Robot DKDC, Fall 2017

Final Project Report

Team 8

Soowon Kim

Amrita Krishnaraj

Yingtian Xu

Introduction

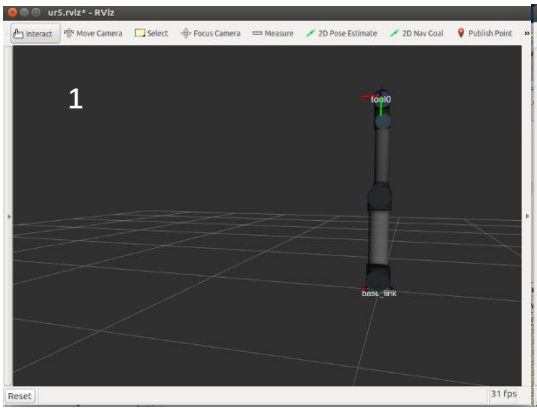
The objective of the final project is to use Inverse Kinematics, Rate Control, and Gradient Control to complete pick-and-place operation.

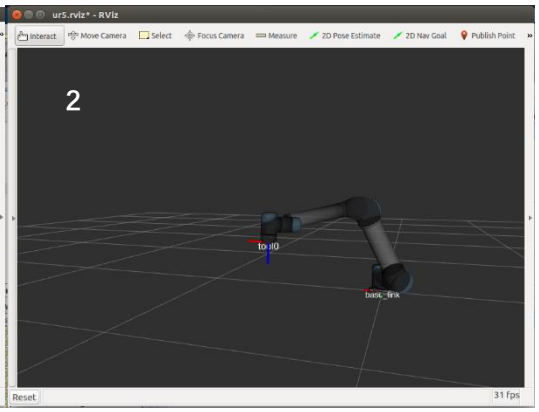
Extra objective of the final project is to use Pseudo Inverse Control to complete the pick-and-place operation and use UR5 to draw the Blue Jay and dance.

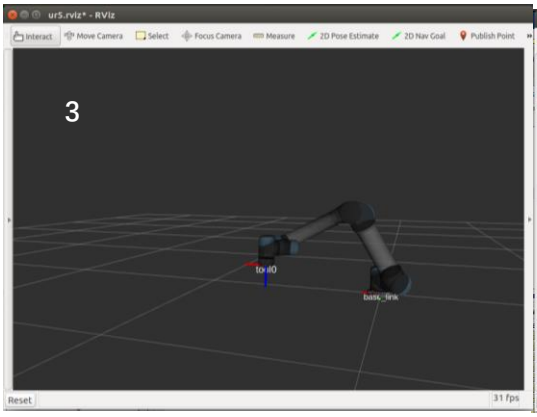
Workflow

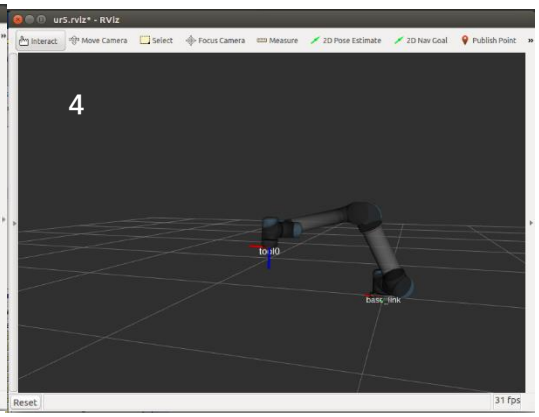
The start and target configurations (g_start and g_target) are taught then the algorithm automatically calculates the configurations that are above the start and the target (g_start1 and g_target1).

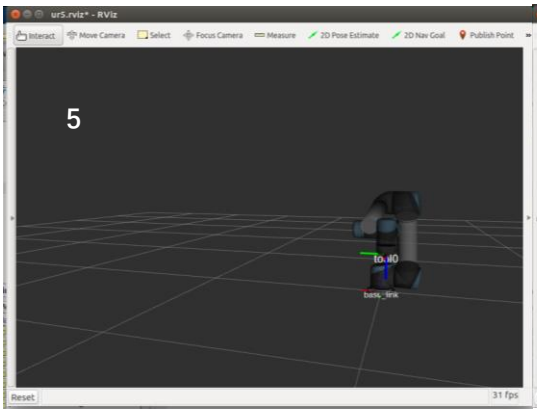
In the pick operation, UR5 first move to the location 10cm above the start position 'g_start1' from home position, open the gripper, move to the start position 'g_start' and then close the gripper to pick the wooden cube. In the place operation, UR5 move up first, then move to the location 10cm above the target position 'g_target1.' Then move down to the target 'g_target', open the gripper, and move up and go back to the home position. During the whole process, gripper keeps a safe distance from the table. Home position of Inverse Kinematics is default home configuration, namely, ur5.home whereas other control methods uses our custom configuration, which poses less susceptibility on singularities, namely Q_home_ctrl = [0.9818 -1.8475 1.6436 -1.3615 -1.5274 -0.4957]'. Below images of sequence illustrates how the pick-and-place task is performed.

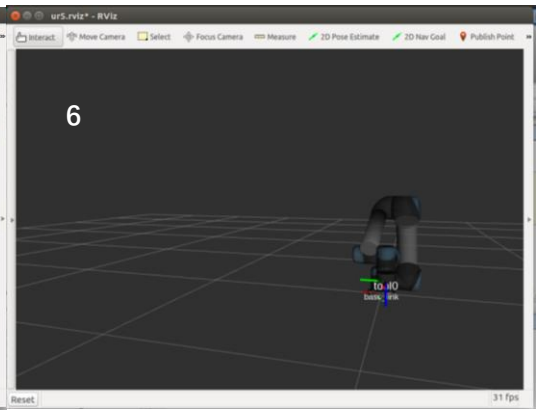

ur5.rviz* - RViz
Interact
Move Camera
Select
Focus Camera
Measure
2D Pose Estimate
2D Nav Goal
Publish Point
1
tool0
base_link
Reset
31 fps

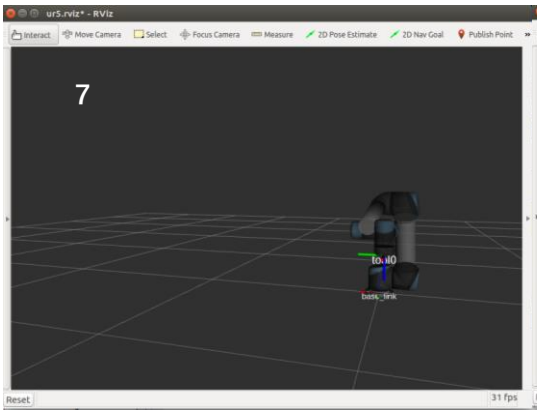

ur5.rviz* - RViz
Interact
Move Camera
Select
Focus Camera
Measure
2D Pose Estimate
2D Nav Goal
Publish Point
2
base_link
Reset
31 fps


ur5.rviz* - RViz
Interact
Move Camera
Select
Focus Camera
Measure
2D Pose Estimate
2D Nav Goal
Publish Point
3
base_link
Reset
31 fps


ur5.rviz* - RViz
Interact
Move Camera
Select
Focus Camera
Measure
2D Pose Estimate
2D Nav Goal
Publish Point
4
base_link
Reset
31 fps


ur5.rviz* - RViz
Interact
Move Camera
Select
Focus Camera
Measure
2D Pose Estimate
2D Nav Goal
Publish Point
5
base_link
Reset
31 fps


ur5.rviz* - RViz
Interact
Move Camera
Select
Focus Camera
Measure
2D Pose Estimate
2D Nav Goal
Publish Point
6
base_link
Reset
31 fps


ur5.rviz* - RViz
Interact
Move Camera
Select
Focus Camera
Measure
2D Pose Estimate
2D Nav Goal
Publish Point
7
base_link
Reset
31 fps


ur5.rviz* - RViz
Interact
Move Camera
Select
Focus Camera
Measure
2D Pose Estimate
2D Nav Goal
Publish Point
8
tool0
base_link
Reset
31 fps

Method

1. Invers kinematics

Calculating inverse kinematics of ‘g_start’, ‘g_start1’, ‘g_target’ and ‘g_target1’, results in 8 possible solutions for each configurations. To prevent collision, the 3rd joint and the gripper should be above the table, which means the range of 2nd joint angle is (-π , 0) and the 4th joint configuration g_joint4 (3,4) > 0.05. Below figures shows when the conditions described above do not satisfied.

Still there are several joints solutions. The solution is chosen which poses minimum norm of joint configurations between the current and the next step, *e.g.* norm(Q_start1_valid(:,k)-ur5.home). To guarantee that the robot safely reaches to the target, it pauses the same time as the time interval used in the function ur5.move_joints. There is about 1s during open/close the gripper.

2. **Rate control and Gradient control**

For both controller the gain K is dynamically changing. Both controller use σ_{min} as a measure of manipulability, the threshold for both is 0.0001. Both controllers do not feed q on the real robot every time when the next q is calculated. As a result, qk+1, qk+2, qk+3,⋯ are calculated (without moving the joints) until when it find q_desired. After finding the q_desired, the algorithm just feed this angle to the robot and thus move the robot to the desired target. The advantage of this method is that the robot will never stops or slow down between the start and target configuration since function ‘ur5.move_joints’ is called only one time and is used when the target configuration is achieved.

2.(1) Rate control:

$$q_{k+1} = q_k - KT_{step}[J_{st}^b(q_k)]^{-1}\xi_k$$

the threshold of norm v is 0.0003, norm w is 0.0005 to get reasonable accuracy, and the K is dynamically changing as a function of norm_tt:

$$K = 4.5 / 0.4 * norm\_tt - 9.75;$$

$$norm\_tt = norm(gdesired \backslash gst)$$

K is proportional to the norm_tt, K is large when norm_tt is large, K is small when norm_tt is small. This custom f(K) gave the robot a reasonable gain value, resulting in stable operation during the control. The time interval is constant as 5 seconds.

2.(2) Gradient control:

$$q_{k+1} = q_k - KT_{step}[J_{st}^b(q_k)]^{T}\xi_k$$

K is dynamically changing which is calculated based on the q_normk1, and q_normk. q_normk is the norm of desired q and current q. q_normk1 is the norm of desired q and next q. norm_q contains all q_normk-q_normk1 with all possible gain K from 0.1 to 1 which is a vector, We use the same approach on the position vector x, y, z and produces norm_pos. And overall weighting function is $a * norm\_pos + b * norm\_q$. We choose a=0.01 and b=1. Then optimal gain K is selected which produces the minimal value of the weighting function.

$$norm\_weighted = 0.01 * norm\_pos + norm\_q$$

Time interval is also dynamical changing as a function of :

$$time\_int = 20 * norm(gdesired - gst) + 2.5$$

and this time interval function produces reasonable amount of joint velocities.

Since gradient based controller tends to be slower than other methods, we implemented two controllers with high and low accuracy. Controller with high accuracy is used for intricate maneuver, for example, reaching to and grabbing the block, and controller with low accuracy is used for moving longer distance which requires less precision. Norm(v) and norm(w) is

0.015 and 0.015 in the controller with low accuracy and 0.0015 and 0.0035 for high accuracy.

Result

All results are recorded and can be found in [3].

1. Inverse kinematics: In simulation, the whole process of pick-and-place operation takes 57s, the UR5 robot uses 58s.
2. Rate control: In simulation, the pick-and-place operation takes 44s, the UR5 robot uses 46s. The range of dynamic changing K is (1.5 , 4.24).
3. Gradient control: In simulation, the pick-and-place operation takes 47s, the UR5 robot uses 47s. Range of K is (0.1 , 1), range of time_int is (2.4 , 2.8).

Extra

1. Pseudo inverse control

Control equation:

$$\Delta\theta = \alpha J^{\#}\Delta x + (I - J^{\#}J)(\theta_0 - \theta)$$

where '#' operator is Pseudo inverse.

For a small step Δx, minimize with respect to $\Delta\theta$ the cost function:

$$F = \frac{1}{2}(\Delta\theta + \theta - \theta_0)^T(\Delta\theta + \theta - \theta_0) + \lambda^T(\Delta x - J(\theta)\Delta\theta)$$

Where λ^T is a vector of Lagrange multipliers.

Solution:

(1) $\frac{\partial F}{\partial \lambda} = 0 \rightarrow \Delta x = J\Delta\theta$

(2) $\frac{\partial F}{\partial \Delta\theta} = 0 \rightarrow \Delta\theta = J^T\lambda - (\theta - \theta_0) \rightarrow J\Delta\theta = JJ^T\lambda - J(\theta - \theta_0) \rightarrow \lambda = (JJ^T)^{-1}J\Delta\theta + (JJ^T)^{-1}J(\theta - \theta_0)$

Insert (1) into (2)

(3) $\lambda = (JJ^T)^{-1}\Delta x + (JJ^T)^{-1}J(\theta - \theta_0)$

Insertion of (3) into (2) gives the result:

$$\lambda = J^T\lambda - (\theta - \theta_0) = J^T(JJ^T)^{-1}\Delta x + (JJ^T)^{-1}J(\theta - \theta_0) - (\theta - \theta_0)$$
$$= J^{\#}\Delta x + (I - J^{\#}J)(\theta_0 - \theta)$$

(a) Operating Principle: Optimization in null-space of Jacobian using a kinematic cost function $F = g(\theta)$ e.g. $F = \sum_{i=1}^{d}(\theta_i - \theta_{i,0})^2$

(b) Advantages of Pseudo inverse control:

Computationally fast

Explicit optimization criterion provides control over are configurations

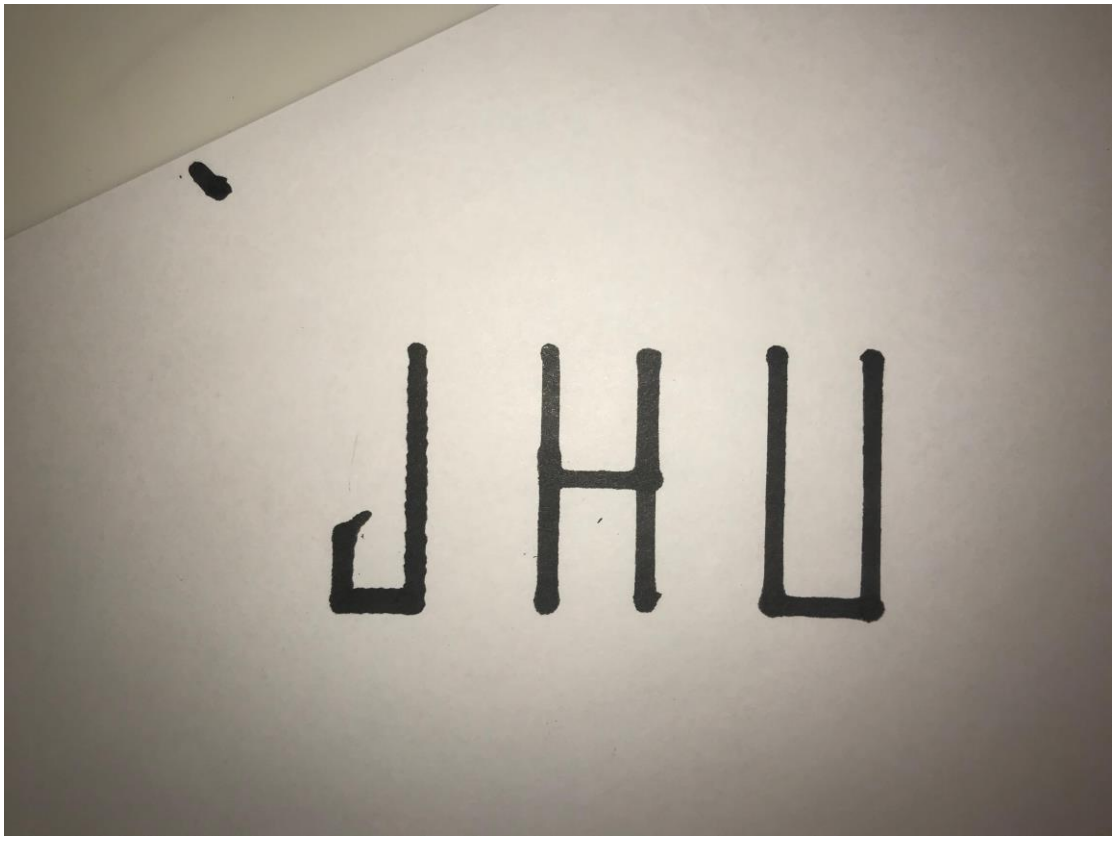
It can overcome singularity configuration

(c) Result simulation 48s, real 46s

The deteails and derivations can be found[1], [2].

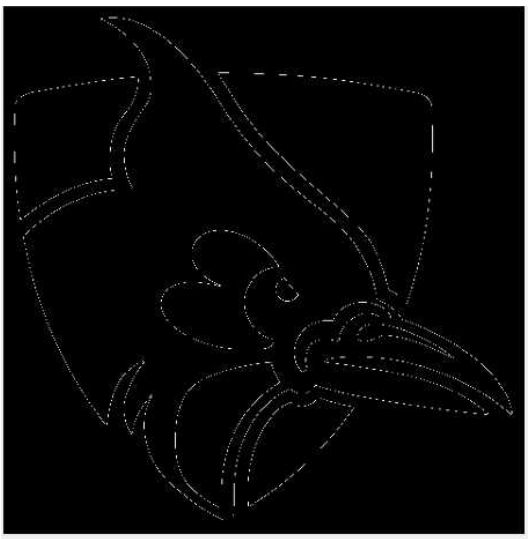
2. JHU

Pseudo inverse control was tested by writhing JHU on a plain sheet of paper. Each letters are printed first, then coordinates are hard coded into the main script. Reference coordinate is located on the top left corner of the letters. Below image shows the result.



3. Blue Jay

For an extra work, blue jay (JHU symbol) was drawn on a plain sheet of paper. Generic image of blue jay was found on the internet. Then edge detection algorithm is used which is built in function in Matlab. Below figure shows before and after edge detection.



After edge detection, coordinates are hard coded into the main script, usually connecting points are denser on the curved region and less dense on the straight line region.
To avoid possible singular configurations, pseudo inverse controller is used. Similar to the previous work, the robot was taught manually where is the starting configuration by pinpointing 0,0 (x,y) location. Below figure is shown to compare with the result.

Reference

1. C. A. Klein and C. H. Huang, "Review of pseudoinverse control for use with kinematically redundant manipulators," in *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-13, no. 2, pp. 245-250, March-April 1983.
2. https://homes.cs.washington.edu/~todorov/courses/cseP590/06_JacobianMethods.pdf
3. https://youtu.be/8-JqkNzUTkc