ECE 470: Lab 1 - Robot Programming I: UR3

Xiaoman Li
NetID: 3220111422 xiaoman9
Lab Partner: Jiaxin Cao
TA:
Section: Thursday 6PM

March 7, 2025

1 Introduction

In this lab, we will use a UR3 robot arm to solve the Tower of Hanoi problem by moving a stack of three blocks from one designated location to another while following its fundamental rules. The Tower of Hanoi is a well-known mathematical puzzle introduced by French mathematician Édouard Lucas, consisting of three rods and a set of disks of varying sizes. The objective is to transfer the stack from a starting rod to a target rod with the following constraints:

1. Only one block can be moved at a time.
2. A block can only be placed on an empty rod or on top of a larger block.
3. A block cannot be placed on a smaller block.

In this lab, the UR3 robot will be programmed using its Teach Pendant, employing different motion types—MoveJ (joint motion), MoveL (linear motion), and MoveP (process motion)—to precisely manipulate the blocks. Through this experiment, we explore robotic motion control, understand the limitations of different movement strategies, and develop a structured approach to solving a robotic task.

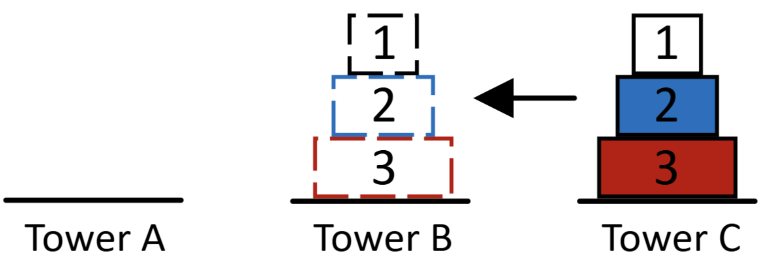

Figure 1: Moving Blocks from Tower C to Tower B
The blocks 1, 2, and 3 are arranged from smallest to largest (white, blue, red) as they move from Tower C to Tower B in this lab.

2 Method

2.1 Setup

At the beginning of the experiment, we have three blocks (1, 2, and 3) stacked at Tower C, and our goal is to move them to Tower B following the rules of the Tower of Hanoi as shown in Figure 1. In this experiment, blocks 1, 2, and 3, from top to bottom, correspond to white, blue, and red, respectively, in increasing size. The movement strategy is derived from the standard algorithm, which states that for n disks, the minimum number of moves required is $2^n - 1$. Since we have three blocks, the total number of moves should be $2^3 - 1 = 7$.

To implement the movement, we number the three positions from left to right as A, B, and C, where:

- C is the initial position (start)
- B is the target position (end)
- A is used as an auxiliary position

The correct sequence of movements includes 7 steps:

$$C \rightarrow B => C \rightarrow A => B \rightarrow A => C \rightarrow B => A \rightarrow C => A \rightarrow B => C \rightarrow B$$

To execute this plan using the UR3 robotic arm, we follow these steps for each movement:

1. Lift the arm to avoid collision with other blocks.
2. Move horizontally towards the designated block.
3. Lower the arm and activate the suction function.
4. Wait for stabilization.
5. Raise the arm to provide clearance.
6. Move horizontally toward the destination.
7. Lower the arm to place the block and deactivate the suction.
8. Repeat the process for the next step.

Initially, we attempted to move blocks directly from one position to another. However, we observed that collisions occurred when moving over existing blocks. To avoid such collisions, we ensured that the robotic arm always lifted the block before moving horizontally. After completing all movements, we halted the machine to ensure safety.

By following this structured approach, we successfully transferred the tower while maintaining stable block stacking and collision-free movements.

2.2 Programming the UR3

To implement the Tower of Hanoi task, we divided the entire process into seven movement units, corresponding to the seven moves required to transfer the three-block tower. Each movement unit consists of a complete pick-and-place cycle, including:

1. Moving to the block's initial position.
2. Activating the suction cup to pick up the block.
3. Moving to the target position.
4. Deactivating the suction cup to place the block.

Each movement unit follows a command structure similar to Table 1, where key robot instructions are executed in sequence.

Command	**Description**
Set DO[0] = Close	Initialize the Gripper
point1	Initialize Position
point2	Lower the Block
wait: 0.8s	Wait to Stabilize
Set DO[0] = Open	Suck the Block
point1	Lift the Block
point3	Horizontal Shift
point4	Lower the Block
wait: 0.8s	Wait to Stabilize
Set DO[0] = Close	Drop the Block

Table 1: UR3 Operation Sequence for One Move

2.2.1 Command Explanation

Each instruction in Table 1 plays a crucial role in ensuring stable and collision-free block movement.

- Set DO[0] = Close / Open
  The digital output (DO[0]) is used to control the suction gripper. Setting DO[0] = Close means Set DO[0] to low, which turns off the suction, allowing the gripper to releasing the block. Conversely, setting DO[0] = Open turns on the suction, pick up the block.
- Waypoints (point1, point2, point3, point4)
  Each point represents a waypoint in the movement unit, defining positions for lifting, shifting, and placing blocks. These waypoints ensure precise path control during the motion.

2.2.2 Motion Type Selection

To evaluate the performance of different movement settings, we tested MoveJ, MoveL, and MoveP across different movement units:

- For movement units 1-3 → MoveJ (Joint Motion)
  - Used for initial fast repositioning.
  - Allows faster movement but path is not linear.
  - Suitable for non-precise repositioning where direct path control is not needed.
- For movement units 4-5 → MoveL (Linear Motion)
  - Ensures the robot moves in a straight-line path between waypoints.
  - Provides better path control, avoiding unexpected deviations.
  - Necessary for precise block stacking and controlled placement.
- For movement units 6-7 → MoveP (Process Motion)
  - Ensures smooth and continuous motion between multiple waypoints.
  - Maintains constant velocity, reducing jerky movements.
  - Best suited for ensuring smooth and stable block placement.

By comparing MoveJ, MoveL, and MoveP, we aim to analyze their differences in path accuracy, execution time, and stability. The results and observations of this comparison will be discussed in later sections of this report.

2.3 Circular Motion Implementation

To implement circular movement, MoveP was used to ensure a smooth transition between pick and place positions. MoveP was particularly useful for reducing jerky motion and maintaining a consistent velocity when moving the blocks. The process involved:

- Using waypoints in movement units 6-7 to form a circular path.
- Testing different blend radius to achieve the most stable arc motion.

Observations showed that MoveP provided a more fluid motion compared to MoveJ and MoveL, which caused some abrupt changes in velocity.

3 Data and Results

3.1 Complete Movement Command Code

To analyze the performance of the UR3 robotic arm in solving the Tower of Hanoi problem, we recorded all executed movements, their corresponding execution times, and any observed deviations. Each movement follows a structured command sequence, ensuring the precise control of the robotic arm while minimizing the risk of collision or misalignment.

The full sequence of movement commands executed by the UR3 robotic arm is shown in Listing 1. This listing provides a detailed breakdown of each movement unit.

Listing 1: UR3 Tower of Hanoi Movement Commands

```
MoveJ:
    // Movement 1: Move block from C to B
    Set DO[0] = Close  // Activate gripper
    Up_C  // Lift block from C
    Down_C_1  // Move down to grasp position
    Wait: 0.8s  // Wait for stabilization
    Set DO[0] = Open  // Release block
    Up_C
    Up_B
    Down_B_1
    Wait: 0.8s
    Set DO[0] = Close

    // Movement 2: Move block from C to A
    Up_B
    Up_C
    Down_C_2
    Wait: 0.8s
    Set DO[0] = Open
    Up_C
    Up_A
    Down_A_1
    Wait: 0.8s
    Set DO[0] = Close

MoveL:
    // Movement 3: Move block from B to A
    Up_A
    Up_B
    Down_B_1
    Wait: 0.8s
    Set DO[0] = Open
    Up_B
    Up_A
    Down_A_2
    Wait: 0.8s
    Set DO[0] = Close

    // Movement 4: Move block from C to B
    Up_A
    Up_C
    Down_C_3
    Wait: 0.8s
    Set DO[0] = Open
    Up_C
    Up_B
    Down_B_1
    Wait: 0.8s
    Set DO[0] = Close

MoveP:
    // Movement 5: Move block from A to C
    Up_B
    Up_A
    Down_A_2
    Wait: 0.8s
    Set DO[0] = Open
```

```
    Up_A
    Up_C
    Down_C_3
    Wait: 0.8s
    Set DO[0] = Close

    // Movement 6: Move block from A to B
    Up_C
    Up_A
    Down_A_3
    Wait: 0.8s
    Set DO[0] = Open
    Up_A
    Up_B
    Down_B_2
    Wait: 0.8s
    Set DO[0] = Close

    // Movement 7: Move block from C to B
    Up_B
    Up_C
    Down_C_3
    Wait: 0.8s
    Set DO[0] = Open
    Up_C
    Up_B
    Down_B_3
    Wait: 0.8s
    Set DO[0] = Close
```

4 Conclusion

Through this experiment of Tower of Hanoi problem, we explored different movement strategies and their impact on precision, efficiency, and stability.

4.1 Keep Block Stacks Neat

Maintaining neat block stacks requires careful waypoint alignment and smooth path transitions. To be specific, we lifted the arm to avoid collision with other blocks. Also, we gave every waypoint a proper name to clearly show its position, which contributed much to the neatness.

4.2 Observations about MoveJ, MoveL, and MoveP

Each motion type exhibited distinct characteristics as shown in Table 2:

Move Type	Expected Path	Actual Path
MoveJ	Non-linear	Unpredictable but fast
MoveL	Straight line	Slight deviation
MoveP	Smooth curve	Continuous transition

Table 2: Comparison of MoveJ, MoveL, and MoveP

- MoveJ proved to be highly efficient for rapid repositioning but introduced jerky movements, making it less suitable for precise placements.
- MoveL ensured accurate linear motion, but minor deviations occurred due to joint constraints, necessitating careful tuning.
- MoveP provided the smoothest and most continuous motion, making it the optimal choice for executing circular and fluid transitions.

Overall, programming and executing motions using the Teach Pendant enhanced our understanding of robotic kinematics and motion control. By analyzing the trade-offs among different movement types, we gained insights into optimizing robot trajectories for various applications.