AuE 8220- Autonomy: Mobility and Manipulation

Homework 4: Control of Mobile-Manipulator Robot using MATLAB-CoppeliaSim

Authors: Tanmay Samak, Chinmay Samak, Riccardo Setti, Olamide Akinyele

Problem 1:

1-A

Concept: We align the robot w.r.t. global X-axis of the simulator and make it drive straight for 5 m with a velocity of 0.1 m/s using the following IK:

$$\begin{bmatrix} u_{fr} \\ u_{fl} \\ u_{rr} \\ u_{rl} \end{bmatrix} = \frac{1}{r} * \begin{bmatrix} 1 & 1 & l+w \\ 1 & -1 & -l-w \\ 1 & -1 & l+w \\ 1 & 1 & -l-w \end{bmatrix} * \begin{bmatrix} v_x \\ v_y \\ \omega_z \end{bmatrix}$$

where, $u = \begin{bmatrix} u_{fr} \\ u_{fl} \\ u_{rr} \\ u_{rl} \end{bmatrix}$ is the control input vector for the 4 wheels (f=front, r=rear; l=left, r=right); r is the radius of the wheels; l is the half-wheelbase of the robot (distance of wheels from center of robot w.r.t. local X-axis); w is half-track-width (half of the distance between each pair of front/rear wheels w.r.t. local Y-axis); and $\begin{bmatrix} v_x \\ v_y \\ \omega_z \end{bmatrix}$ is the velocity vector of the mobile robot (linear velocity w.r.t. X and Y axes, and angular velocity w.r.t. Z axis).

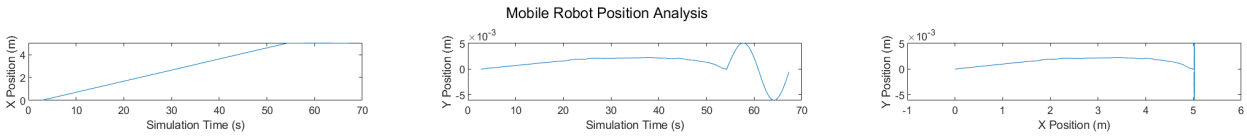
Code: `KUKA_YouBot.m` lines 78-106

1-B

Concept: We detect the position of robot and once it reaches at the end of 5 m linear trajectory (from Problem 1-A), we command it a linear velocity of 0 m/s and an angular velocity of 0.5 rad/s (using the same IK as described in Problem 1-A).

Code: `KUKA_YouBot.m` lines 107-132

Results for 1-A and 1-B



The mobile robot moves in "almost" straight line w.r.t. X-axis at a linear velocity of 0.1 m/s from t≈5s to t≈55s (the negligible motion in Y-axis is due to dynamics-based instability of the robot being simulated using CoppeliaSim). It then rotates about the Z-axis at an angular velocity of 0.5 rad/s from t≈55s to t≈70s.

Note: *The plots above include timesteps during which the mobile robot moved linearly as well as when it rotated. The* `Results` *folder hosts additional plots (generated from CoppeliaSim) for reference.*

1-C

Concept: We detect the amount of rotation that the mobile robot undergoes to determine the completion of 1 full rotation and then stop the mobile robot completely. We then compute the pose of end effector at the starting point of the task-space linear trajectory using `ikine()` function from PCRTB. We then plan a quintic polynomial joint-space trajectory between the current and commanded joint angles using `jtraj()` function from PCRTB. We then transmit the joint angles at each instant in trajectory to CoppeliaSim to command the manipulator robot to move its end effector from current pose to the prescribed pose at the starting point of the task-space linear trajectory. We now plan a quintic polynomial task-space trajectory between the initial `transl(350,0,0)` and final `transl(350,0,0)` task-space cartesian coordinates using `ctraj()` function from PCRTB. We then compute the pose of end effector at each point in the task-space linear trajectory using `ikine()` function from PCRTB. Finally, we transmit the joint angles at each instant in trajectory to CoppeliaSim to command the manipulator robot to move its end effector along the task-space linear trajectory.

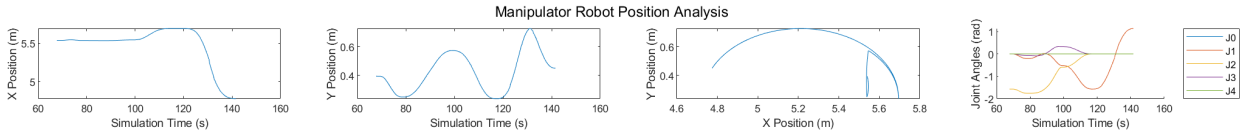
Code: `KUKA_YouBot.m` lines 133-225

1-D

Concept: We plan a quintic polynomial joint-space trajectory between the current and commanded joint angles (maximum reach of arm in forward direction, i.e. [0,0,0,0,0] degrees) using `jtraj()` function from PCRTB. then transmit the joint angles at each instant in trajectory to CoppeliaSim to command the manipulator robot to move its end effector from current pose to the prescribed pose at the starting point of the task-space semicircular trajectory. Similarly, we now plan a quintic polynomial joint-space trajectory between the current and commanded joint angles (maximum reach of arm in backward direction, i.e. [0,-65,0,0,0] degrees) using `jtraj()` function from PCRTB. Finally, we transmit the joint angles at each instant in trajectory to CoppeliaSim to command the manipulator robot to move its end effector along the task-space semicircular trajectory (of maximum reach).

Code: `KUKA_YouBot.m` lines 226-312

Results for 1-C and 1-D



The manipulator robot first moves to the start location of linear trajectory (t≈70s to t≈80s), then executes a linear task-space trajectory (t≈80s to t≈100s). It then moves to the start location of semicircular trajectory (t≈100s to t≈110s), then executes a semicircular task-space trajectory (t≈110s to t≈140s).

Note: *The plots above include timesteps during which the manipulator robot moved. The* `Results` *folder hosts additional plots (generated from CoppeliaSim) for reference.*