Chris Chang

ctchang

Robot Kinematics and Dynamics

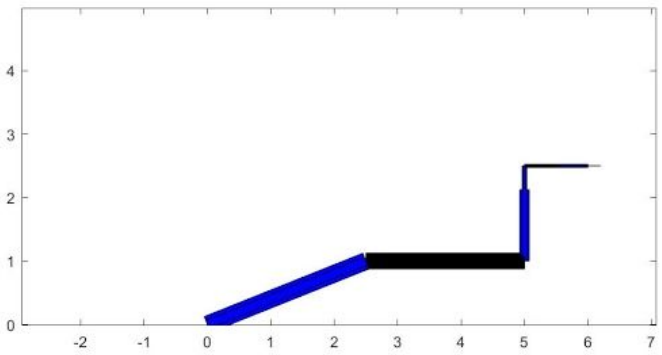
12/10/2020

Capstone Writeup

The goal of this project was to design and program a simulated robot to follow a wire existing in the workspace [-5 5]x, [-5 5],y [0 5]z. After building the simulator, the first step was to develop a RingBot class with the method 'followWire' capable of generating a number of waypoints along an arbitrary path in the designated workspace and the joint angles and link extensions to reach each waypoint. Because the number of points varied between each wire, a spline sampled from 250 points along the wire was used to generate the location of each waypoint. The end effector of the RingBot was a ring (as the name suggests) with the z-axis through the center of the ring. To maximize the allowed tolerance between the target location and actual location of the ring centroid, at each waypoint, the z-axis was defined to be the unit tangent vector of the wire trajectory. In order to minimize the required link lengths of the robot, the $x_{waypoint}$ was then defined as the unit vector in the direction v - proj(v, $z_{waypoint}$) where v is the vector from the base of the robot to the waypoint, and $y_{waypoint}$ as the cross(z, x). From these axes, the ZYZ euler angles of each waypoint could be extracted. At each waypoint the wire pose was defined to be $[x, y, z, \phi, \theta, \psi]^T$.

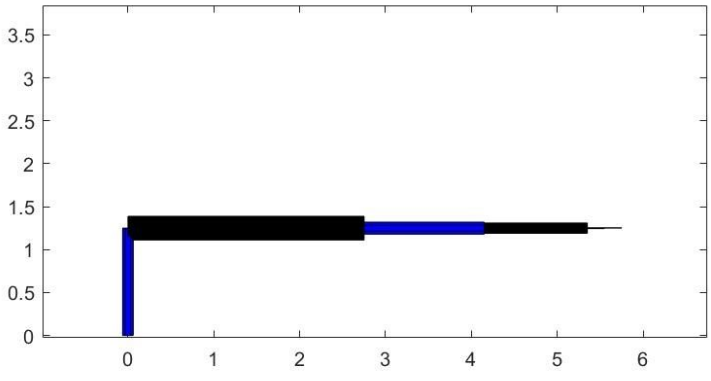
Iterative design and redesign was central to this project, so I opted to use a numerical implementation of both jacobian and inverse kinematics, as the design of each robot (and consequently its forward kinematics, dynamics, inverse kinematics, etc) would be changing constantly. While numerical IK/gradient descent take much longer to compute than analytical IK, the robustness of the algorithm made it ideal for this project. Additionally, numerical IK produces a much smoother trajectory between two waypoints, as if there are multiple solutions to an IK problem, gradient descent will find the solution nearest to the last configuration, while analytical IK does not. To generate the trajectory of the robot along the wire, the inverse kinematics were computed for the wire pose at each waypoint.

Robot 1 is very similar to a scara arm with a spherical wrist, however instead of approaching from above it does so from below. Approaching from below made it easier for the robot to reach the wires without collisions with the links, as well as keeping link lengths



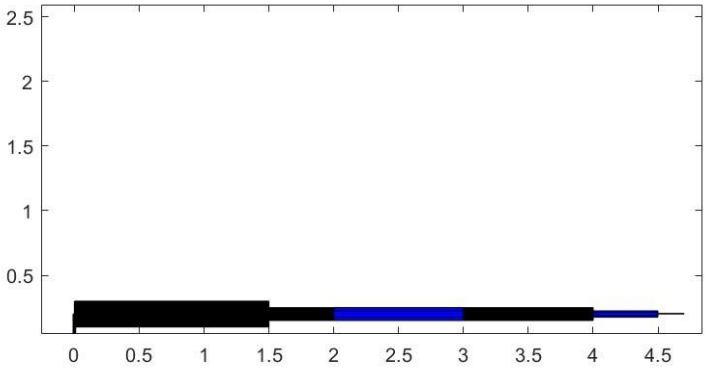
shorter. In the configuration shown, the joint axes are [z, z, z, y, x] with link 3 being prismatic. The initial configuration of this robot had very short link lengths in the wrist and longer 'main links' to compensate. The idea was that revolute joints 1 and 2 along with the prismatic joint would handle end effector position and the wrist would handle orientation. However, this approach failed as it required the fixed length of the prismatic joint to be far too long, resulting in body collisions with the wire. The z component of link 1 was also shortened to allow for the longest possible prismatic joint. Additionally, in its initial configuration, the x component of link 1 was longer than link 2 constraining the robot to an approximately tubular workspace. In order to expand the robot's workspace, each link's x component was made to be 2.5, expanding the workspace from a tube to a cylinder.

Robot 2 is a 6 link arm with 6 revolute joints and 0 prismatic joints. As shown in this configuration its joint axes are [z, y, y, x, z, x]. The robot's initial configuration had all link lengths equal to 1 for simplicity. While the design was successful in reaching its target poses, the time it took to compute its configuration for each waypoint was far too long. To address this, the wrist links (5, 6) were shortened, and links 2 and 3 were made longer to maintain the former workspace. This did end up shortening the computation time for the robot's inverse kinematics, presumably because the $[x,y,z]^T$ gradient with respect to joints 4, 5, and 6 would now be insignificant compared to the gradient with respect to joints 1, 2, 3, allowing for much smoother (and therefore quicker) descent to the target position. Like robot 1, link 1 was made to be the same length as link 2+3 to move the workspace from approximately a hollow sphere to approximately a sphere with the same radius. Lastly, in its elbow down configuration between links 1,2,3, the robot would sometimes clip its elbow through the ground. To compensate for this, link 1 was extended, however this also created manipulability issues for wire positions near (0,0,2). Because of this, some experimentation needed to be done with the size of length 1 to find a happy medium, but both issues were always prominent for every length, leading me to ditch the design and move on to a new robot configuration.



Robot 3 is an RPR robot with a spherical wrist. As shown in this configuration, its joint axes are [z, y, x, z, x] with link 2 being prismatic. (link 1 is very short). Like Robot 2, the wrist components were shortened and the remaining links extended to maintain the workspace. Link 1 was made to be extremely short and links 2 and 3 were extended, making the robot's workspace a hemispherical shell. Robot 3

struggled with singularity issues when its ring was near the z axis, so the links in the wrist were made slightly longer to increase the manipulability of the robot in these configurations. The largest issue that Robot 3 faced was with maximizing its workspace. While swapping a revolute joint for a prismatic joint aided in avoiding physical constraints such as contacting the ground and the wire, it severely reduced the workspace of the non-wrist part of the robot. To deal with this issue, the wrist was made to be even longer, to extend the workspace of the overall robot past the shell created by joints 1,2,3. This solved the workspace problem, but extended the necessary time to compute the robot's inverse kinematics. Like Robot 2, some work was done to maximize the balance between workspace and computation time, however both issues always persisted, prompting the design of Robot 1.



My implementation of numerical inverse kinematics allowed the arms many iterations of gradient descent to reach an error threshold within the ring radius, so to quantify performance between each arm, I will be comparing the average time it took each arm to compute and execute the trajectory along all three wires, as each robot was capable of reaching every waypoint along each wire with a similar number of collisions. To score cost, I added one point for each revolute joint and two for each prismatic joint. Robustness was scored qualitatively, ranking each robot in terms of the amount of time/effort I needed to spend tuning each arm to find a successful configuration.

	Robot 1	Robot 2	Robot 3
Execution Time (s)	52	87	59
Cost	7	6	7
Robustness	9	4	2

To compare each robot and select a best, these parameters were compared, prioritizing robustness, execution time, and then cost. I believe that robustness is by far the most important parameter, as the robots will also be tested on mystery wires, so the robot that required the least amount of tuning to succeed on the three given wires is the most likely to succeed. Execution Time was the next most important parameter, as the robots also had a time constraint to run on the test wires. Robot 2 averaged an

87s execution time on my machine, and because there was no guarantee that the simulation will run as fast on a TA's laptop, Robot 2 was disqualified from consideration. Between Robots 1 and 3, Robot 1 was by far more robust, as well as slightly faster, making it the obvious choice. The biggest problem I can foresee for robot 1 is its execution on low wires (lower than 2m). In this region, the robot is extremely likely to intersect with the wire, as the lowest point on the fixed portion of the prismatic joint is ~2m from the ground and the base of the prismatic joint can occupy a circle in the xy plane.

I learned a lot about the many nuances of trajectory planning in this project. In the early stages of this project, I attempted to use analytical inverse kinematics, which would always produce a correct solution quickly, however there was never a guarantee that the elbow orientations of a given waypoint would match the elbow orientations of the previous waypoint, resulting in an unachievable trajectory. After switching to numerical inverse kinematics, this problem was solved, but there were a number of new issues. The most prominent was runtime, which persisted throughout the entire project. If I were to repeat this project, I would add many more properties to the RingBot class such as homogeneous transforms, jacobians, and rotation matrices in order to save on computation time. Overall, this project taught me about the trade-offs that exist when designing and developing a robotic system. Every time I wanted to make a change to a robot, I had to be aware of the way that change would impact the robots workspace and manipulability, inverse kinematics runtime, and all the new ways the robot could collide with the wire. Trying to blindly design a robot with all of these things in mind would have been impossible, so I will absolutely be using simulation to some degree in every future project.

Feedback

I wish we would have had the opportunity to implement more of what we learned in class into code. I feel like I have strong knowledge of the theory behind robot kinematics and dynamics, but felt very lost moving into the capstone project. I'm assuming that this is because of remote classes, and our inability to use the REL, but I think having some sort of coding component to assignments 6-8 would have made the final project feel much more familiar. I think the strongest parts of this course were the OLI modules and online notes/lectures, as they were very helpful to reinforce learning past lectures.