until the last minute and must plan their workloads to meet advertised or notified deadlines. It is your responsibility to manage your time effectively.

Late submission of the assignment will **not** be accepted. Unless advised, assessment items received after the due date will receive a zero mark unless you have been approved to submit the assessment item after the due date.

In the event of exceptional circumstances, you may submit a request for an extension. You can find guidelines on acceptable reasons for an extension here https://my.uq.edu.au/information-and-services/manage-my-program/exams-and-assessment/applying-extension All requests for extension must be submitted on the UQ Application for Extension of Progressive Assessment form at least 48 hours prior to the submission deadline.