Robotic Manipulation – Final Project

In this project I wrote a software that plans a trajectory for the end-effector of the youBot mobile manipulator, performs odometry as the chassis moves, and performs feedback control to drive the youBot to pick up a block at a specified location, carry it to a desired location, and put it down.

The final project contains 3 milestones that where implemented in the code "Full_Program.py". The main function calls the 3 milestone functions:

1. **NextState()** - This function compute the configuration of the robot in the next time step.
2. **TrajectoryGenerator()** - This function Generates the reference trajectory for the end-effector frame {e}.
3. **FeedbackControl()** - This function calculates the kinematic task-space feedforward plus feedback control law.

In the first milestone, I defined a function **NextState()**, that uses the robot current configuration and speed to find the next configuration. The function also gets a value of maximum angular speed and restrict the speed to the maximum speed allowed.

In the second milestone, I defined a function **TrajectoryGenerator()** that uses the ScrewTrajectory() function from the Modern Robotics library to find the configurations along the robots path, as it goes through several waypoints:

1. Starts from rest in initial position
2. From initial position to standoff position (near the cube position)
3. From standoff position to grasp position
4. Closing the gripper
5. From grasp position to standoff position (near the cube goal position)
6. From standoff position to goal position
7. From goal position to final position
8. opening the gripper
9. From final position to standoff position

I used the get_list_from_matric() function to convert the matrix returned after each segment into a list, and append it to the existing trajectory list of last segments.

In the third milestone, I defined a function **FeedbackControl()**, that uses the current, the desired and the next end-effector configuration to compute The commanded wheel and arm joint controls, and the error between the current and the desired states. The controls are found using feedforward plus feedback control laws.

After completing all the milestones code, I wrote a **full program** code that uses the mentioned function to find the trajectory between the waypoints, while controlling the motion parameters.

The code's main function defines the controller's proportional and integral gains, that can be tune according to the desired results.

Through the main function we can also control the robots initial configuration, or change the cube start of finish position.

The function returns:

1. A saved csv file represents the N configurations of the end-effector along the entire concatenated eight-segment reference trajectory, including the gripper state (a 13-vector)
2. A saved csv file represents the Xerr (a 6-vector)
3. A saved plot of the Xerr changing with time
4. A saved log file

Using the full program code, I could compute the trajectory and error of 3 different cases – the "best" case (no error, no overshoot), The "overshoot" case (no error, with overshoot), and the "newTask" case (different initial and final block configurations, different initial robot configuration).

The code mentioned can be found in the folder "code", and the results for the "best", "overshoot" and "newTask" can be found in the folder "results". For more information on the cases implementation, you can go to the README.pdf file in everyone of the folder mentioned above.