Mobile Manipulation Capstone Final Project

Yimin Han(netid: YHA1926), Northwestern University

Introduction:
In this project, I wrote a software to plan and control the trajectory of the KUKA youBot to pick up a block at the start location, carry it and put it down on the destination. The simulations were accomplished in CoppeliaSim.

My work is composed of following parts:
1. Implement TrajectoryGenerator() function to plan a reference trajectory for the end-effector of the youBot mobile manipulator
2. FeedControl() function is to calculate the control law. The feedback control of the youBot is calculated by the kinematic task-space feedforward plus feedback control law:

$$\mathcal{V}(t) = [\mathrm{Ad}_{X^{-1}X_d}]\mathcal{V}_d(t) + K_p X_{\mathrm{err}}(t) + K_i \int_0^t X_{\mathrm{err}}(\mathrm{t})dt.$$

This equation means that we need the current actual end-effector configuration, current and future reference end-effector configurations and Kp, Ki gains to calculate the end-effector twist ***V*** expressed in end-effector frame {e}. Then, implementing the pseudoinverse of the mobile manipulator Jacobian J_e to compute the control signal:

$$\begin{bmatrix} u \\ \dot{\theta} \end{bmatrix} = J_e^{\dagger}(\theta)\mathcal{V}.$$

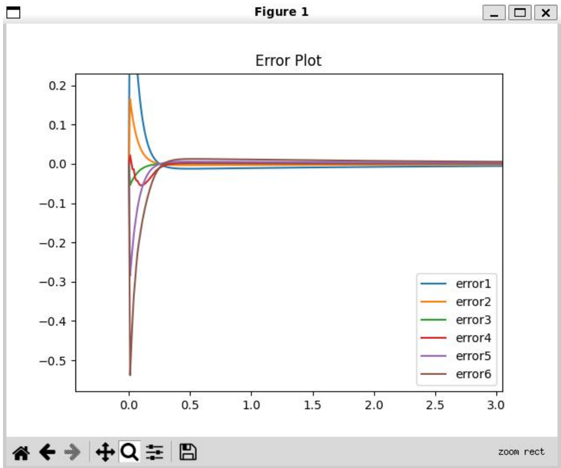
The control signal will be used by NextState() function to evolved the kinematics of youBot into future state after Δt.
3. Use NextState() function to simulate the kinematics of youBot. Given the current configuration(chassis phi, chassis x, chassis y, J1, J2, J3, J4, J5, W1, W2, W3, W4, Gripper State), the function will return the configuration of next moment(Δt = 0.01s)

To execute the program, please run:
python capstone.py

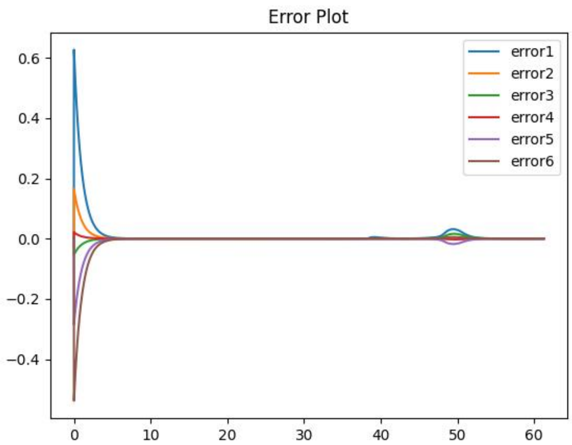
Results:
Simulate the youBot movement by CoppeliaSim with different gain or cube positions. The movement is smooth enough. The video can be found in the results/best&overshoot&newtask folder respectively.

Xerr of the control law has six dimensions. So I plot these six elements respectively as six error plots in the same plot:

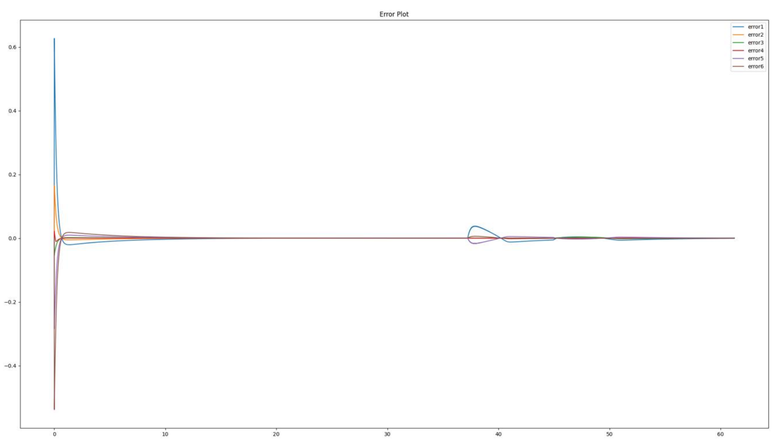
“best”:(zoomed in version in oreder to show the convergence process)



“overshoot”:



“newtask”:



We can find that the motion is smooth without any overshoot and the settling time is short.