Instructions to Run the code

Make sure you have python 2 installed with the following libraries.

- ❏ scipy
- ❏ numpy
- ❏ matplotlib

This program has been tested on python 2.7 on Ubuntu 16.04

Run the python file

Example: In linux, open a terminal and type "*python main.py*"

Folder Descriptions

/code :

Contains the source code of the assignment. Run python main.py inside code directory, to run the program

/results :

Contains results generated using the program. This folder has 4 sub directories

/results/best/ :

Results of the program, using tuned gain (Kp, Ki) parameters

/results/overshoot/ :

Results of the program, using underdamped gain (Kp, Ki) parameters

/results/new :

Results of the program, for a new Cube configuration

/results/collisionAvoidance :

Comparision showing the performance of developed self collision avoidance method

Program Details

INPUTS

The program takes 4 inputs.

1) Initial Configuration of end effector (phi, x, y, J1, J2, J3, J4, J5, W1, W2, W3, W4)
2) Initial Configuration of cube (x, y, theta)
3) Goal Configuration of cube (x, y, theta)
4) Gains (kp, ki)

For further details on input configuration formats, please refer
http://hades.mech.northwestern.edu/index.php/Mobile_Manipulation_Capstone
Note: All measurements are in meters, radians and seconds.

CONSTANTS

- The timeStep of the program simulation is fixed to 0.01 seconds. Hence, the program will save End Effector configurations at each 0.01 second interval.

- However, the robot feedforward and feedback controller will run at 1kHz (per every 0.001) second.

- The robot's End effector trajectory's initial reference is fixed.
  Tse_refInitial = [[0, 0, 1, 0], [0, 1, 0, 0], [-1, 0, 0, 0.5], [0, 0, 0, 1]]

- The robot's joints are velocity limited to following values.
  velocityLimits (phi, x, y, J1, J2, J3, J4, J5, W1, W2, W3, W4) = [10, 10, 10, 20, 20, 20, 20, 20, 10, 10, 10, 10]

OUTPUTS

At the end of execution, the program will save two CSV files.

youBot_Trajectory.csv

youBot_Trajectory file will contain the configuration of the youBot end effector, at each 0.01 second time interval. Comments will be printed in the file, indicating the specific format of the configuration (phi, x, y, J1, J2, J3, J4, J5, W1, W2, W3, W4). ***The youBot_Trajectory.csv file can be loaded into CoppeliaSim Scene6_youBot_cube.***

trajectory_Xerr.csv

trajectory_Xerr file will contains the error-twist, at each 0.01 second time interval. Comments will be printed in the file, indicating the specific format of the error (Wx, Wy, Wz, Vx, Vy, Vz)

Finally the program will also visualize the trajectory_Xerr data using a multi-line plot.

Implementation Notes

1) End Effector trajectory generation will use fifth-order polynomial time scaling.
2) The implemented controller is a feedforward + feedback PI controller
3) No joint limits have been imposed on the robot arm
4) Robot self collision avoidance has been implemented (See below for details)

The simulated trajectory has 8 sections.

Trajectory Component	**Motion Time (Seconds)**
A trajectory to move the gripper from its initial configuration to a "standoff" configuration a few cm above the block	3
A trajectory to move the gripper down to the grasp position	1.5
Closing of the gripper	1
A trajectory to move the gripper back up to the "standoff" configuration	1.5
A trajectory to move the gripper to a "standoff" configuration above the final configuration	3
A trajectory to move the gripper to the final configuration of the object	1.5
Opening of the gripper	1
A trajectory to move the gripper back to the "standoff" configuration	1.5
Total Motion Time	14

Self Collision Avoidance

Joint value limiting was implemented at first, according to the Capstone guide, at *http://hades.mech.northwestern.edu/index.php/Mobile_Manipulation_Capstone*

However, as the guide suggests, "Your approximation should be conservative, meaning that allowed configurations are never in self-collision. But it should not be so conservative that the arm's workspace is overly constrained, preventing the robot from doing useful work". After testing with CoppeliaSim Scene3_youBot, I realized that it is difficult to find joint limits that guarantee no-self-collisions without restricting the robot's feasible task space significantly.

Hence I have developed another way to avoid self-collisions, without imposing strict joint limits.

1. The robot base and the arm base is modeled by a cuboid and a vertical cylinder.
2. This modeled space is designated as restricted space for the end effector. By observing youBot, we can guarantee that if the end effector is not within the restricted space, there cannot be any self collision.
3. At each 'nextConfiguration' simulation using the youBot kinematic simulator, I evaluate whether the next configuration penetrates the restricted space.
4. If nextConfiguration penetrates the restricted space, the robot Jacobian columns relevant to robot arm (5 columns) are set to zero and the controls are calculated again. When the arm jacobian columns are set to zero, the robot controller will not request any motion from the robot arm, instead it will try to generate the required twist by changing the chassis configuration.

Performance of this collision avoidance method is shown in results.

Example Inputs

Given below are example inputs for each test case. Refer the 4 sub directories in the Results folder.

Best Results

initial configuration = [0.6, 0.0, 0.0, 0.0, -0.35, -0.698, -0.505, 0.0, 0.0, 0.0, 0.0, 0.0]
Tsc initial = [1, 0, 0]
Tsc goal = [0, -1, -math.pi/2]
gains = [0.18, 0.00015]

Overshoot Results

initial configuration = [0.6, 0.0, 0.0, 0.0, -0.35, -0.698, -0.505, 0.0, 0.0, 0.0, 0.0, 0.0]
Tsc initial = [1, 0, 0]
Tsc goal = [0, -1, -math.pi/2]
gains = [0.3, 0.1]

New Results

initial configuration = [math.pi/2, 0.0, 0.0, 0.0, -0.35, -0.698, -0.505, 0.0, 0.0, 0.0, 0.0, 0.0]
Tsc initial = [0.7, 0.7, math.pi/4]
Tsc goal = [0, 0, math.pi]
gains = [0.18, 0.00015]

Cube Configuration	X (meters)	Y (meters)	Phi (radian)
Initial	0.7	0.7	0.78539
Goal	0	0	3.14159

Self Collision Avoidance Demo - Simulation 1

initial configuration = [math.pi/2, 0.0, 0.0, 0.0, -0.35, -0.698, -0.505, 0.0, 0.0, 0.0, 0.0, 0.0]
Tsc initial = [0.7, 0.7, math.pi/4]
Tsc goal = [0, 0, math.pi]
gains = [0.18, 0.00015]

Cube Configuration	X (meters)	Y (meters)	Phi (radian)
Initial	0.7	0.7	0.78539
Goal	0	0	3.14159

Self Collision Avoidance Demo - Simulation 2

initial configuration = [0.0, 0.0, 0.0, 0.0, -0.35, -0.698, -0.505, 0.0, 0.0, 0.0, 0.0, 0.0]
Tsc initial = [1, 0, 0]
Tsc goal = [0, 0, 0]
gains = [0.18, 0.00015]

Cube Configuration	X (meters)	Y (meters)	Phi (radian)
Initial	1	0	0
Goal	0	0	0

Additional Notes:

The unit_tests.py file in the '/code' directory has separate functions for each milestone for the project. If required, the results of the milestone can be evaluated by calling the required function.

If you are running the program to get an output yourself : Please do not copy from the ReadMe.pdf files and paste the inputs directly in the program terminal. Copied lines from a pdf file are strings, and the program requires the inputs as python Lists. Hence, **either copy the inputs from the Program Log.txt file, or else type the input yourself, as it is in the ReadMe.pdf file**.

Thank you for taking your time to review my project!