Arm Lab 2020 (Team Assignment)

16-311: Introduction to Robotics

Contents

Learning Objectives

1. Utilize kinematics, and knowledge of path planning principles to generate a configuration space for a two-link RR robot.

2. Utilize forward kinematics, inverse kinematics, controls, and path planning to safely navigate a known workspace with simulated forces like gravity.

Introduction

0.1 Version Control

This is a team lab. You will work in your group of three from earlier in the semester. Most of the work is in code, so we recommend using GitHub or some other way to share your code with your teammates and keep track of the different versions. Here are some GitHub resources:

- Basic tutorial: `https://guides.github.com/activities/hello-world/`
- Official learning lab: `https://lab.github.com/githubtraining/introduction-to-github?overlay=register-box-overlay`
- Great visualization tool to understand the Git timeline and branching: `https://try.github.io/`
- Clear and brief intro: `http://rogerdudler.github.io/git-guide/`
- Fun app to play and practice with: `https://github.com/jlord/git-it-electron`

0.2 Starter Code

Here is starter code for the lab: `https://drive.google.com/drive/folders/1PGo6I1-mJi6ws0__DPHdkv6YW62JsA3n?usp=sharing`. We are giving it all to you now so you can work ahead if you choose. You do not have to use the starter code Week 1. You must use it for Week 2 so we can see how your controller works in the simulated environment.

The starter code uses gym for the simulation (which you will need for Week 2). You should have gym installed from Homework 4, but here are some instructions in case you are on a new computer: `http://gym.openai.com/docs/`.

Provided files:

- rendering.py: you should not need to edit this. This file is used by arm.pyc in order to produce the visualization.
- arm.pyc: you cannot edit this. This is what will create your simulation. The link lengths, link weights, and motor limits are encoded here.
- main.py: this is where you will add your code. This file simply sets up a loop to perform the simulation. You are welcome to edit this file as you see fit, as long as it calls the arm rendering for Part 2.

You cannot use libraries that would trivialize this assignment (forward kinematics, invers kinematics, path planning, controls). However, you can use code that you wrote for previous labs in this class.

0.3 Lab Problem Statement

In this lab you will create a configuration space, plan a path, and execute a path for a two-link robot. This robot has two revolute joints that are parallel to the ground as in the image below.

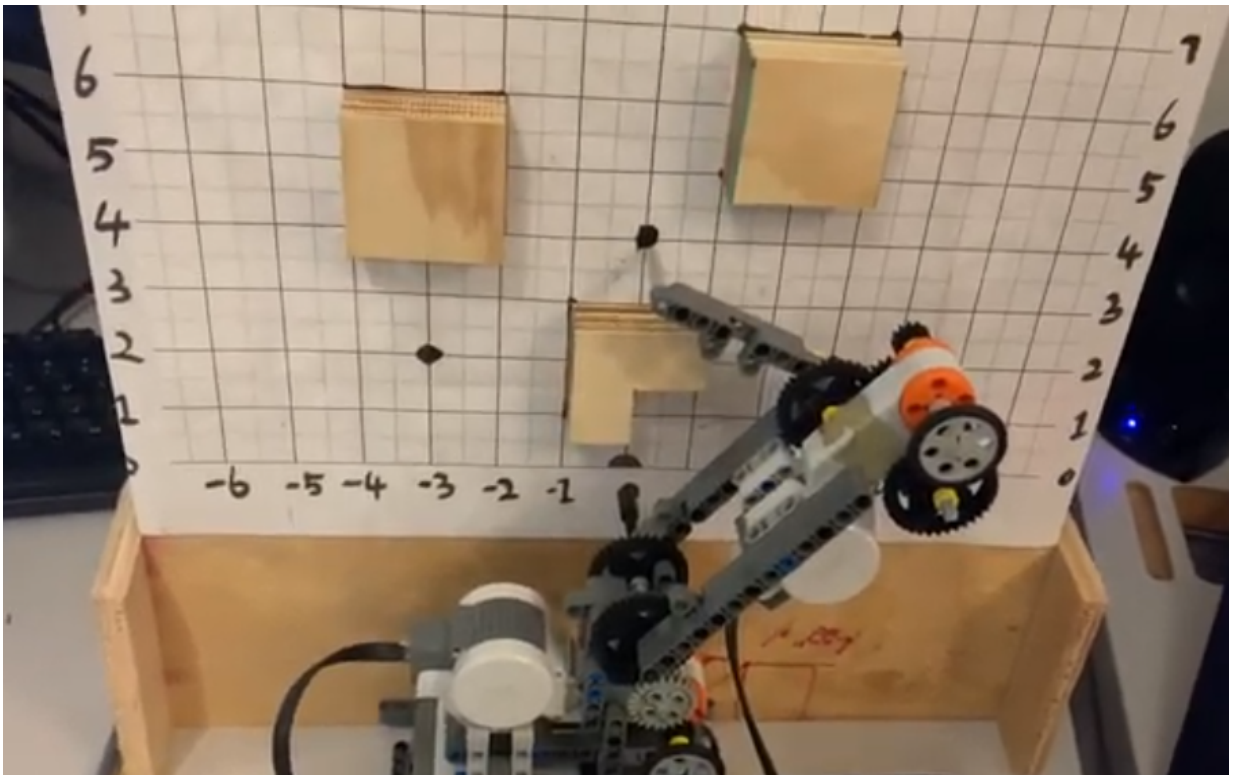


Figure 1: Example robot from 2019 used with permission. Note how the end effector could hit obstacles if a bad path was planned but the rest of the arm is out of the plane of the obstacles. For this lab that is allowed.

0.4 Suggested Workflow

Week 1: To create the configuration space, we recommend stepping through θ_1 and θ_2s and determining if that combination will cause the end effector of reasonable

dimension to intersect an obstacle using forward kinematics. You will need some way of visualizing this configuration space, you can reuse code that you wrote for Lab 5. Some visualization tools include tkinter `https://docs.python.org/3/library/tkinter.html` and Pygame `https://www.pygame.org/wiki/GettingStarted`.

Week 2: Just like in Lab 5, you will need to plan a path in a known environment. You should first decide if you want to represent your configuration space as a grid or waypoints for path planning. Pick and implement a data structure that will be able to store this information. Then choose a path planning algorithm and implement it. For this lab, there is the additional step of converting the start and endpoints from the (x,y) to (θ_1,θ_2) using inverse kinematics.

Once you have your path, you will send commands to the arm simulator and you will see the actions in the visualization. Since you are sending commands to the simulator in terms of torque but you actually care about angles, you will need to implement some kind of control. We recommend looking back at your implementation of PID in Homework 3 and Lab 4. Since you are combating a constant force (gravity), this may be a good time to use feed forward. However, you likely do not want a constant feed forward term since you do not want to add the same "nudge" when the arm is completely vertical as you would if the arm is completely out to the side.

1 Part 1 (Week 1)

1.1 Written [20 points]

Your simulated robot has its first joint at the origin, first link 3.75 inches, second joint with no joint limits (wrap around permitted), and second link 2.5 inches long.

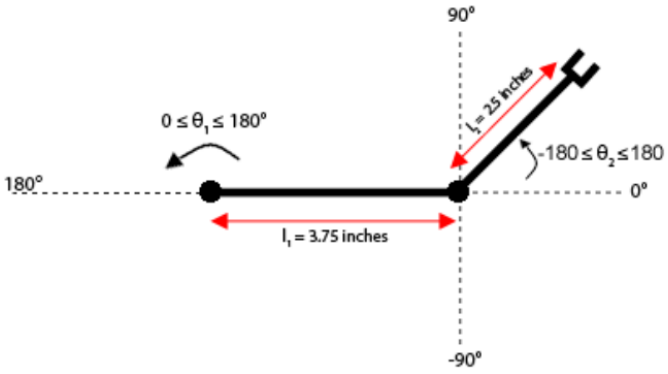


Figure 2: Workspace in the XY.

1. Imagine that you are designing such an arm using two motors (one to control each link). Where would you place the motors and how would you transfer the rotation from the motor to the link? Explain your answer in two to three sentences. You could suggest mechanisms not found in the LEGO kits. [5 points]

2. Would you want a high or low gear ratio for each link? Explain your answer in one to two sentences. [5 points]

3. Explain 2 design priorities for your robot that would be beneficial for doing this lab in the real world. [10 points]

1.2 Programming [30 points]

Here is the environment that your robot will be operating in:

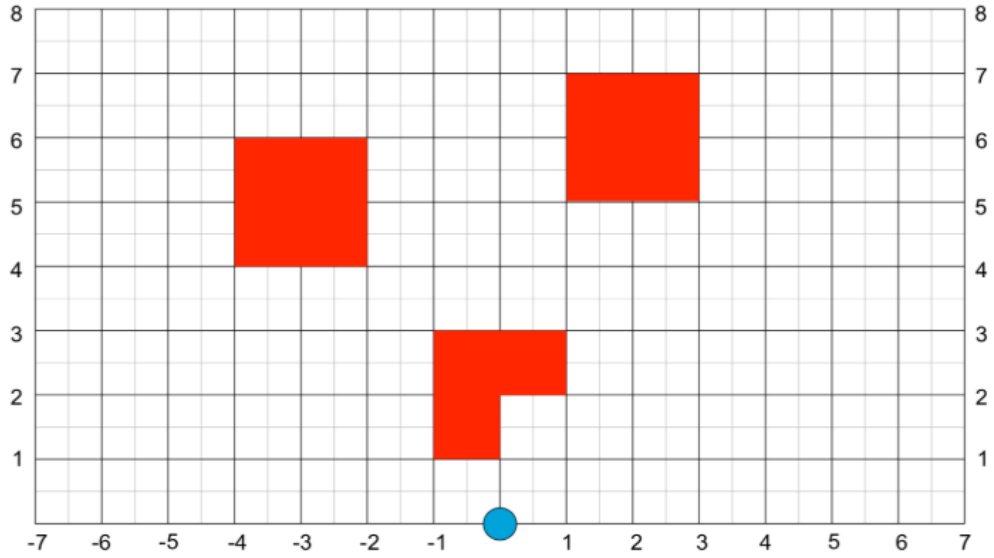


Figure 3: All dimensions here are in inches.

Create a configuration space for your robot where the x axis is the angle of the first joint with respect to the horizontal axis and the y axis is the angle of the second joint with respect to the first joint. Here is an example with a different arm and different obstacles (this example is slightly different from ours because for this one, the arm cannot exist "over" obstacles):

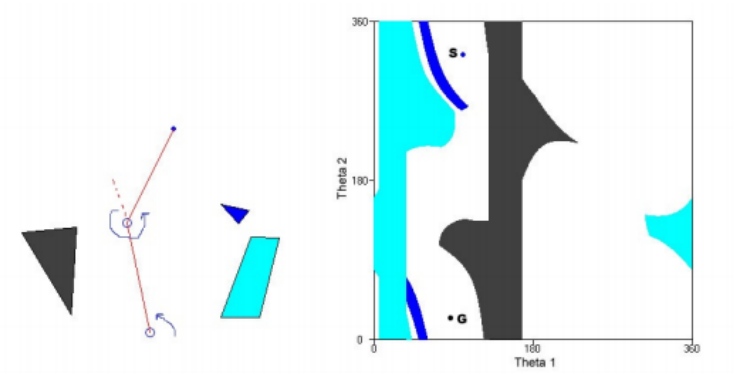
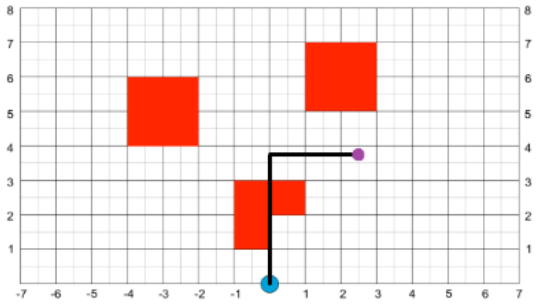
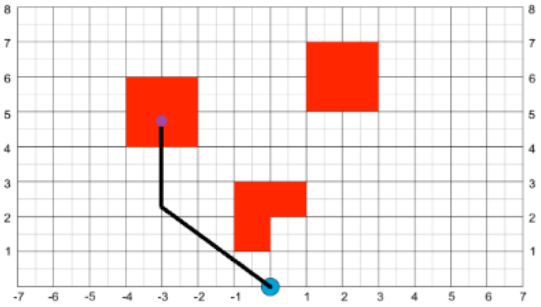


Figure 4: Configuration space for a sample RR arm with no joint limits.

As mentioned in the Introduction, the arm can exist "over" the obstacles but the end effector is not allowed to be "over" an obstacle.



Good configuration



Bad configuration

2 Part 2 (Week 2)

2.1 Written [10 points each]

1. How will you represent viable points for the arm to travel to? Will you use a grid? Waypoints? Which data structure will you use? Explain why you made these choices.

2. Which path planning algorithm(s) do you plan do use? Explain why you chose this.

3. How will you control the arm's angle using only torque? How many controllers? How will it work?

2.2 Programming [70 points]

Given three pairs of (x,y) waypoints from the user, plan and implement a path in theta space for your robot that goes to each point in order starting from the horizontal axis as a starting location. Display this path on the configuration space [15 points for a path, 30 points for a path that does not intersect obstacles]. Use torque commands to control the robot to follow this path in simulation. Use the starter code show this using the gym simulation. [10 points for attempting to go between the given points, 15 points for a path that hits obstacles slightly, 20 points for a path that does not hit obstacles but is slightly off the waypoints, 40 points for a path that does not hit obstacles and overlaps each waypoint completely] In order to get full points, you must get to each waypoint in less than 5 seconds (5 theoretical seconds, it may take longer to render). Please pause over each waypoint so that the TAs can tell if you just happened to pass over it or if you actually controlled your robot to that point.

ADDED INFO: you can access the arm's state via arm.state. arm.state[0] is angle of link 1, arm.state[1] is the angle of link 2, arm.state[2] is the angular velocity of link 1, arm.state[3] is the angular velocity of link 2.

The simulation is in SI units. The end effector has radius 0.005 m. Link 1 weighs 0.005 kg, link 2 weighs 0.001 kg. You are controling joint torques in Nm.

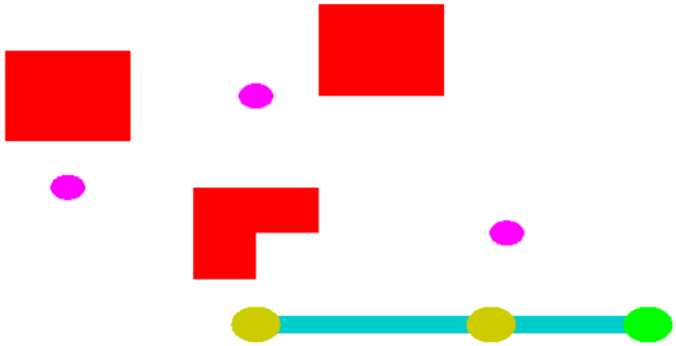


Figure 5: Here the arm is colored cyan. The end effector is green. Waypoints that the user writes in are pink. And the obstacles are red.

What To Submit

Submissions are due on Autolab by the date specified in the Syllabus.

1. Week 1: Submit the written portions as a .pdf on Gradescope. Zip all necessary files for Part 1 for Autolab. Submit AS A GROUP for both.

2. Week 2: Submit the written portions as a .pdf on Gradescope. Zip all necessary files for Part 2 (including any that you already submitted) for Autolab. Submit AS A GROUP for both