**Final Project: Report**

Due: Dec.16th, 2020

Eugene Lin

Jiaqi Yang

Yishun Zhou

1. Algorithms/Workflow
   1. IK-based

The transformation matrix from base frame **base** to the gripper frame **tool0** can be defined as :

Where is the transformation from the base frame to frame S, is the transformation represents the transformation to the desired location and is the transformation from frame T to the gripper frame **tool0.**

Then use the function “ur5InvKin.m” to get a 6X8 matrix. Each column represents a possible solution of joint angles. The best solution should satisfy several following requirements:

1. **Theta2** should be between 0 and -Pi to prevent the robot from hitting the ground.

2. **Theta3** should be non-negative and does not equal to PI to prevent the robot from hitting itself.

3. **Theta1**, **theta4** and **theta5** should be limited within a certain range to prevent over-rotating.

There is usually only one solution that meets all the requirements above. If multiple solutions exist, compare and find out the one which has the least changes of six angles. If no solution satisfies, it will terminate and return.

For the safety check, one location is valid if and only if this location is placed:

1. Above the ground (gdesired(3,4) > 0).

2. Inside the hemisphere with a radius of L1+L2+L4 where the end-effector can reach.

3. Outside the hemisphere with a radius less than L1+L2+L4 (pick 10 cm in this lab) to prevent the robot from hitting itself.

*\*Two hemispheres share the same center located at the bottom of the robot.*

* 1. DK-based

This is based on the resolved-rate control introduced in class, which used the following equation to update the next joint configuration that the robot should go to.

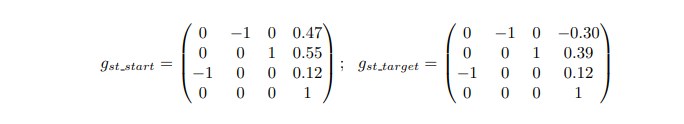
* 1. gradient-based

This method is similar to the DK-based one except that transpose of jacobian is used.

\*Note:The distance we set above the start and target position is 0.2m.

1. Simulation Results

We used the same target and start poses in the instruction file. See the videos for details.



1. Discussion

We tuned the gain and number of iterations for the DK-based method using inverse of jacobian and the gradient-based method using transpose of jacobian. It turned out that it took longer to converge to the target using the gradient-based method than the DK-based method. In order to save time for simulation, we put a limit on the number of loops the resolved rate control methods, which makes the gradient-based control loop end before the error threshold is within the limit. That is why using the third method gives you more errors.

Using the start and target poses given in the instruction file, we compare the error of start and target locations in the following table.

| Error of start and target locations | IK-based | DK-based | gradient-based |
| --- | --- | --- | --- |
| Error of start orientation (m) | 1.5983e-05 | 2.3312e-05 | 0.0024 |
| Error of start position (m) | 7.7382e-06 | 0.0011 | 0.0083 |
| Error of target orientation (m) | 1.1279e-05 | 1.0778e-05 | 0.0025 |
| Error of target position (m) | 3.6272e-06 | 0.0011 | 0.0050 |

As we can see from the table above, the inverse kinematics-based method generates the least errors.

1. Workload Distribution

Jiaqi worked on IK-based control and Eugene worked on DK-based and gradient-based control. Yishun worked on the main script and the report.