Andrew Tarnoff

Modern Robot Programming Problem Set #4

CODE LINK: https://github.com/art81/EECS373
Navigate to:
problem_set_5/src/irb120_reactive_task_commander_with_coords.cpp

Theory Of Operation:

The program first sets up the environment and then takes in two inputs, one for the desired final X-coordinate of the gear and one for the desired final Y-coordinate of the gear. In order to achieve this I have three main while loops. The first and largest one contains the two others. The first/outer loop will continue to run until both the x coordinate and y coordinate are within the desired precision as described later in the report. One of the inner loops operates to get the gear within the desired precision for the X-coord and the next operates to get the gear within the desired precision for the Y-coord. These two loops basically operate the same way. They will run until each individual coordinate is within the desired precision. This is done by first lowering the flange to the correct level and on the correct side of the gear (based on which coord is being moved and which way the gear needs to be pushed). The flange then moves a certain amount (defined by some simple subtraction and addition) so that it pushes the gear into the “correct” location. After this it will then command the flange to move directly keeping the same x, and y coordinate and commanding a z coordinate of 0.15m. If the gear still isn’t in the correct location then the flange will take the same path as before but will move 0.002m further (when pushing the gear because that is the precision I am trying to achieve) and this will happen infinitely until the desired precision is achieved for that axis. The next loop will then run to achieve the precision of the y coordinate. It will continue to do this until both the x and y coordinate are within the desired precision and will then command that the flange goes back to the origin (all joints are 0).

Observations of Behavior and Limits

Observations

One observation that I had was that when I commanded a z coordinate that should have been low enough to push the gear, sometimes it would slip over the gear after pushing the gear for a short distance. This does not make any sense because the inverse kinematics makes sure that the flange’s linear and rotational coordinates are exactly what we set them to be so there is no reason why the flange should be low enough to push the gear sometimes and not others even though they were commanded to have the same z-coordinate. In the end, the robot still pushed the gear the whole way because it continues to run until the desired precision is reached but I fixed the issue by lowering the z coordinate even further so that a little variation in z-coordinate value still allowed the flange to push the gear.

Limits when pushing from (X1,Y1) to (X2,Y2) where X1 = Y1 and X2 = Y2:
Robot can push from a minimum of about ***(0.16m, 0.16m)*** to a maximum of about (***0.4m, 0.4m).***
Robot can pull back from a maximum of about ***(0.34m, 0.34m)*** and can pull to a minimum of about ***(0.13m, 0.13m).***

Limits when pushing along just the X (Y = 0) or just the Y (X = 0) axis:
Can push from a minimum of about ***0.22m*** along given axis to a maximum of about ***0.59m***
Can pull back from a maximum of about ***0.45m*** and can pull to a minimum of about ***0.15m***.

Precision

The precision of my program is that it will always be able to push the gear within a precision of 0.002m or 2mm of the goal pose's X and Y coordinates as defined by user input. Because in every loop the xMove and yMove variables are incremented by 0.002m (meaning that the flange will move an extra 0.002m every single loop) the gear will always move 0.002m more in the correct direction if the gear still wasn't within the correct precision after the first try. Since this will keep happening until the precision is within 0.002m, it is guaranteed that the robot will always succeed (as long as nothing else goes wrong).