EN.530.646: Robot Devices, Kinematics, Dynamic and Control Final Project*

Jin Seob Kim, Ph.D., JHU

Out: 04/28/2021;
Due: 05/12/2021 midnight EST on the Gradescope

Introduction

The objective of this project is to simulate the UR5 robot using R-VIZ to perform a "move-and-place" task. In this project, you will implement a program that move the robot from a given start location to a given target location (here "location" means position and orientation). It is essentially the same task as "pick-and-place", except the gripper movement. The "move-and-place" task should be done as follows. First, the robot is at the home configuration. From there, the robot moves such that the end-effector (or tool frame) goes to the pose which is straightly above the start location (the distance can be chosen by you, and must be stated in the project report). Then the robot moves straight down to the start location, after which the robot moves back to the previous pose (right above the start location). After that, the robot moves to the pose which is straightly above the target location (again, the distance can be chosen by you), which is followed by moving the robot straight down to the target location. See Fig. 1 for a graphical illustration.

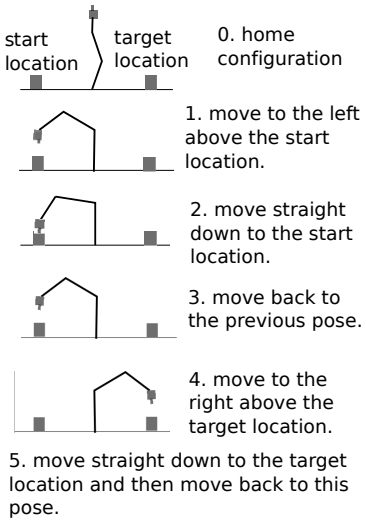


Figure 1: An illustration of the move-and-place task

The goal for the move-and-place task is to move the UR5 robot from the start position to the target position, both of which will be inside the workspace of the UR5, and close to the bottom surface. This makes the task equivalent to the pick-and-place task. Before moving to the start position, the robot should be at the home configuration, or an intermediate configuration that you define, which is equidistant from both of the positions (this can be a "home" configuration of the robot). Note that the intermediate (or home) configuration must be reasonably high from the surface. You have already implemented code to drive the robot to a given position and orientation (pose) in Cartesian space using Resolved Rate Control. Now you need to develop a complete program that moves the robot end-effector from the start location to the stop location using the UR5 and the end-effector.

*This document was originally developed by Prof. Noah Cowan, and modified by Dr. Jin Seob Kim

Main Task: Move-And-Place Task

You will need to create three implementations of the move-and-place task. The start and target locations should be given as inputs so that TA's can determine (let's call it as "teach the robot").

The recommended way to do this is use **pause** or another function like **waitforbuttonpress** in Matlab, which will be needed when you move frame and create frame which will communicate with Gazebo and RViz. Be sure to test your program on several different configurations to be sure it is robust.

Using the frame locations that you "teach", you will plan a trajectory for the robot in Cartesian space, i.e. in $SE(3)$. The process should begin and end with the robot in some "home" configuration that you decide is best (the intermediate frame is a good candidate). You must implement the control trajectory in three different ways:

1. **Inverse kinematics:** Using inverse kinematics, move the robot along the desired Cartesian path by finding the corresponding path in joint space.
2. **Resolved-rate control using differential kinematics:** Find the desired velocity (or small, differntial Cartesian offset), and "pull that back" through the inverse of the Jacobian to find the joint space velocity (increment) that moves you in the right direction.
3. **Transpose-Jacobian control:** Do the same thing, but now use the *transpose*, not inverse, of the Jacobian matrix.

In all three tasks, you have to implement the safety check such that the robot does not hit the bottom surface (in real situations, the collision between the end-effector and the table surface), joint limits, and so on.

Extra Task

After you complete the main task above, you are all welcome to try any interesting idea which involves robot motion using inverse/differential kinematics. You may use available resources or third party libraries (with approval of the instructor or the TAs due to safety concerns). This will lead to the extra credits. Remember to COMPLETE THE MAIN TASK before becoming adventurous with the further credit!! Contact TA's if you need extra help to setup a special environment.

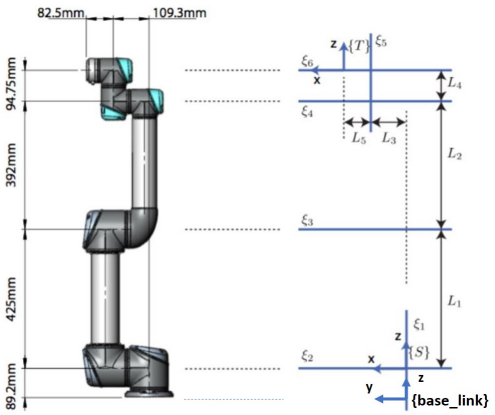
Inverse Kinematics

The function "ur5InvKin.m" will be provided that calculates all eight possible solutions to the inverse kinematics for the UR5 at a given frame in the workspace. Several of the solutions will not be practical to use and you will need to select the "best" solution to move the robot to during your "Inverse kinematics" control. Explain this in your report.

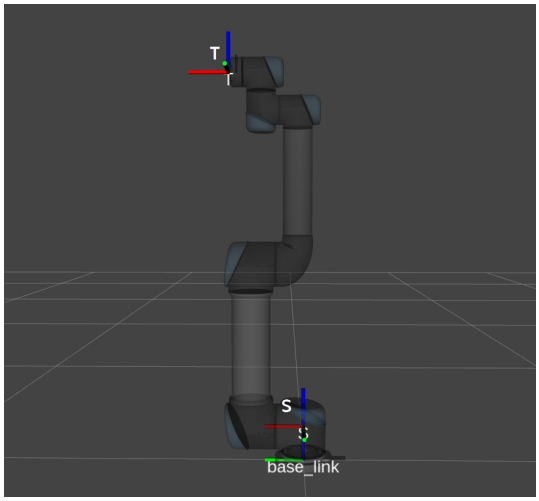
1. Input: g_{06} (or T_0^6 as defined in [1]), the 4x4 homogeneous transformation matrix.
2. Output: A 6×8 matrix, each column represents one possible solution of joint angles.

The function computes the solutions following the Denavit-Hartenberg (D-H) convention described in [1]. Therefore you need to correctly define the (constant) transformations from the "0" frame in [1] to the "S" frame and from the "6" frame in [1] to the "T" frame. The "S" and "T" frames are the same convention used in the problem sets and previous labs. They are illustrated again in Figure 2 for reference.

A new RVIZ "urdf" model (ur5.urdf.xacro) of the robot will be provided that includes these frames for clarity. To use it simply replace the old file ur5 urdf.xacro file in catkin ws/src/universal robot/ur description/urdf/ and relaunch RVIZ. Also note that if you move the joints on the UR5 to `[0 0 0 0 0 0]` the robot will be will be placed at the Ryan Keating home position.



(a) Description of "S" and "T" frames



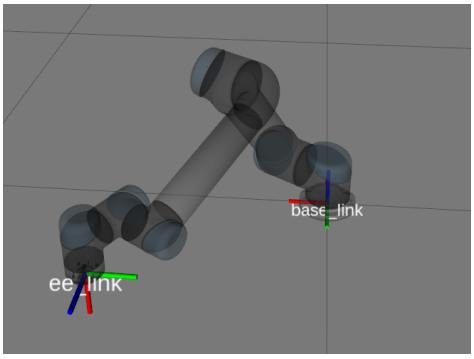
(b) New RVIZ model with "S" and "T" frames

Figure 2: Possible solutions to the inverse kinematics for simulated start and target frame described above

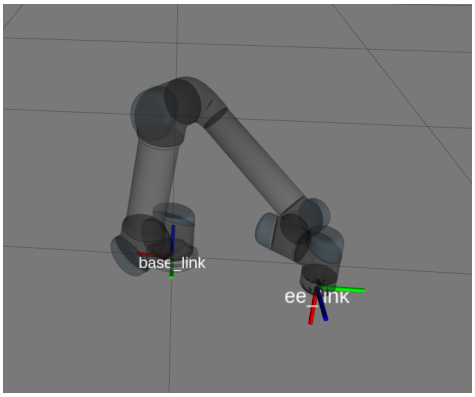
Start and Target Locations

The start and locations will be specified near the surface. You may have to adjust the thresholds on your control algorithms to achieve accurate placement. To test your code in RVIZ use the following start and target frames that were recorded in the setup shown Figure 3:

$$g_{st\_start} = \begin{pmatrix} 0 & -1 & 0 & 0.47 \\ 0 & 0 & 1 & 0.55 \\ -1 & 0 & 0 & 0.12 \\ 0 & 0 & 0 & 1 \end{pmatrix}; \quad g_{st\_target} = \begin{pmatrix} 0 & -1 & 0 & -0.30 \\ 0 & 0 & 1 & 0.39 \\ -1 & 0 & 0 & 0.12 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$



(a) Possible start pose



(b) Possible target pose

Figure 3: Possible solutions to the inverse kinematics for simulated start and target frame described above

Other Hints

1. You should have a process, or controlled procedure for achieving the desired "move-and-place" performance.
2. It is recommended to enable some kind of debugging output, or some kind of callback (print the current status when a certain key is pressed), it will expedite your simulation.

3. Modularize your code.

Deliverables

1. Submit a single zip file titled "FinalProject_TEAM#_MemberInitials" (e.g., FinalProject_TEAM2_DL_ZH, if Dimitri and Zhuohong are a team 2) to the gradescope "Final Project: Code and Movies", which includes all codes and movie files. Your codes must include a main script called ur5_project.m. In this script, the user should be able to choose the method of control: IK-based, DK-based, or gradient-based. Also make sure that the TA's can set their own the start and the target locations in your main script. Check to verify that the files you hand in run. You also have the option to create new custom Matlab m-files for this project, just make sure that they are well documented and all the necessary files are included in the zip.

   Also, make sure your code is clearly documented with sufficient comments to make it easy to understand. Sloppy code will not receive full credit.

   All codes needed to run your program should be in this submission (including any code you made for previous labs!).

2. Movie files (.mp4 or equivalent) for three control algorithms (hence at least 3 movie files). In this case, the start and the target locations must be clearly written in your report.

3. A PDF report specifying your algorithm (workflow), simulation results, and all other important considerations, as well as workload distribution between team members. This report must be submitted to the gradescope "Final Project: Report". Name the file as "FinalProjectReport_TEAM#_MemberInitials". In particular, report errors between the desired location and the actual location during the simulation (this corresponds to both start and target locations). For example let $g_d = (R_d, \mathbf{r}_d) \in SE(3)$ be the desired start location, and let $g = (R, \mathbf{r}) \in SE(3)$ be the actual start location that the robot end-effector reaches in your simulation. Then report two errors for this start location as

$$d_{SO(3)} = \sqrt{\mathrm{Tr}\left((R - R_d)(R - R_d)^T\right)}$$
$$d_{\mathbb{R}^3} = \|\mathbf{r} - \mathbf{r}_d\|.$$

   Do the same for the target location as well. These errors are applied to all three algorithm cases.

4. In your zip file, also include a movie file for your team's extra task (its name should include "extra_task").

5. Submission should be done by only one member of your team. Hence the report must clearly state team members, with workload distribution as mentioned earlier.

References

[1] Ryan Keating, UR5 Inverse Kinematics. 2014