Robot Devices, Kinematics, Dynamic and Control (EN.530.646) Final Project

Kinjal Shah, Divya Ramesh, Wei-Lun Huang, Ge Sun

I Main Task: Place and Mark with Intention

Please view the linked video to view the performance of the Pick and Place with Intention task.

In order to perform the place-and-mark-with-intention task, we needed to perform the following steps:

1. **"Teach" the robot:** In order to "teach" the robot each pose, we used the UR5 interface to manually position the robot. We then used the `ur5.get_current_joints` to extract the joint angles for that position and our ur5FwdKin.m file to calculate the transformation.
2. **Establish new home:** Establish configuration that avoids the singularities met by the ur5.home position
3. **Intention Pose:** Display direction "intention" by establishing intention pose
4. **Perform control trajectory:** Inverse Kinematics, Resolved Rate Control, Gradient Control to achieve the following motion profile

    i. From intention pose, move to a point above the first mark position
    ii. Move down to the mark 1 position
    iii. Move back up to the above mark 1 position
    iv. Move over to the above mark 2 position
    v. Move down to the mark 2 position
    vi. Move back up to the above mark 2 position
    vii. Go to ur5.home configuration

    We discuss our algorithms and results for each of these components in the sections below.

A Initialization: Intermediate Pose, Intention Pose, "Above Mark" position

Intermediate Pose

Since the UR5 robot meets multiple singularity conditions in its ur5.home configuration (Appendix A), we needed to establish a new "home" position for the UR5 to go to prior to and after each motion. We initially began by having the robot "bow" towards the location of the motion, by moving joints 3, 4, and 5. However, we found that via this intention mechanism, there were certain areas in the robot work-space that would cause the robot to hit singularities during its motion. Therefore, we decided to manually move the robot to a neutral position in the work-space (**Figure 1B**) and collected the joint angles. Prior to performing any motion, we first have the robot move from ur5.home (**Figure 1A**) to this new neutral home position.

Intention Pose

From the new neutral home (we will call this the "intermediate pose" from now on), the robot then indicates the intention of motion, depending on the location of the first target in the work-space (**Figure 1C**). Our current process for setting the intention pose is to adjust joints 4 and 5 by a small angle based on the positional relationship between the intention pose and the intermediate pose.

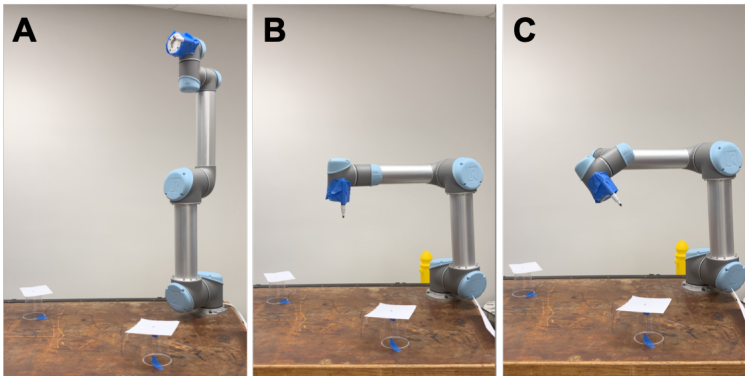

Figure 1: Depicts (A) ur5.home configuration (B) Intermediate Pose (C) Example Intention Pose

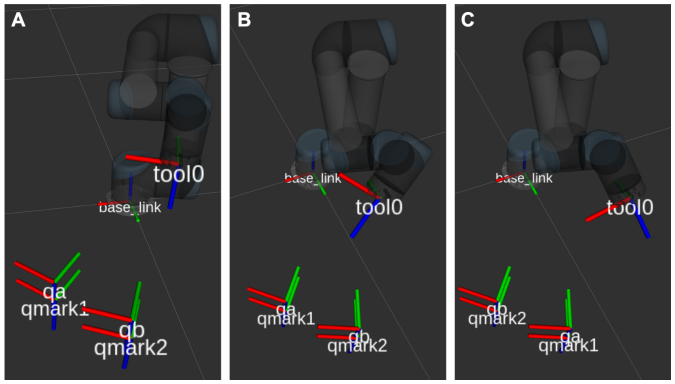

Figure 2: Depicts functionality of intention pose. When the robot is asked to go to "qmark1" first, from the intermediate pose (A) it tilts to the left as its first move (B). When asked to go to qmark2 first, it tilts to the right as its first move (C).

Process to establish intention pose:

1. Measure relationship between home configuration x,y position of the `tool0` frame and the desired `tool0` frame configuration
2. If the position in the "x-axis" of the target position is greater than the position in the "x-axis" for the intermediate pose, then the 4th joint angle will be decreased by $\pi/6$
3. If the position in the "x-axis" of the target position is greater than the position in the "x-axis" for the intermediate pose, then the 4th joint angle will be increase by $\pi/6$
4. If the position in the "y-axis" of the target position is greater than the position in the "y-axis" for the intermediate pose, then the 5th joint angle will be increased by $\pi/6$
5. If the position in the "y-axis" of the target position is greater than the position in the "y-axis" for the intermediate pose, then the 5th joint angle will be decrease by $\pi/6$

Figure 2 above depicts the functionality of this algorithm.

"Above Mark" Position

As per the project instructions, we first moved the robot such that it arrived at a position directly above the target mark position. We established this position as 5cm above the target mark. On each target trajectory, the robot will first pause at this position before continuing its motion down to the target mark. The three trajectory control methods discussed below in sections C-E, were implemented twice per mark, first to arrive at the "above mark" position, and the second time to move from the above mark position to the target mark.

B Forward Kinematics

As shown in the inverse kinematics file by Ryan Keating, the UR5 calculates kinematics with respect to the configuration between the `base_link` and `tool0` rather than the `S` and `T` frames we leveraged in Homework 6 [1]. Therefore, we recalculated the forward kinematics equations for this final project.

$$\vec{\omega}_1 = [0,0,1] \qquad \vec{q}_1 = [0,0,0]$$
$$\vec{\omega}_2 = [0,1,0] \qquad \vec{q}_2 = [0,0,L0]$$
$$\vec{\omega}_3 = [0,1,0] \qquad \vec{q}_3 = [L1,0,L0]$$
$$\vec{\omega}_4 = [0,1,0] \qquad \vec{q}_4 = [L1+L2,0,L0]$$
$$\vec{\omega}_5 = [0,0,-1] \qquad \vec{q}_5 = [L1+L2,L3,0]$$
$$\vec{\omega}_6 = [0,1,0] \qquad \vec{q}_6 = [L1+L2,0,L0-L4]$$

$$\vec{\xi}_i = \begin{pmatrix} -\omega_i \times q_i \\ \omega_i \end{pmatrix}$$

$$R_0 = R_x(\pi/2)R_y(-\pi)$$

$$\vec{p}_0 = [L1+L2, L3+L5, L0-L4]^T$$

$$g_{bt}(0) = \begin{pmatrix} R_0 & \vec{p}_0 \\ \vec{0} & 1 \end{pmatrix}$$

$g_{bt}(0)$ is the transformation between the `base_link` and tool0 frames at the ur5.home configuration. g_{bt} is the transformation between the `base_link` and tool0 at any other configuration.

$$g_{bt} = e^{\hat{\xi}_6\theta_6}e^{\hat{\xi}_5\theta_5}e^{\hat{\xi}_4\theta_4}e^{\hat{\xi}_3\theta_3}e^{\hat{\xi}_2\theta_2}e^{\hat{\xi}_1\theta_1}g_{bt}(0)$$

C Inverse Kinematics

As seen in class and in the inverse kinematics file written by Ryan Keating , for a given desired homogeneous transformation $g_d \in SE(3)$, the inverse kinematics finds a set of possible solutions of joint angles using the sub-problems 1,2 and 3 that allow the robot arm to reach the desired position. Our aim is to find the optimal solution amongst this set of possible solutions of joint angles. The optimal solution eliminates the possibility of collision while having the smallest angle movement. The following algorithm provides the step by step process to find this optimal solution.

Algorithm

1. The set of all possible solutions $q = [q_1, q_2, ..., q_n]$ are found using the inverse kinematics.
2. For each solution q_i where $i \in [1, 2, ..., n]$:
   - The body Jacobian matrix, J_b^i is first calculated.
   - The next step is to compute the manipulability measure as a measure of the configuration.
   - Check for singularity such that if $|\mu > 0.0001|$ move on to the next step otherwise eliminate q_i as the possibility of being the optimal solution.
   - Check to see if the end-effector at q_i is above the table i.e. the second joint angle of q_i should be less than 0 and greater than -pi. If the condition is satisfied move on to the next step otherwise eliminate q_i as the optimal solution.
   - Find the homogeneous transformation of q_i which is denoted as g_i using forward kinematics.
   - Find the translation vector p_i from the g_i computed.
   - In this step we check to see if q_i causes collision. If $p_i > 0$ we move on to the next step otherwise q_i is eliminated as the optimal solution.

- q_{best} should have the smallest angle movement. Keeping this in mind we do the following:
  For $i = 1$ we initialize q_{best},

  $$q_{best} = q_i$$

  For $i > 1$, we check to see if the following condition is satisfied (q_{sp} corresponds to the joint angles of the robot arm at start position):

  $$||q_i - q_{sp}||_2 < ||q_{best} - q_{sp}||_2$$

  If the above equation is satisfied then,

  $$q_{best} = q_i$$

  Otherwise we eliminate q_i as the optimal solution and move on to q_{i+1}.

Experimental results

Please view the video at this link to view the live demo using Inverse Kinematics to perform the control trajectory. As you can see in the video, the inverse kinematics is much faster and more accurate that the other two motion control mechanisms.

The images **Figure 3** confirm via the RVIZ simulation environment that the Inverse Kinematics calculations are highly accurate by showing that the `tool0` frame exactly aligns with the `qa,qmark1,qb,qmark2` frames.

The video together with the simulation prove that from the 8 potential solutions given by the sub-problem method of solving the inverse kinematics, we were able to select the joint angles solution that provided efficient motion to the target position.

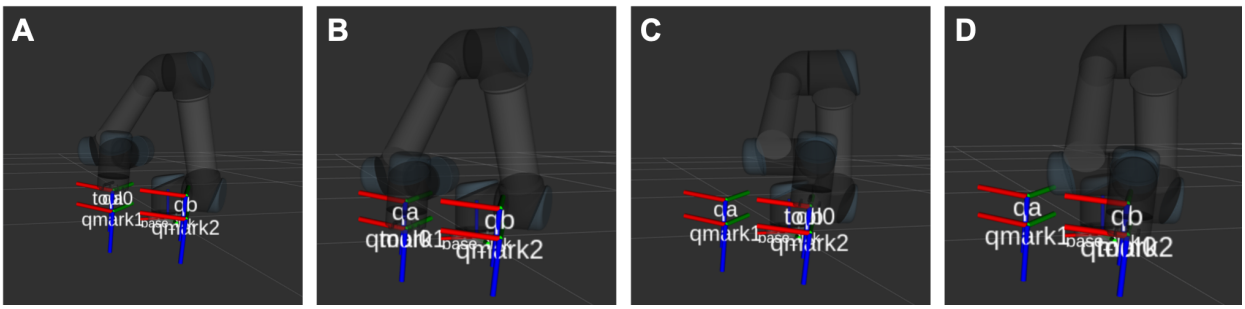

Figure 3: Sequence of images from the RVIZ simulation environment depicting that the Inverse Kinematics algorithm is able to achieve highly accurate motion for reaching the above mark positions (A,C) and mark positions (B,D).

D Rate control using differential kinematics

Algorithm (work flow)

1. The gain of the controller, K is taken as 1 and while the time step, T_{step} is taken to be 0.1. The initial error, e_{t_0} is 0.
2. Get the current joint angles, q_{t_0}.
3. The homogeneous transformation of the current joint angles, $g_{st}^{t_0}$, is calculated using the forward kinematics.
4. ξ_{t_0} is calculated from the following relation:

   $$e^{\hat{\xi}_{t_0}\theta} = (g_{desired})^{-1} g_{st}^{t_0}$$

5. $\vec{v_{t_0}}$ and $\vec{\omega_{t_0}}$ are obtained from the relation:

   $$\xi_{t_0} = [\vec{v}_{t_0}\ \vec{\omega}_{t_0}]^T$$

6. As long as $\vec{v}_t$ or $\vec{\omega}_t$ are greater than the control thresholds $0.0035m$ and $\pi/24rad$ respectively:
   - Find J_b which is the body Jacobian matrix of q_t.
   - Compute the manipulability measure as a measure of the configuration.
   - Check for singularity such that if $|\mu < 0.001|$ then return the error e_{t+1} as -1.
   - Update q_t:
   $$q_{t+1} = q_t - KT_{step}((J_b)^{-1}\xi_t)$$
   - Repeat steps 3-5 for q_{t+1} to get g_{st}^{t+1} and ξ_{t+1}.
   - Find g_{tt} and extract the corresponding translation vector $\vec{p}_{t+1}$:
   $$g_{tt} = (g_{desired})^{-1} g_{st}^{t+1}$$
   - Move the robot arm with the joint angles q_{t+1}.
7. The final error given by,
$$e_{t_f} = ||\vec{p}_{t_f}||_2 * 100cm$$

Experimental results

Please view the video at this link to view the live demo using Resolved Rate Control to perform the control trajectory. As you can see in the video, the Resolved Rate control is slower than the inverse kinematics since as it nears the target location, it motion per iteration becomes more limited. Additionally, when compared to the accuracy of the inverse kinematics, there is a very slight error associated with the Resolved Rate control mechanism.

The images in **Figure 4** confirm via the RVIZ simulation environment that our resolved rate control algorithm is able to move the robot to the target locations by showing that the `tool0` frame closely aligns with the `qa,qmark1,qb,qmark2` frames. However, due to the nature of the control mechanism, there is some minor error when compared to the Inverse Kinematic plots. While our tuning process reduced a significant portion of the error, the error can be further reduced by continuing to tune the parameters to achieve an optimal setting.

The video of the live demo together with the simulation depict the successful implementation of the Resolved Rate control mechanism.

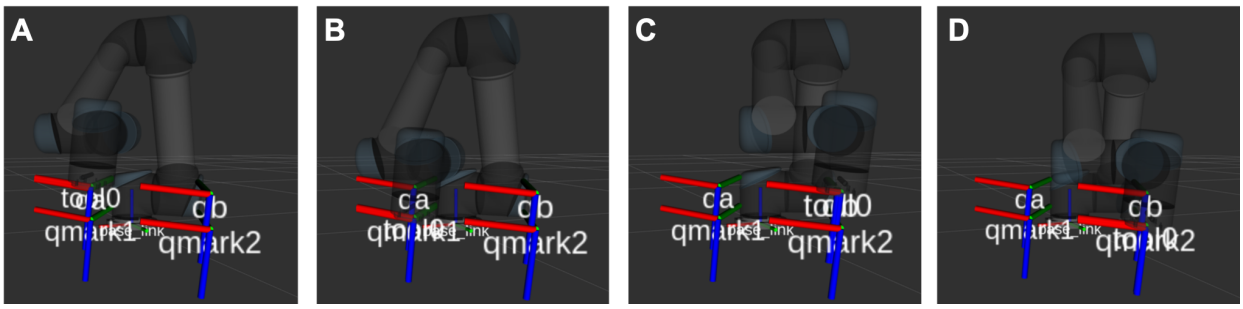

Figure 4: Sequence of images from the RVIZ simulation environment depicting that the Resolved Rate control algorithm is able to arrive at the target locations for the above mark positions (A,C) and mark positions (B,D).

E Gradient-based control

Algorithm

1. The gain of the controller, K is taken as 1 and while the time step, T_{step} is taken to be 0.1. The initial error, e_{t_0} is 0.
2. Get the current joint angles, q_{t_0}.

3. The homogeneous transformation of the current joint angles, $g_{st}^{t_0}$, is calculated using the forward kinematics.
4. ξ_{t_0} is calculated from the following relation:

$$e^{\hat{\xi}_{t_0}\theta} = (g_{desired})^{-1} g_{st}^{t_0}$$

5. $\vec{v_{t_0}}$ and $\vec{\omega_{t_0}}$ are obtained from the relation:

$$\xi_{t_0} = [\vec{v}_{t_0}\ \vec{\omega}_{t_0}]^T$$

6. As long as $\vec{v}_t$ or $\vec{\omega}_t$ are greater than the control thresholds $0.0035m$ and $\pi/24rad$ respectively:
   - Find J_b which is the body Jacobian matrix of q_t.
   - Compute the manipulability measure as a measure of the configuration.
   - Check for singularity such that if $|\mu < 0.001|$ then return the error e_{t+1} as -1.
   - Update q_t:
   $$q_{t+1} = q_t - KT_{step}((J_b)^T \xi_t)$$
   - Repeat steps 3-5 for q_{t+1} to get g_{st}^{t+1} and ξ_{t+1}.
   - Find g_{tt} and extract the corresponding translation vector $\vec{p}_{t+1}$:
   $$g_{tt} = (g_{desired})^{-1} g_{st}^{t+1}$$
   - Update the error:
   $$e_{t+1} = ||\vec{p}_{t+1}||_2 * 100cm$$
   - If $e_t - e_{t+1}$ is greater than the threshold 0.2, move the robot arm with the joint angles q_{t+1}.
   - If $e_t - e_{t+1}$ is less than the threshold 0.2, vary T_{step} by a factor of 10 and then move the robot arm with the joint angles q_{t+1}.

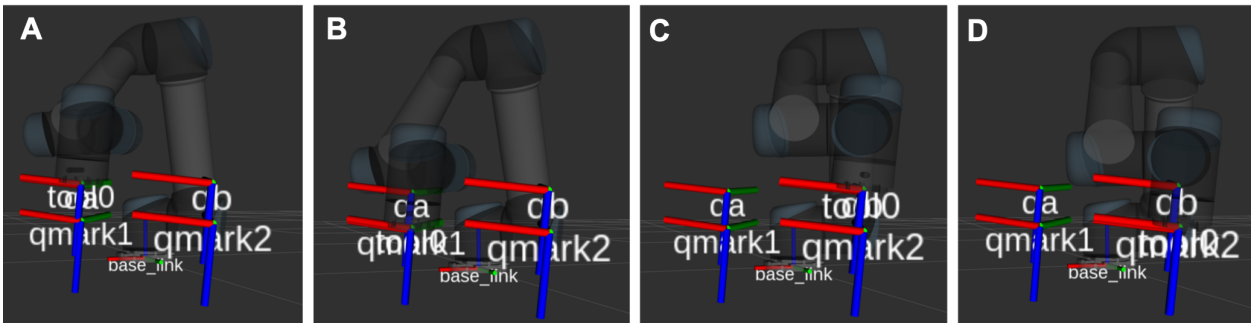

Figure 5: Sequence of images from the RVIZ simulation environment depicting that the Gradient Control algorithm is able to arrive at the target locations for the above mark positions (A,C) and mark positions (B,D).

Experimental results

Please view the video at this link to view the live demo using Gradient Control to perform the control trajectory. As you can see in the video, the Gradient control is slower than the inverse kinematics since as it nears the target location, it motion per iteration becomes more limited. It is also slower than the Resolved Rate control algorithm, however, for the gradient control we implemented a dynamic time-step based on the error in position from the target, which significantly sped up the process. Like the resolved rate control method, when compared to the accuracy of the inverse kinematics, the gradient control method has a slight error.

The images in **Figure 5** confirm via the RVIZ simulation environment that our gradient control algorithm is able to move the robot to the target locations by showing that the `tool0` frame closely aligns with the `qa,qmark1,qb,qmark2` frames. However, due to the nature of the control mechanism, there is some minor error when compared to the Inverse Kinematic plots. This error can be further reduced by continuing to tune the parameters to achieve an optimal setting.

The transnational error for both the gradient and resolved rate control methods was approximately 0.0035mm, the tolerance level set for the iteration loop. Thus, we believe if we were to set this tolerance lower, we would be able to achieve higher accuracy.

II Extra Credit Task

Please view the linked video to view the performance of the Write "RDKDC" at Any Orientation task.

The aim is to make the robot arm write "RDKDC" on a paper that can be placed at any orientation. This is done in two steps which are explained in detail in the Algorithm of this section.

We chose to implement the inverse kinematics control method for this task due to the increased speed and accuracy over the other two control methods. After teaching the robot the desired positions of key mark points in the text once, we were able to successfully have the robot write "RDKDC" with high accuracy. As you can see in **Figure 6**, the hand written drawing is able to be replicated exactly. The minor shift in the location of the text on the page is due to a change in the location of the marker. During the training phase, the marker was positioned in the center of the end-effector. However, during subsequent experimental runs, the marker was taped to the side of the end effector for increased stability.

The main challenge we faced while designing this algorithm was establishing the local coordinate frame of the paper with respect to the robot. Given the frame transformation concepts discussed during class, we knew that once we established the location of the 27 points with reference to a local coordinate frame of the paper, we could then successfully write RDKDC regardless of the orientation of the paper by establishing the relationship between the coordinate frame of the paper and the `base_link` of the robot.

Additionally, in the extra credit task, instead of the robot always first going to the "above mark" position, followed by the mark position, it now had 3 different potential paths:

1. Move to above position, then down to mark position ("above before")
2. Move directly from one mark position to the next
3. Move from mark position, to the above mark position ("above after")

As you can see in **Figure 6A**, the dots circled in yellow correspond to the "above before" points, since the robot arm always approaches these mark points by first moving to their "above mark" position, and then down to the mark. Similarly, the dots circled in pink correspond to the "above after" points, since the robot arm always moves from these mark positions to the corresponding above mark positions, before moving to the next location. The points not highlighted are points where the robot arm is instructed to remain at a constant level in the "z-axis" in order to keep the marker in contact with the paper.

The algorithm below highlights the steps necessary to establish the local coordinate frame of the paper, defined the 27 points with reference to this local coordinate frame, and then instruct the robot to write RDKDC by defining the relationship between the robot's base coordinate frame and the coordinate frame of the paper.

Algorithm

1. Algorithm to Store the data from the original points:
   (a) Create target drawing of "RDKDC" text on a paper at an orientation and define key points necessary to guide the robot (total = 27 points) as seen in Fig. 6A.
   (b) Collect the joint angles of all the 27 points, $q_p = [q_{p_1} q_{p_2}, ..., q_{p_{27}}]$.
   (c) Collect the joint angles of all the four corners of the paper, $q_c = [q_{cTL}, q_{cTR}, q_{cBR}, q_{cBL}]$.
   (d) For each of the corner points, find the transformation matrix, $g^j_{c_{original}}$ where $j \in [TL, TR, BR, BL]$) , using forward kinematics.

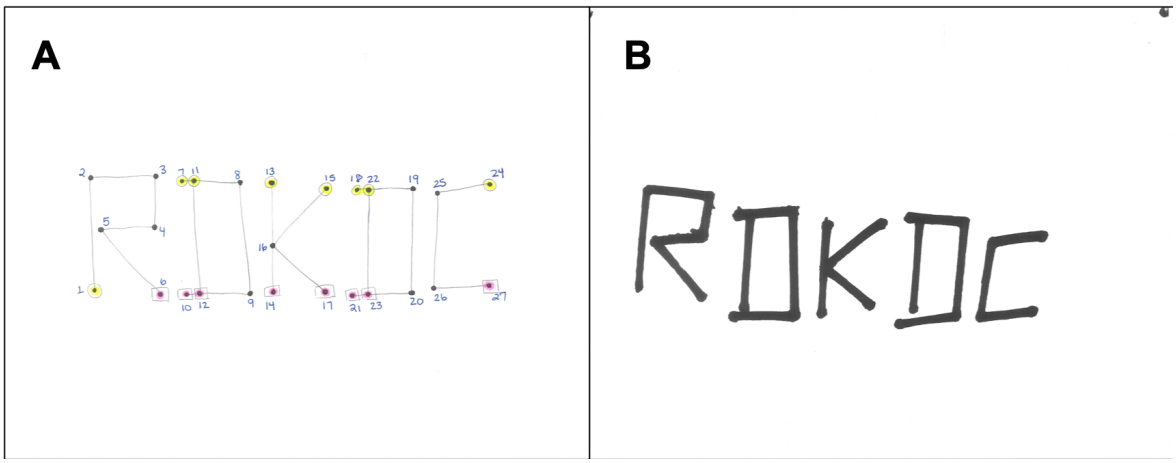

Figure 6: (A) Drawing used to train robot. (B) Final Output Produced. *Please note: the final output is shifted slightly to the left of the target drawing because we shifted where we placed the pen. When training the robot, the pen was located in the center of the end effector. In the final output the pen is located on the side of the end effector.

(e) Find the transformation matrix for each of the 27 points, $g^i_{p_{original}}$ where $i \in [1, 27]$ using forward kinematics.

(f) $g^i_{p_{original}}$ for all the points and $g^j_{c_{original}}$ for all the corner points are stored offline to be used in the next step.

2. Algorithm for the robot arm to write on the paper that has been placed at any orientation:

(a) The stored $g^i_{p_{original}}$ for all the points and $g^j_{c_{original}}$ for all the corner points are loaded.

(b) The translation vector of the original corner points, $p^j_{c_{original}}$, are found.

(c) The orientation of the paper from the original data is calculated as follows:

$$\Delta p_{original} = p^{TL}_{c_{original}} - p^{TR}_{c_{original}}$$

$$\Delta p_{original} = [\Delta p_{xoriginal}, \Delta p_{yoriginal}, \Delta p_{zoriginal}]^T$$

(d) The angle between the robot frame and the paper frame for the original data is found using the following formula:

$$\theta_{original} = tan^{-1}(\frac{\Delta p_{xoriginal}}{\Delta p_{yoriginal}})$$

(e) We extract the translation vector of each point, $p^i_{p_{original}}$ from $g^i_{p_{original}}$.

(f) We now calculate the x and y position of each of the 27 points from the original data relative to the top left corner of the paper in the robot frame:

$$p^i_{pTL_{Roriginal}} = p^i_{p_{original}} - p^{TL}_{c_{original}}$$

Here,

$$p^i_{pTL_{Roriginal}} = [p^i_{xpTL_{Roriginal}}, p^i_{ypTL_{Roriginal}}, p^i_{zpTL_{Roriginal}}]^T$$

(g) We then translate the x and y position of $p^i_{pTL_{Roriginal}}$ from the robot frame to the paper frame:

$$p^i_{xpTL_{paper}} = p^i_{xpTL_{Roriginal}} cos(\theta_{original}) - p^i_{ypTL_{Roriginal}} sin(\theta_{original})$$

$$p^i_{ypTL_{paper}} = p^i_{xpTL_{Roriginal}} sin(\theta_{original}) + p^i_{ypTL_{Roriginal}} cos(\theta_{original})$$

(h) We place the paper at any orientation on the table and get the joint angles of the robot at the top left corner of the paper and top right corner of the paper which are denoted as q_{TL} and q_{TR} respectively.

(i) We then find the transformation matrices g_c^{TL} and g_c^{TR} from q_{TL} and q_{TR} respectively using forward kinematics.

(j) The translation vectors of the new corner points, p_c^{TL} and p_c^{TR}, are extracted from g_c^{TL} and g_c^{TR} respectively.

(k) The orientation of the paper from the new data is calculated as follows:

$$\Delta p = p_c^{TL} - p_c^{TR}$$

$$\Delta p = [\Delta p_x, \Delta p_y, \Delta p_z]^T$$

(l) The angle between the robot frame and the paper frame for the new data is found using the following formula:

$$\theta = tan^{-1}(\frac{\Delta p_x}{\Delta p_y})$$

(m) We find the x and y position of each point relative to the top left of the paper at the new orientation:

$$p_{xpTL}^i = p_{xpTL_{paper}}^i cos(\theta) + p_{ypTL_{paper}}^i sin(\theta)$$

$$p_{ypTL}^i = p_{xpTL_{paper}}^i sin(\theta) + p_{ypTL_{paper}}^i cos(\theta)$$

(n) We now find the x and y position of each point relative to the robot base:

$$p_c^{TL} = [p_{xc}^{TL}, p_{yc}^{TL}, p_{zc}^{TL}]^T$$

$$p_{xpTL_R}^i = p_{xc}^{TL} + p_{xpTL}^i$$

$$p_{ypTL_R}^i = p_{ypTL}^i + p_{ypTL}^i$$

(o) We find the rotation matrix R_c from g_c^{TL}.

(p) The translation vector t_i for each point is now defined as by,

$$t_i = [p_{xpTL_R}^i, p_{ypTL_R}^i, p_{zc}^{TL}]^T$$

(q) We finally find the transformation matrix from the robot base frame to each point:

$$g_p^i = \begin{pmatrix} R_c & t_i \\ \vec{0} & 1 \end{pmatrix}$$

3. Move robot via Inverse Kinematics method for trajectory control

(a) Establish 2 arrays, one for "Above Before" points, and a second for "Above After" points.

(b) Move robot to ur5.home

(c) Via optimal inverse kinematics algorithm discussed above (section C of the Main Task), calculate the "above mark" position for point 1 in the robot frame and move robot to "above mark" position 1

(d) Calculate the mark 1 position in the robot frame via the optimal inverse kinematics algorithm and move the robot to mark 1

(e) Move the robot through the rest of the 27 points by iterating through each of the frames (g_i to $g_{numPoints}$ where $i = 2, ..., 27$) following the steps below

i. If point i exists in "Above Before", first move to the above position ($g_{i_{above}}$) from the prior position (g_{i-1}) and then move down to the mark (g_i)

ii. If point i exists in "Above After", first move to the mark position (g_i) from the prior position (g_{i-1}) and then move up to the above mark position ($g_{i_{above}}$)

iii. If point i is in neither list, then move robot from prior position (g_{i-1}) to the next position (g_i) without going to any above mark position (motion remains in a constant plane)

Experimental results

Please view the live demo link to view the execution of the extra credit task. The video shows 3 sample runs. In all runs, the robot was able to successfully write "RDKDC" with high accuracy and speed. The speed was primarily limited by the pause times required by the communication speed between the computer and the UR5. A sample result is also depicted in **Figure 6B**.

III Workload Distribution

We worked collaboratively across all aspects of this project, from brainstorming initial solution ideas to actual implementation of code. This collaboration ultimately resulted in the success of our extra credit project, as we all brought a variety of strengths to tackling the project we set out to execute.

Figure 7: Group picture following a successful demo.

IV Appendix

A Singularity Conditions

- $\theta 3 = 0, \pi$
- $\theta 5 = 0, \pi$
- $\theta 2, \theta 3, \theta 4 = 0, \pi$ (all 3 angles are either 0 or π at the same time)

B YouTube Videos

- Main Task: Pick and Place with Intention
- Extra Credit: Write "RDKDC" at Any Orientation

References

[1] R. Keating, "UR5 Inverse Kinematics." 2014.