ECE 470: Lab 0/Lab 1
Introduction to Robot Arm UR3:
Three Block Towers of Hanoi

Shenghua Ye, Xuan Tang, Jie Wang
Student ID: 3200112420, 3200115277 3200112404
Instructor: Yang Liangjing

February 4, 2024

1 Introduction

The Tower of Hanoi is a classic mathematical given by the French mathematician Édouard Lucas in 1883. This puzzle involves three pegs and a set of disks of different sizes which are initially stacked in ascending order on one peg, with the smallest disk at the top, forming a conical shape.. The objective of the puzzle is to move the entire stack to another rod, obeying the following simple rules:

1. Only one disk can be moved at a time.
2. Each move consists of taking the upper disk from one of the stacks and placing it on top of another stack or on an empty rod.
3. No disk may be placed on top of a smaller disk.

But the goal of our first lab is fairly simple. Using UR3 Robot Arm, we apply the principles of the Tower of Hanoi puzzle to move a "tower" of three blocks from one location on a table to another. With practice on graphic, low level programming method, we understand basic robotic movements and robot kinematics,.

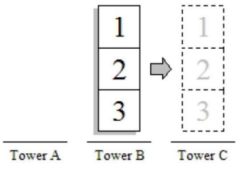

Figure 1.1: Example start and finish tower locations.

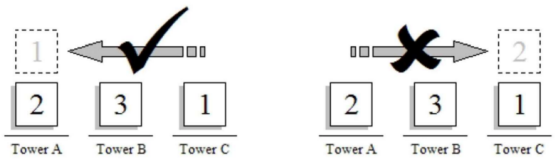

Figure 1.2: Examples of a legal and an illegal move.

Figure 1: Objective of Hanoi Tower

2 Method

2.1 Setting up the Environment

The laboratory provides experiment setup included three designated locations for the placement of rods, along with three circular blocks of different sizes—small, medium, and large—colored white, blue, and red, respectively. These blocks were to be moved across the designated spots on the table, simulating the Tower of Hanoi puzzle.

Firstly, we identified three positions on the robot's table to act as the base `waypoints`, labeled as **S0** (start), **S1** (intermediate), and **S2** (end), with **S0** being the leftmost spot and **S2** the rightmost. This arrangement was chosen to facilitate a clear path for the robot arm to move the blocks from the starting position to the ending position

To optimize the robot arm's movement and ensure efficient handling of the blocks, we chose three overhead positions, approximately 5cm above each base waypoint, named **Home0**, **Home1**, and **Home2**. They are reference points for the robot arm to return to before moving to the next base point. The vertical alignment of these overhead points with their corresponding base points aids in the robot arm's precise vertical movements, enabling accurate pickup and placement of the blocks.

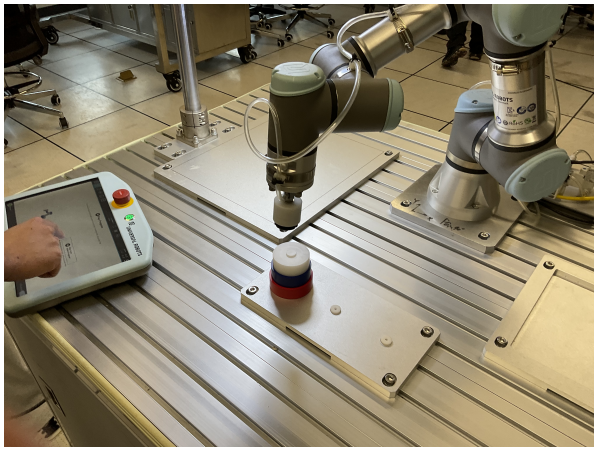

Figure 2: Setup of the Robot Arm with marked positions for the Tower of Hanoi.

This configuration not only ensures the robot arm's movements are optimized for the task but also minimizes the risk of error during the block transfer process. The strategic placement of these points was important as the block may fall over easily while moving between `waypoints`.

2.2 Programming the Robot

To solve the Tower of Hanoi problem, we created a program using the Teach Pendant of the UR3 robot arm. We experimented with `MoveL` (linear move) and `MoveP` (process move) commands. `MoveL` commands move the Tool Center Point (TCP) along a straight line, ideal for transferring blocks from one point to another directly. `MoveP` commands allow for continuous motion at a constant speed, which can be used to create circular arcs, adding efficiency and smoothness to the robot's movements.

The solution is demonstrate in the pseudo code below:

2.3 Implementation and Bug Report

The implementation of circular movement required the use of the `MoveP` command, chosen for its ability to execute movements with swift yet careful consideration of the robot's kinematics. This command was pivotal in ensuring that the TCP executed a smooth, curved trajectory.

During the course of our implementation, we encountered several issues:

Algorithm 1 Solving the Tower of Hanoi Problem

// Initial setup: 3 plates on nail 1
$MovePlate(1, 3)$ {Move smallest plate to nail 3}
$MovePlate(2, 2)$ {Move middle plate to nail 2}
$MovePlate(1, 2)$ {Move smallest plate to nail 2, atop middle plate}
$MovePlate(3, 3)$ {Move largest plate to nail 3}
$MovePlate(1, 1)$ {Move smallest plate back to nail 1}
$MovePlate(2, 3)$ {Move middle plate to nail 3, atop largest plate}
$MovePlate(1, 3)$ {Move smallest plate to nail 3, atop middle plate} =0

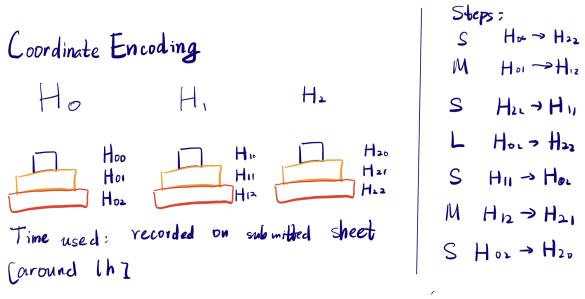

Figure 3: Coordinate encoding strategy for block transfer positions.

1. The suction cup attached to the TCP was partially damaged. The small cracks make it hard to grasp the largest block (L).

   Solution: We approached this by precisely placing the TCP close to the block. After **activating** the suction and **wait** for 2 seconds, we ensured that the block was vertically secured before proceeding to move it.

2. Initially, we did not adequately modify the naming of `waypoints`. The default names, such as **'waypoint_1', 'waypoint_2**', were prone to cause confusion. For instance, while configuring the coordinates for a subsequent waypoint, we accidentally overwrote a previous waypoint, leading to several debugging iterations.

   Solution: As illustrated in Figure 3, we adopted a coordinate encoding strategy for the block transfer positions. Rational naming conventions expedited the programming process and enhanced the safety of our outcomes.

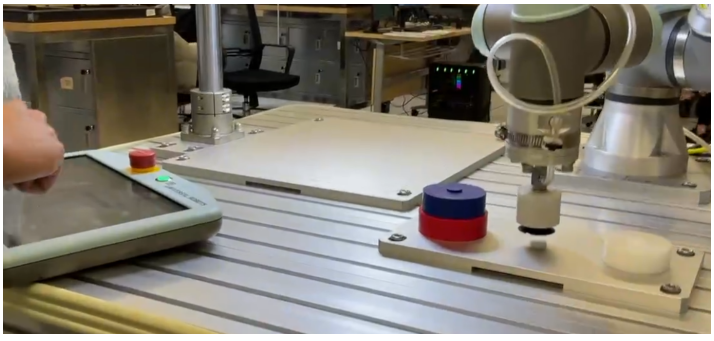

Figure 4: Detailed view of the TCP with a damaged suction cup.

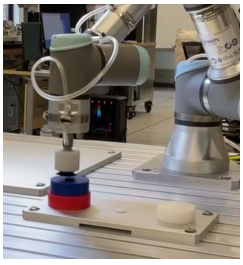

Figure 5: Waypoint naming issue encountered during programming.

3 Conclusion

Throughout this lab, we gained valuable hands-on experience with the UR3 robot arm, learning not just about basic movement commands but also about advanced programming techniques for robotic manipulation. The experiment highlighted the differences between `MoveJ`, `MoveL`, and `MoveP` movements, particularly showcasing how `MoveL` and `MoveP` can be used to achieve more fluid and precise control over the robot's actions. Overall, this lab has prepared us for more complex challenges ahead.

Movement Type	**Use Case**	**Benefits**
MoveJ	Basic joint movements	Quick, less precise
MoveL	Direct linear paths	Precise, smooth transitions
MoveP	Continuous arcs	Smooth, efficient movements

Table 1: Comparison of `MoveJ`, `MoveL`, and `MoveP` movements, highlighting their use cases and benefits in robotic programming.

References

[1] Universal Robots, "Circle using MoveC," in *UR How-Tos*. [Online]. Available: https://www.universal-robots.com/how-tos-and-faqs/how-to/ur-howtos/circle-using-movec-16270/.

[2] Universal Robots, "Circular path using MoveP/MoveC," in *UR How-Tos*. [Online]. Available: https://www.universal-robots.com/how-tos-and-faqs/how-to/ur-howtos/circular-path-using-movepmovec-15668/.

[3] Universal Robots, "Circle with variable radius," in *UR How-Tos*. [Online]. Available: https://www.universal-robots.com/how-tos-and-faqs/how-to/ur-howtos/circle-with-variable-radius-15367/.