530.646: Robot Devices Kinematics, Dynamics, and Control

Final Project: Control Algorithms for UR5

Spencer Witte, Peiyao Zhang, Daniel Hsu

1. Deliverables

- Submit **a single zip file** titled "FinalProject TeamNumber MemberInitials" to the blackboard, which includes 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.
- A **PDF report** specifying your algorithm (workflow), experimental results, workload distribution, and all other important considerations.
- We will set up a **demo** time (see the poll), so that each team show their output to the instructor and the TAs.

2. Objective

The objective of the project is to demonstrate the effectiveness of using 3 different types of control algorithms (Inverse Kinematics, Resolve Rate, Gradient based) in executing a place-and-mark-with-intention task.

3. Workflow

a. Preparation

1. ur5 initializes to home position.
2. Move arm to start cup, record position in MATLAB
3. Move arm to target cup, record position in MATLAB
4. Calculate intermediate position, add vertical offset
5. Define vertical offset to cup
6. Define gesture movement
7. User inputs which controller they want to use via command window

b. Moving the Arm

8. Return to intermediate position before starting.
9. Gesture towards the start cup
10. Move to a vertical offset above the start cup using specified control algorithm
11. Move down to draw the dot
12. Move to a vertical asset above the target cup using specified control algorithm
13. Move down to draw the dot
14. Return to intermediate position

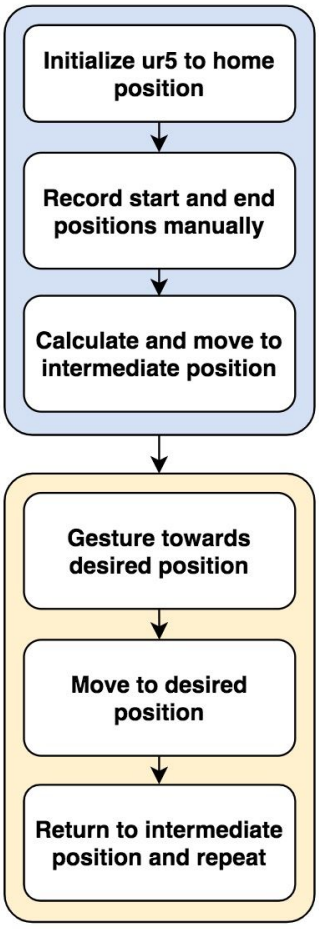

Figure 1: High level overview

4. Control Algorithms in Detail

a. Inverse Kinematics - Daniel Hsu

The inverse kinematics algorithm has already been provided to us. To implement it for our application, we wrote the IK_based.m code that takes into a transformation *g* and outputs the corresponding joint configuration of the end effector. Since the provided inverse kinematics algorithm returns all 8 possible configurations, we decided to choose the one that is closest its current position (by finding the min() of the norm of the target and current positions).

b. Differential Kinematics (Resolved-Rate Control) - Spencer Witte

The controller consists of a loop that iteratively calculates the desired joint velocities and computes the joint positions one time step ahead by using the inverse Jacobian. The controller also checks to see if the new joint angles cause a singularity issue, and if not, commands the robot to move to that position. The constant K is the gain constant. Smaller values of K will increase precision of the arm but decrease the speed, and larger values of K will increase the speed of the arm but decrease precision. Additionally, threshold values were coded into the controller to determine if the arm had reach its goal position, and could be tuned to increase the precision. If our code did not have singularity detection, some of the joint velocities would become very large when the arm moved near the singularity configuration.

c. Gradient-based (Transpose Jacobian) - Peiyao Zhang

The gradient-based transpose jacobian method is based on the one discussed in the 1988 paper [3]. In theory, the transpose jacobian approach should be less computationally expensive than the differential kinematics (which uses the inverse jacobian) and should be able to apply to any type of robot regardless of its geometry compared to the differential kinematics approach (jacobian may not have inverse but always have transpose). The algorithm is implemented in Dr_based.m which is essentially identical to DK_based.m except that the inv(J) is changed to transpose(J) and tuned with slightly different parameters.

5. Experimental Results

We were able to successfully implement all three controllers to perform the “pick and mark with intention” task using the positions we “teach” the UR5 arm.

Below is a table that shows how long it took for our controllers to execute their motion. Note this was done with chosen start and target positions and would produce different results with different positions.

Below are photos showing each stage of the arms movement.

Figure 2 : teaching positions (start cup left photo, target cup right photo)

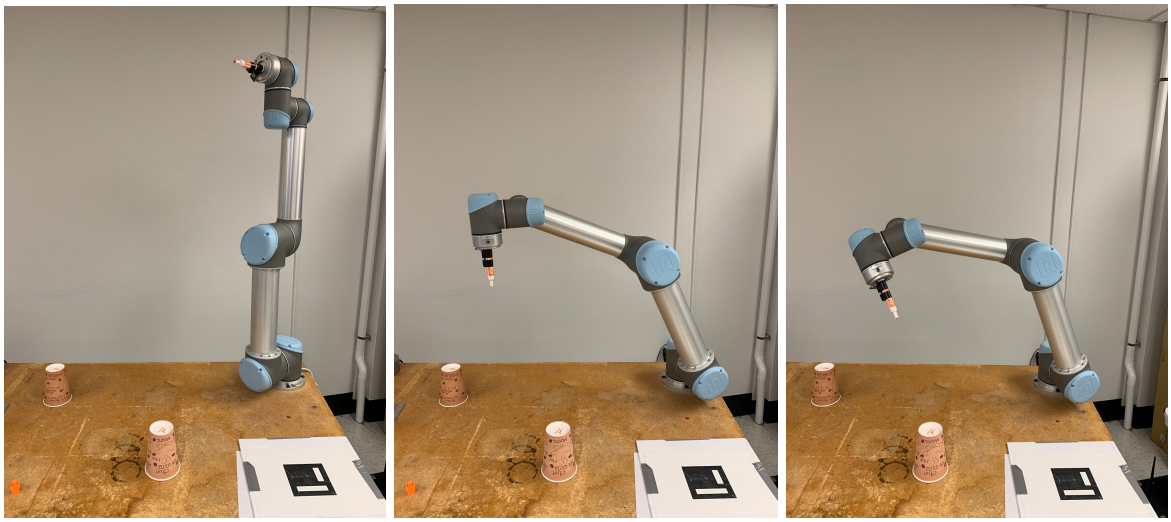

Figure 3 : (left) home position (middle) intermediate position (right) gesture towards start

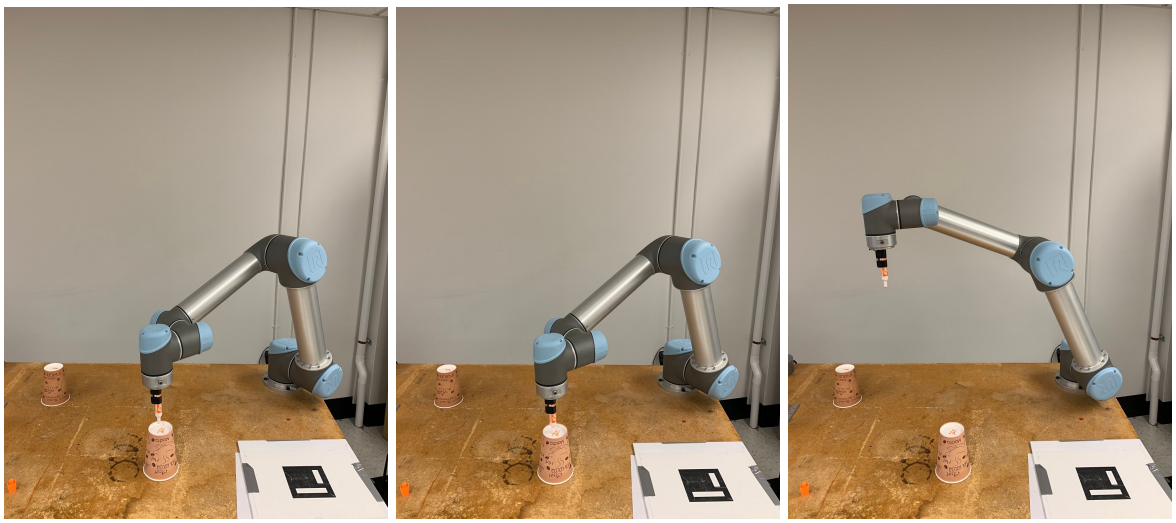

Figure 4 : (left) offset from start (middle) moves down to mark (right) moves back to intermediate

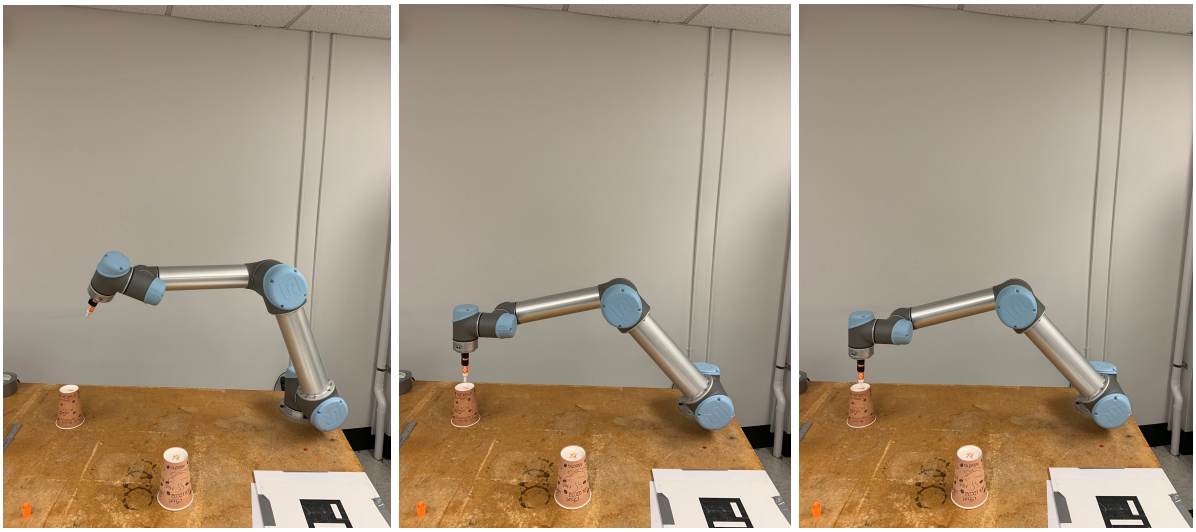

Figure 5 : (left) gesture towards end (middle) offset from end(right) moves down to mark

6. Discussion & Conclusion

Despite the success of our control algorithms in achieving the place-and-mark-with-intention task, we believe there are a few improvements that can be made. First we noticed that in the differential kinematics based controller and gradient based controller “stutters” as it moves. We believe this is because these controller are iterative methods where the controller calculates several waypoints along the path to its goal position and uses the move_joint command to drive the arm to each waypoint. The “stuttering” occurs because the move_joint command is motion profiled and decelerates and stops at each waypoint. The motion profile is depicted in **figure_** below. This could be solved for by using a function that does not have the motion profiling implemented in the command, however, we would then want to implement it into our controller.

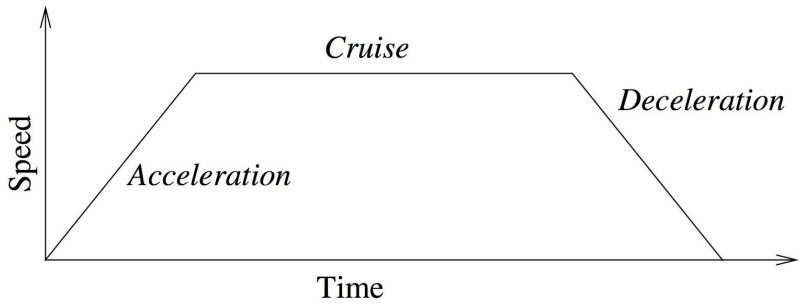

Figure 6 : ur5 speed profile for a motion [1]

The second thing we noticed was that the gradient controller was very slow. It may be possible to speed it up with further tuning, however, we were having trouble tuning it to be faster.

7. Extra Credit

For extra credit, we decided to apply our control algorithm to a drawing task. The robot (with a marker attached at the end effector) is moved to the starting location manually such that the marker is in contact with a piece of paper taped on the table. Then we apply the inverse-kinematics controller to draw a star and a circle around it. We seperated the circle into 30 segments to find a balance between the drawing time and the smoothness of the circle. The drawing result is attached below.

Figure 7 : extra credit task in progress

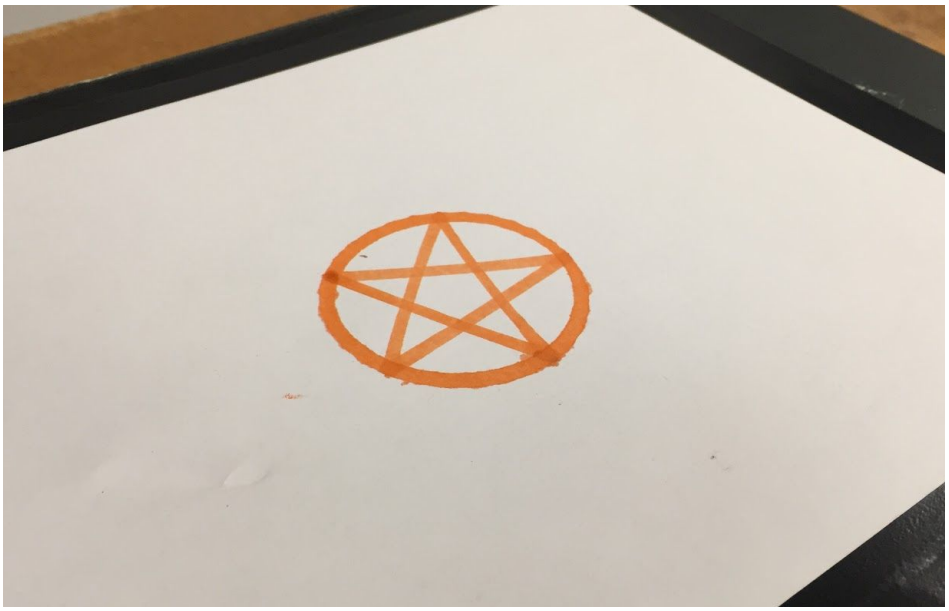

Figure 8 : drawing result

Figure 9 : end effector setup

8. File Directory

Provided:

ur5_interface.m:	ur5 member functions
tf_frame.m:	maintain a frame in tf in ROS
ur5InvKin.m:	Ryan Keating's inverse kinematics solution [2]
ur5BodyJacobian.m:	calculates body jacobian

Important functions:

ur5_project.m:	main function to run
IK_based.m:	inverse-kinematics algorithm
DK_based.m:	differential kinematics algorithm (resolved-rate)
gradient_based.m:	gradient-based algorithm
runController.m:	runs either of the 3 control algorithm based on user input

ur5*fwdKin.m:* forward kinematics
manipulability.m: calculates and warns user when near singularity

Helper functions:
SKEW3.m: transform 3x 1 vector into 3 x 3 skew-symmetric matrix
getXi.m: outputs twist coord. corresponding to a homog. transformation
ROTX.m ROTY.m ROTZ.m : rotation matrices

9. References

[1] *ur5 User Manual (https://www.usna.edu/Users/weapron/kutzer/_files/documents/User%20Manual,%20UR5.pdf)*
[2] Ryan Keating, UR5 Inverse Kinematics. 2014
[3] Chiacchio, Pasquale, and Bruno Siciliano. "A closed-loop jacobian transpose scheme for solving the inverse kinematics of nonredundant and redundant wrists." Journal of Robotic Systems 6.5 (1989): 601-630.