EN.530.646

Robot Devices, Kinematics, Dynamic and Control

Final Project Report

Introduction

The objective of this final project is to use the UR5 robot to accomplish the place-and-draw task under three types of control methods: inverse kinematics (IK), resolved-rate control (RR), and transpose-Jacobian control (JT). For IK and RR control, the trajectory turns at two intermediate locations before reaching the goal location. For JT control, a straight-line segment is drawn between them after the start and target locations are taught.

The extra task is to apply RR control to write the letters “RDKDC”.

Workflow and Method

To accomplish the place and draw task safely, after taught start and destination points configurations, UR5 is moved to the ready position which is 10cm above the start points with a similar configuration of joint. For IK and RR, it then moves straight down to the start point and then the middle point1, middle point2, and finally the destination points in sequence as required by the assignment (turning in two right angles at the middle points). For JT, it moves directly from the start to the destination. Fig 1. shows the drawing procedure in sequence.

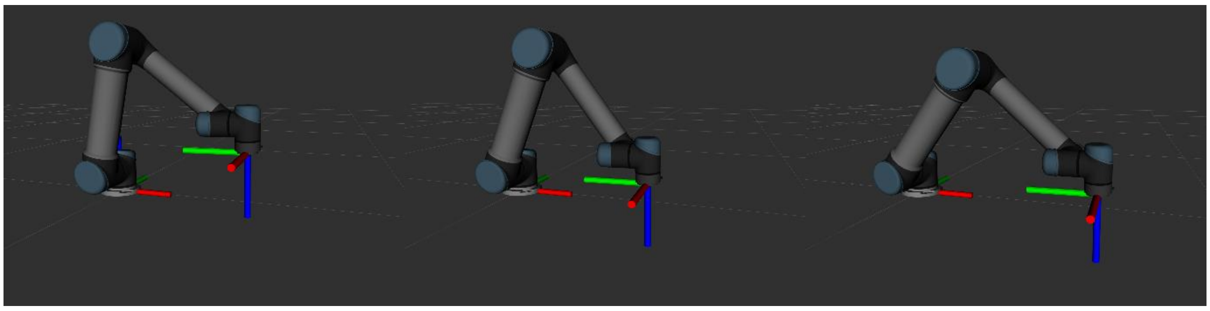

Fig 1. Drawing procedure: (a) ready location; (b) start location; (c) goal location.

The error of orientation and position are reported at the start and destination point, and the completion time of task is counted, as shown in the Result section. Concerning safety, every time the joints are moved, a safety check function (*safety_check.m*) will be executed to check the collision situation (the elbow and wrist joints collision with drawing plane) and excess of the joint limits. It will abort the execution before that happens.

1. Inverse kinematics (IK)

Inverse Kinematics control is implemented relying on the API provided for IK (*ur5InvKin.m*). *test_ur5InvKin.m* was executed to test the accuracy of their ur5FwdKin.m functions. The result is shown as below:

```
result:
test_ur5InvKin
The Average Joint Errors are: [0.002, 0.011, 0.003, 0.008, 0.001, 0.017]
```

As for the IK control mechanism, the angle and position are discretized between the start and destination points by interpolation using *linspace*. The angular and positional step gain and speed are determined by two pre-determined hyperparameters, *move_time* (time between each intermediate point) and *resolution* (every step will go resolution mm by move_time). Moreover, because the *ur5InvKin.m* returns eight (8) possible solutions so there is the *theta criteria.m* function to select the optimal solution to be the next configuration moving to.

2. Resolved-rate control (RR)

A discrete-time resolved-rate control is also implemented in the project (*ur5RRcontrol.m*). For the RR control mechanism, it involves the calculation of inverse Jacobian and unscaled twist to achieve a desired end-effector pose. In each time step Δt, we calculate the unscaled twist as $e^{\hat{\xi}_i} = g_{st}^{-1} \cdot g_{st}$, and we update the next joint configuration as follows.

$$\overline{q}_{i+1} = \overline{q}_i - K \cdot \Delta t \cdot \left(J_{st}^{b}(\overline{q}_i)\right)^{-1} \cdot \overline{\xi}_i$$

$\overline{q}_i$ is the current joint configuration and $\overline{q}_{i+1}$ is the next joint configuration. $\left(J_{st}^{b}(\overline{q}_i)\right)^{-1}$ is the inverse of the current body Jacobian and K is the gain on the controller. We update the joint configuration in a loop until we reach the desired end-effector pose. To achieve a smooth and stable movement, we choose $K = 0.25$ and $\Delta t = 0.4s$.

3. Transpose-Jacobian control (JT)

Like RR, the joint coordinates are calculated discretely based on the equation

$$\overline{q}_{i+1} = \overline{q}_i - K \cdot \Delta t \cdot \left(J_{st}^{b}(\overline{q}_i)\right)^{T} \cdot \overline{\xi}_i$$

Here, since JT control is slower than the other two methods, we choose $K = 1$ to speed up the speed up the drawing process.

To make the end-effecter follow a straight path, we interpolated 49 points on the desired trajectory, dividing the line into 50 segments with equal length. The improvement in tracking is significant, at the cost of a longer operation time.

Result

To calculate errors between the desired location and the actual location during the simulation, we denote the desired start/goal location as $g_d = (R_d, p_d) \in SE(3)$, and the actual start/goal location as $g = (R, p) \in SE(3)$. Then the error is calculated from

$$err_{SO(3)} = \sqrt{Tr\left((R - R_d)(R - R_d)^T\right)}$$

$$err_{R^3} = 100 \cdot norm(p_d - p)$$

where the 100 is introduced to convert the unit from m to cm. The simulation errors for all three controls, and the operation time, are listed in Table 1.

Table 1. Simulation error and operation time

	Start location		Goal location		Operation time (s)
	$err_{SO(3)}$	err_{R^3} (cm)	$err_{SO(3)}$	err_{R^3} (cm)	
IK	1.1510e-15	0.0159	8.2846e-16	0.0170	73
RR	2.4665e-11	0.1968	0.0014	0.1978	43
JT	0.0016	0.1991	0.0016	0.1966	510

Extra Task

Our extra task is to write the letters RDKDC. First, we need to choose the starting and ending points for writing. Then, we will draw an underline to indicate that we will complete the text along this line. The size of the RDKDC letters and the spacing between them will be determined by the writing range set at the beginning. In fact, we have coded the drawing methods for all English letters, and the content we write can also be other text. The algorithm for controlling the movement of the robotic arm during this process is Resolved-rate control. The results are shown in Fig 2.

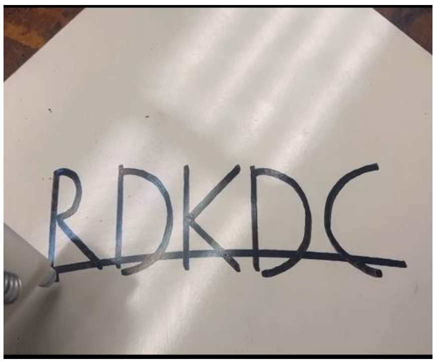

Fig 2. Extra task result

Reference

1. Murray, Richard M., et al. A Mathematical Introduction to Robotic Manipulation, Taylor & Francis Group, 1994. ProQuest Ebook Central.