Alex Valente
Qing Ye
April 26, 2019
CSCI 445

Final Project Write up

We are Alex Valente and Qing Ye, senior students and both Computer Science majors. We took this class because we find the topics relating to robotics, such as planning, signal processing, learning, localizing etc., are very interesting. Also, because of this class, we have both gained more hand-on experience with python and general robotic programing. In this write up we go over our thoughts and process while developing the final result. First we start with an explanation of the forward and inverse kinematics and go into detail on the implementation of the code. And for the other aspect of the write up we address the use of our modified RRT algorithm.

Forward and Inverse kinematics

Forward kinematics takes in the joint angles of the arm and calculate the positon of the end effector. The method includes using of trigonometry to calculate and the length of the arm between joints to calculate the end position.
Inverse kinematics takes in the end-effector position and calculate the joint angles the arm need to pose. The calculation involves using inverse trigonometry so the range of inputs is very limited. Usually, there will be at least two combinations of the join angles to make sure the end point is at the inputted position. The algorithm will choose one of the based on certain criteria.

Code Explanation

My code will use the inverse kinematics function to move the end point into a given location that is good for the gripper to grab the cup. I will then use forward kinematics function and self.arm.go_to function to move up the arm and put the cup of the shelf. To make sure the arm is moving smoothly so that the cup won't be thrown away, I have put my moving functions inside for loops. In this way, it is very likely that the algorithm will perform well in the real-world situation.

RRT Path Planning

When first thinking of a strategy of locomotion for the robot, I decided that exclusive use of the RRT algorithm would be a good enough solution. But after initial implementations, I found the robot to be jerky. Swaying back and forth, following a jagged randomly generated path. The first thing I did to improve the algorithm was to add a simple curve smoothing function over the points. This created a fairly nice smooth continuous curve to the end. Yet after testing this method, while the robot would reach the destination, it might be off the goal spot by small amounts, some larger than others. And with the very precise nature of the robotic arm picking up the cup, this would create issues. What I did to combat this, was when constructing the random path, set the endpoint, just slightly in front of the final goal, and then constructed another RRT path between the end of the original path, and the goal. This path was maybe 6 inches long but it gives the robot a second chance at calculating the final approach. While it was significantly more successful with this method, the success rate of getting the robot to position was approximately 50%. What I came to realize, is that on the second smaller RRT path I create, the curve fitting would often not have enough space or data points to create a well fit curve and that would bring the robot away. To solve this problem, I decided that the jagged movements were exactly what I needed to get that final approach to be perfect. So, I tested again with the curve smoothing not being used on the smaller second RRT path, and the robots positioning success rate increases to almost 8/10. This was the final implementation we used for both the simulation with the robotic arm and on the physical demo.

Summary

The best thing we have learned from this project is that theoretical work may not be a good replica for the real world. For example, I have programmed the algorithm for the arm in the first wee and I thought that the final project might be even easier than the programming assignment 2. However, when I've come to the lab to demo my method to the TA, KR has told me you should avoid gripping the cup upside down because it is very likely that the gripper can't hold the cup tightly enough. Therefore, I have been working on programming a more effective and smooth way of transmitting the cup. When the moving routine is good enough, I also have to think about the moving speed of the arm as if I have moved the arm too fast, the cup will be thrown away. The seemingly easy problem requires lots of improvement and testing when dealing with real work robot performance. This is one of the best lesson I have learned.