RDKDC Final Project Report

Pick and Draw Task with UR5

Group 4
Pranav Bajaj
Alex Alessi
Chen Xiang

Content

1. Algorithm

1.1 Controllers

1.1.1 ur5InvKinControl()

The inverse kinematics controller relies upon the analytical ur5 inverse kinematics solution. Given a start and end location represented as an SE3, a series of intermediate points (SE3's) are generated via linear interpolation in the x, y, z coordinates. The inverse kinematics solution is applied to these points, generating a series of sequential joint configurations which trace a linear path in cartesian space. To select from the 8 potential joint configuration solutions for a given SE3 we choose the joint configuration which is "closest" to the robots current pose. Closest means the L2 norm between a given joint configuration and the robot's current pose is minimal. Executing this final array of joint configurations using the ur5 controller produces the desired behavior.

1.1.2 ur5RRControl()

The resolved rate controller relies upon the resolved rate control update equation given by

$$\boldsymbol{q}_{k+1} = \boldsymbol{q}_k - K\,T_{\text{step}}\left[J_{st}^{b}(\boldsymbol{q}_k)\right]^{-1}\boldsymbol{\xi}_k$$

Given a start and end configuration we enter a control loop which will execute until the current rotation and position error are below a finite threshold. For each iteration of the control loop a body jacobian, error twist and gain value K are calculated. K is determined dynamically such that the maximum joint velocity of the ur5 is held constant at any given time. Using these values a "step" is taken and new joint configuration is calculated. The manipulability of this new joint configuration is measured using the inverse condition number of the jacobian to assure it is not singular or nearly singular. Given the manipulability of the proposed joint configuration is above a given threshold it is sent to the ur5 controller and a new error value is calculated.

1.1.3 ur5JTControl()

The Jacobian Transpose controller relies upon the JT control update equation given by

$$\boldsymbol{q}_{k+1} = \boldsymbol{q}_k - K\,T_{\text{step}}\left[J_{st}^{b}(\boldsymbol{q}_k)\right]^{-\intercal}\boldsymbol{\xi}_k$$

After replacing the update equation the control loop for jacobian transpose control is identical to that of the resolved rate control with the exception of the use of a smaller time step.

1.2 Main Script

1.2.1 ur5_project.m

- Interactive main script file.
- After running, user will be asked to select the controller type: RR-based, IK-based and TJ-based
- After selecting the controller, the user will be asked to move the robot to the start location manually.
- After moving the robot to the start location, to record the start location click on the prompted window.
- After the start position gets recorded, do the same for the target location.
- After the target location is recorded, the robot will move back to the home configuration and based on the controller selected robot will trace a trajectory from start location to the target location.
- After reaching the target, the robot will move back to the home configuration.

2. Important considerations

2.1 Forward and Inverse Kinematics

2.1.1 ur5FwdKin.m:

1. Zero configuration is different between real robot and assignment 6 so we add the pi/2 in second and fourth joints to correct that.
2. Base frame is different between real robot and assignment 6 so we **left** multiply a transform matrix to correct that:

```
0.0000   -1.0000         0         0
1.0000    0.0000         0         0
     0         0    1.0000    0.0892
     0         0         0    1.0000
```

3. Tool frame is different between the real robot and assignment 6 so we **right** multiply a transform matrix to correct that and also multiply the transform matrix between tool frame and marker frame.

```
 0.0000         0    1.0000    0.1223
-1.0000    0.0000    0.0000    0.0000
-0.0000   -1.0000    0.0000    0.0490
      0         0         0    1.0000
```

2.1.2 ur5InvKin_wrap.m:

1. Since the provided inverse kinematic takes inputs as the assignment 6 configuration so we have to undo all the correction terms in forward kinematic then feed the gst to the provided inverse kinematics to get the theta.
2. Since the provided inverse kinematic has different zero configuration as to the real robot so when we get the joints value we have to so we subtract pi/2 in second and fourth joints to correct that.

2.1.3 intermediatePoints.m

1. This function is used to divide the trajectory between start and target into 3 segments.
2. First assumption: drawing plane is x-y plane.
3. Second assumption is that the z-coordinate and orientation of the target position is consistent with those of the starting position. This assumption is reasonable because even if the recorded target location has a different orientation or z-coordinate, we can treat it as being identical to the start position. This is because our objective is to move the Marker on a level surface, and the exact rotation around the x and y axes and shift is z positon is not critical as long as the tip of the Marker is in contact with the paper.
4. Function takes two argument: (s,t). s: {x,y} start coordinates; t: {x,y} target coordinates.
5. return: Array of start and end {x,y} location of the 3 line segments.

2.1.4 optimalJoinConfig.m

1. Given an array of joint configuration vectors this function will return the single joint configuration vector whose L2 distance to the current joint configuration is minimal.

2.1.5 interp.m

1. Given two SE3's this function will return "steps" number of intermediate SE3 points by performing linear interpolation in the x,y,z coordinates.

3. Results

3.1 Simulation

3.1.1 Error Terms

All error terms were measured in UR5 simulation as stated in final project pdf

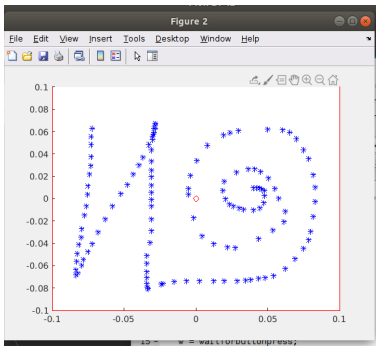
		Inverse Control	RR Control	JT Control
Start Configuration	**Rotation Error (rad)**	2.0556e-05	3.6158e-05	2.3456e-05
	Position Error (m)	6.3902e-06	1.5956e-05	3.3366e-05
Target Configuration	**Rotation Error (rad)**	4.4675e-15	9.9747e-05	0.0099
	Position Error (m)	4.1633e-17	0.0036	0.0044

4. Extra Task

4.1 Drawing figures on paper

Step:

1. Run drawfigure.m file
2. When empty figure window comes, move the robot to the starting point
3. **Click** the empty figure, then draw on the new figure



4. when done drawing, press **Enter** in keyboard

4.1.1 Implementation details:

First we record the initial start joints value and use forward kinematics to record that as initial configuration. Then we create a figure as canvas for the user to draw and record every three 2D points the user drawed, then we normalize the points list by subtracting every point in the list by the first point. After that, we treat the points list as offset to add to the starting points while keep the orientation and z axis the same. Specifically, we will have a sequential configuration list with x,y position being different. After that we feed this list to the inverse kinematics control to draw the figure the user drawed.

5. Workload Distribution

Pranav Bajaj:

- intermediatePoints.m
- optimalJointConfig.m
- Ur5_project.m
- manipulability.m
- UR5 inverse and forward kinematics solutions

Chen Xiang:

- Drawfigure.m
- ur5BodyJacobian.m
- ur5InvKin_wrap.m
- ur5FwdKin.m
- UR5 inverse and forward kinematics solutions

Alex Alessi:

- vecDraw.m
- ur5InvKinControl.m
- ur5RRcontrol.m
- ur5JTcontrol.m
- Interp.m

All team members contributed equally to report, debugging, testing and high level planning.