MTRN4230 Lab 08

1. Aim

The aim of this lab session is to use the RVC toolbox to apply a path planner to solve a 2-link robot planning problem.

2. Pre-lab

There are no pre-lab activities required for this week's lab. Project-1 is due Friday at 17:00. If you need to collect data from the robot for part C of the assessment task, come pre-pared to the lab with the required program.

3. Lab Activities

During this lab, you will use the RVC Toolbox to create a 2 link serial manipulator, perform joint space to cartesian space conversion and compare several path planning algorithms.

Part 1

The code for Part 1 has been provided on moodle, but instructions to replicate the functionality of it follows for those interested.

Create the Arm

In order to perform rudimentary collision detection, we will create a two link serial manipulator and treat it as having 2 bodies. We will check for collision in the first link and then we will check for collision with the second link. As the RVC toolbox does not provide any collision detection functionality this basic method will be applied.

1. Create a one-link serial manipulator with the name "one_link" following DH table

j	theta	d	a	alpha	offset
1	q	0	0.5	0	0

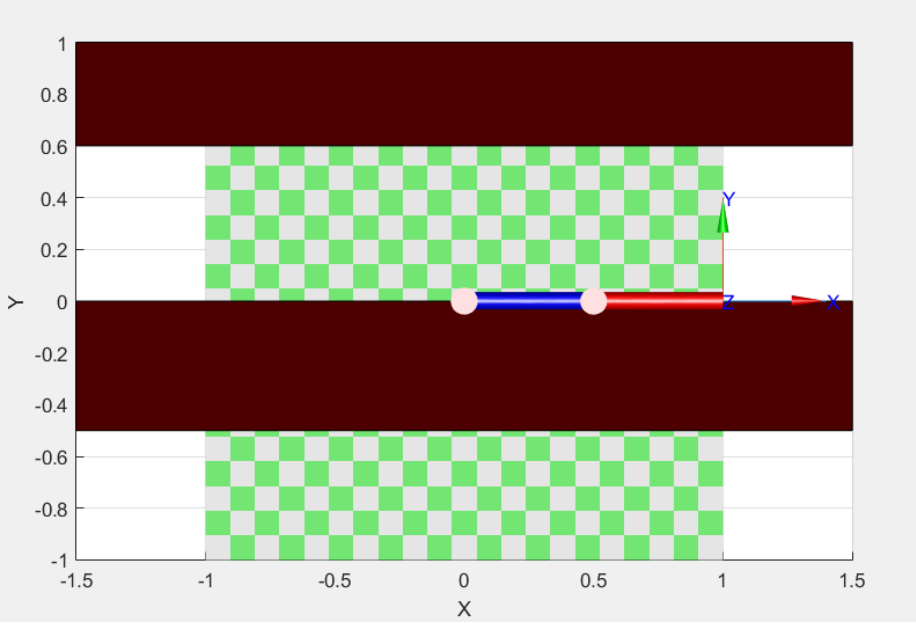
2. Create a two-link serial manipulator with the name "two_link" following DH table:

j	theta	d	a	alpha	offset
1	q1	0	0.5	0	0
2	q2	0	0.5	0	0

Create Collision Planes

Assume that there is a fixed obstacle at y = 0m and y = 0.6m like a horizontal surface below which the manipulator is not permitted to reach.

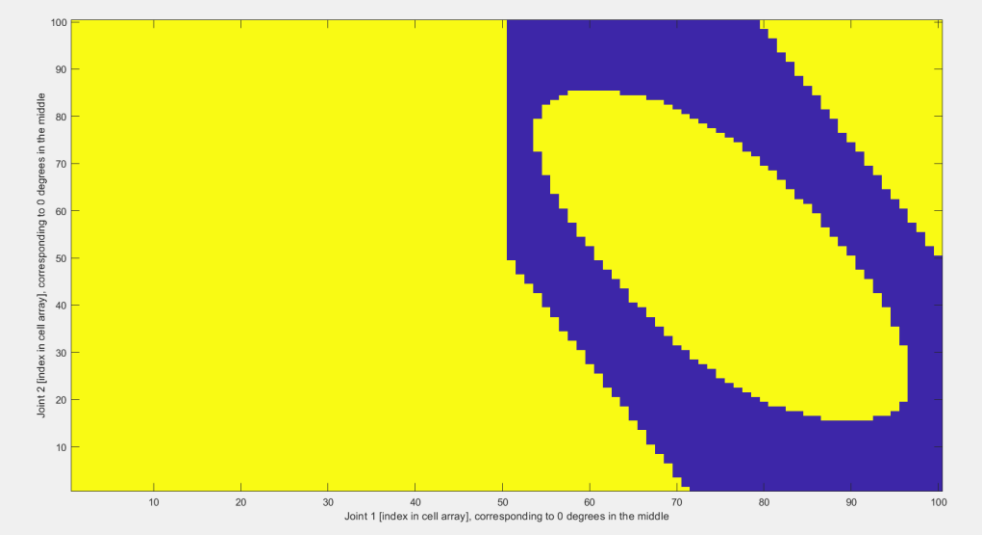
In the "RTB manual" at this webpage, look up the "plot_box" function, and use it to plot 2 separate box to emulate the planes that the robot cannot pass through. Your result should look similar to the following:



(Hint: You may want to use 'hold on', before plotting the planes with the "plot_box" and the robotic arm at a joint angle of [0 0])

Define the Configuration Space

The joint space has been discretised into 100 x 100 cells for computation and has been plotted for you. It shows the free (blue) and occupied (yellow) regions of the joint space.



If you are attempting to implement this yourself joint 1 can rotate from [-180 to +180] while joint 2 can only rotate from [-150 to +150]. You will need to determine the precision required to discretise these joint rotations to 100 steps. For each discretised joint angle position, you will need to calculate it's forward kinematics to determine the [x,y] coordinate of the one_link and two_link manipulators. You

will then need to determine whether or not this [x,y] coordinate is colliding with a plane or not. The function angle2cell has been provided to convert an angle into its cell angles minimum, maximum angles, and number of cells.

In general, the collision detection should be applied to the entire body of the robotic arm. Since the RVC toolbox does not provide this functionality, we will only be looking at the position of the first link and the end effector to determine if robotic arm is colliding with something.

Part 2

This part will focus on comparisons between different path planning algorithms.

In the provided code, the C-space has been saved in the "config_space_binary" parameter. The configuration space is a 100 x 100 matrix. Joint 1 can rotate from [-180 to +180] while joint 2 can only rotate from [-150 to +150].

PRM

Using the "RTB Manual" look up "PRM" to understand how to implement the PRM path planner.

For a starting pose of $[\theta_1, \theta_2]$ = [5,5] and a goal post of [175, -5] generate the path of the end effector and show an animation.

You will need to use the provided cell2angle function which converts a cell value into an angle, given the minimum, maximum angles and the number of cells.

You will need to convert the starting and goal pose into a cell value to enter into the path planning algorithm. Once the path planning algorithm has finished, you will need to convert all the cell values back into joint angles to allow the robot to follow the path.

D*

Using the "RTB Manual" look up "Dstar" to understand how to implement the Dstar path planner. For the same starting and goal pose generate the path of the end effector and show an animation.

You are welcome to play around with other path planning algorithms as well!

1. Describe and discuss the difference between the paths generated in terms of the algorithms used.

4. Post-lab

- Project-1 is due this Friday at 17:00.
- Project-2 will also be released this week. That is due Friday week 12 at 17:00.