1. Things in my zip file:

- In my zip file, there are 2 directories in it, including 1 code directory and 1 results directory.
- The code directory includes 2 directories, one is called text type file, this is my code written in the text file, another directory called python type file includes my code written by vscode.
- The results directory includes 3 directories, including 1 best directory, 1 overshoot directory and 1 newTask directory. Each of these 3 directories includes 1 figure of error, 1 animation video, 1 X_err csv file, 1 YouBot csv file, 1 README text file, 1 log file and 1 directory named ended by Script.
- The result in the best directory is from the well-tuned feedforward-plus-P controller.
- The result in the overshoot directory is from not well-tuned feedforward-plus-PI controller.
- The result in the newTask directory is from the well-tuned feedforward-plus-P controller with a new cube initial and final configuration.
- In the directory ended by Script, there are some separated python file and you can just run the Script file to get the corresponding results to each different condition like best condition, overshoot condition and so on.

2. Brief introduction of my code:

- I use Python to complete my final project.
- There are 6 python files in the code directory, including a TrajectoryGenerator file, a NextState file, a FeedbackControl file, a testJointLimits file, a YouBot file and a Script file.
- The Script is easy for changing the initial configuration, cube position, Kp and Ki and you can just run this file to get the results.
- Then, the YouBot file is for realize the loop.
- The FeedbackControl file is for generating speeds and for adjusting errors.
- The NextState file is for generating the next configuration using previous configuration and speeds.
- The TrajectoryGenerator file is for generating the desired trajectory.
- We can just run the Script file and let it call other file to get results.
- I also have a directory named best_jointlimits including the same files in other three results directories, and you can compare this best_jointlimits result with best result, since the only difference between them is the jointlimit is open.
- In the testJointLimits file, applying the testJointsLimits function still can accomplish the pick and place job, and the only difference between it and without it is that joint 3 and 4 may be constrained in case of singularity, but the animation shows it still can finish its job and the error will be eliminated. Besides, seeing the animation, you can see a very small angle for making sure the joint 3 and 4 not achieving zero at the same time.

3. One surprising thing:

- When I set the height of gripper standoff to 0.3 meters, the arm will be very easy to shake when moving and the error is always overshoot at the latter part of the figure, then it converges again. However, if I lower the height to 0.1 meters, everything is fine, and the arm is not easy to shake, and the error is not easy to overshoot.

- I guess that may be because the workspace of the arm is constrained. However, I have debugged this for many days, it totally surprised me after I found the reason is just the height of gripper standoff position.

4. Detailed introduction of my code:

a. In function TrajectoryGenerator:

- Input: Tse_initial, Tsc_initial, Tsc_final, Tce_grasp, Tce_standoff and k, here I set k=1.
  Output: An n by 13 matrix which contains the end_effector and gripper state information representing the trajectory.
- I use mr.CartesianTrajectory to generate each transformation matrix of end-effector of each step in each of the 8 segments.
- At the beginning, I just set all the end elements in the n by 13 trajectory matrix to 0 and finally I use a for loop to change the end elements in segments 3, 4, 5, 6 to 1 for representing closing the gripper.

b. In function NextState:

- Input: config, speed, delta_t and speed_max.
  Output: a new configuration which is a n by 12 matrix, I will complete it to n by 13 matrix in the main loop in YouBot.py.
- First, limiting the input speed.
  Second, computing the new arm and wheels configurations.
  Third, using odometry to compute the configuration of chassis.
  Finally, put the chassis, arm and wheels configurations together forming the new configuration.

c. In function FeedbackControl:

- Input: config, Xd, Xd_next, Kp, Ki, dt.
  Output: the commanded twist, speeds and X_err.
- First, using the input configuration to compute X.
  Second, writing down Vd, Vb and X_err to compute the commanded twist V.
  Third, using mr.JacobianBody to compute J_arm and using Teb and F6 to compute J_base.
  Fourth, combining J_arm and J_base into Je and using the pseudoinverse and rcond to compute speeds and avoid singularity.

d. In function testJointLimits:

- In this function, I test the joints 3 and 4 and if their angle is larger than -0.2, the corresponding columns in Je will change to columns with all 0, so these joints will not have speeds and stop singularity.

e. In function YouBot:

- First, inside the loop, I first compute the Xd and Xd_next and use function FeedbackControl to compute speeds and X_err.

- Second, I change the order of speeds since the joints speeds need to be in front of the wheels speeds.
- Third, I input the configuration, which will update all the time, and the speeds calculated previously and some parameters to compute a new configuration.
- Finally, put the new configuration in one array and X_err in another array and output these arrays into 2 csv files and plot errors.

f. In file Script:

- In this file, I just generate the desired trajectory calling the function TrajectryGenerator and set up some initial configuration, Kp and Ki and call the function YouBot to get results.