Group7 Lab4
Path Planning with the Mycobot Pro 600: Curved Paths

Andy Tsai
ctsai67@asu.edu
1233653410

April 11, 2025

Abstract

Introduction

This lab builds on the work from Lab 3, where I developed a full image-based path planning pipeline for the MyCobot Pro 600 using a straight-line path. In Lab 4, the goal was to extend that approach to follow more complex, curved paths. The main challenge here was to extract smooth, continuous paths from an image and have the robot follow them accurately in both simulation and the real world.

Instead of rebuilding everything from scratch, I reused the digital twin and ROS 2 infrastructure I set up in the previous lab. I continued using OpenCV for image processing to detect the curved path, and the same coordinate transformation method to map the image points into real-world space. Once I had the path points, I used inverse kinematics to solve for joint angles, then published those angles to both the RViz2 simulation and the real robot via TCP commands. The robot's end effector was required to always point downward, and I made sure this constraint was satisfied throughout the motion.

Method

The overall workflow for planning and executing curved path motion with the MyCobot Pro 600 is the same as Lab 3, with modifications to support curved path interpolation:

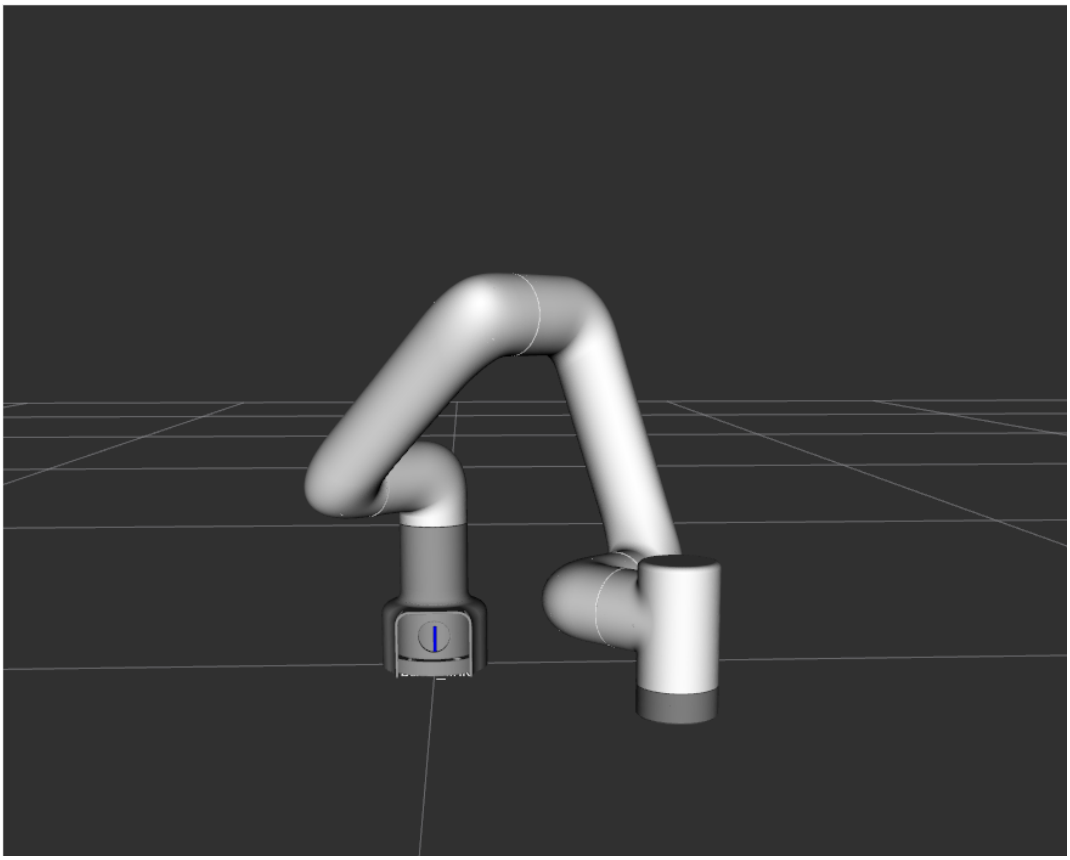
1. **URDF Setup**
   - Used the same URDF model created in Lab 3 from the CAD file in SolidWorks.
   - Loaded the URDF into my ROS 2 workspace and visualized the robot in RViz2.
2. **Image Cropping and Path Extraction**
   - Captured a top-down image of the curved path using a fixed camera.
   - Cropped the image to the usable workspace to remove unnecessary background.
   - Applied OpenCV-based image processing to detect the path and skeletonize it into a single-pixel-wide trace.
   - Sampled 10 evenly spaced points along the skeletonized curve for motion planning.
3. **Pixel-to-World Transformation**
   - Used four known reference points to calculate a transformation matrix from image coordinates to real-world coordinates.
   - Transformed the 10 sampled path points into physical (x, y, z) coordinates.
4. **Inverse Kinematics (IK)**
   - Wrote a custom IK solver to calculate joint angles for each point.
   - Applied the constraint $\theta_2 + \theta_3 + \theta_4 = -\frac{\pi}{2}$ to keep the end effector pointing downward.
5. **Forward Kinematics (FK) Check**
   - Used FK to verify the end-effector position at each IK solution before sending commands.

6. **ROS 2 and Real-World Execution**

   - Published joint angles to the `/joint_states` topic for visualization in RViz2.
   - Sent the same joint angles via TCP to the real robot over Ethernet.
   - Used a ROS 2 timer and appropriate wait intervals to ensure smooth and accurate movement along the curved path.

Results

1. **The Cobot Model (Simulated in RViz2)**



2. **Curve Path Extraction**

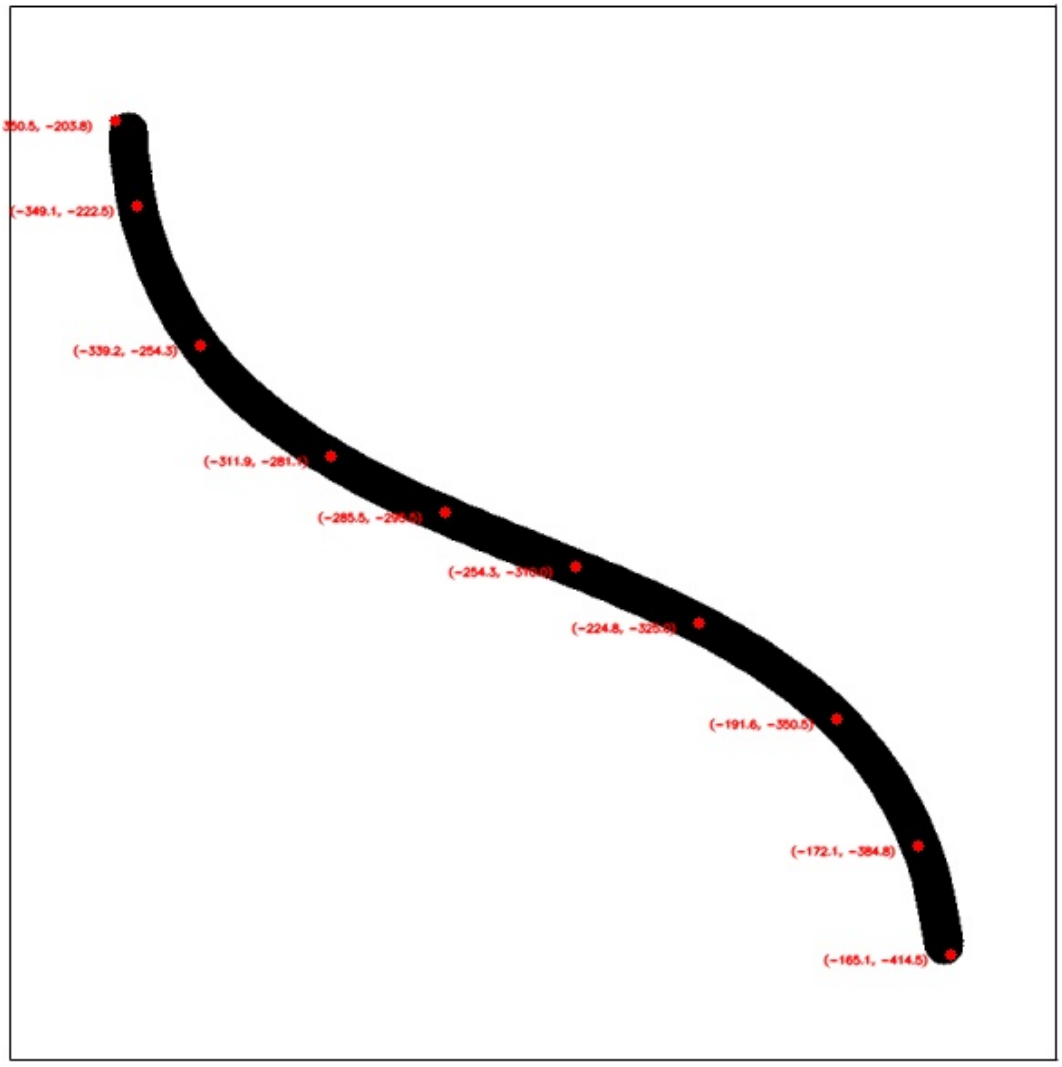


Figure 1: The extracted curve path alone with their coordinates correspondent to the real world coordinates.

3. **Inverse Kinematics & Forward Kinematics Validation**

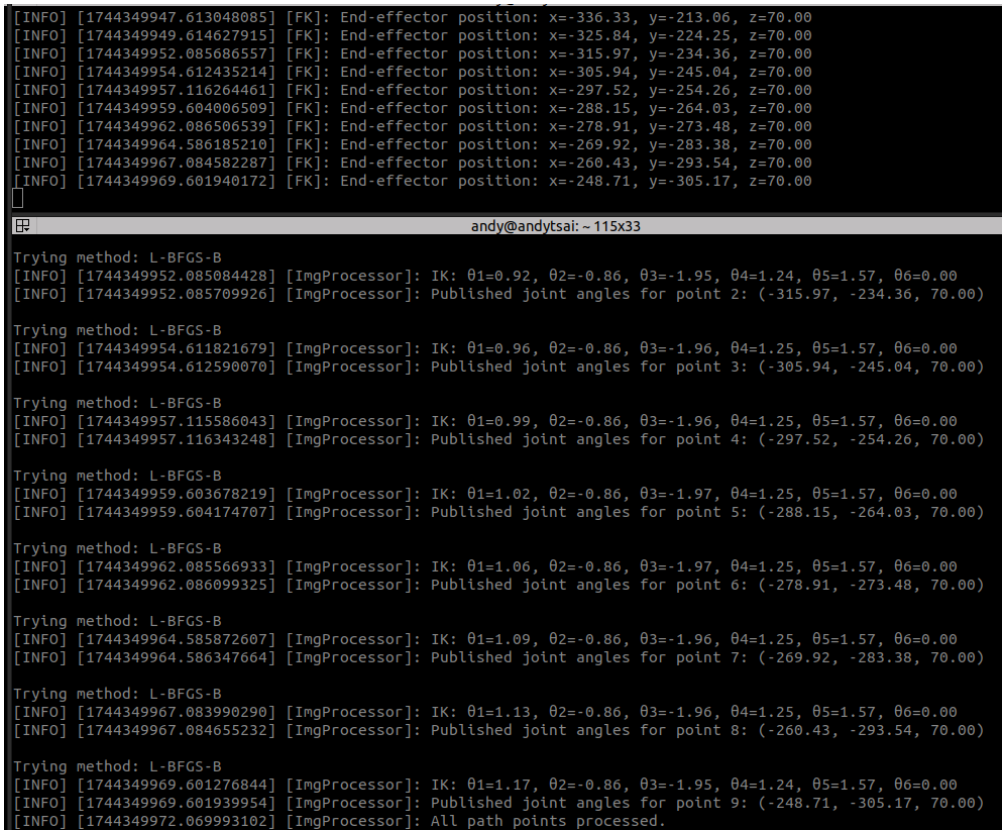


Figure 2: The bottom window is the output messages of applying joints angles from inverse kinematics. The top window is the validation by forward kinematics.

4. **ROS 2 Nodes Graph**

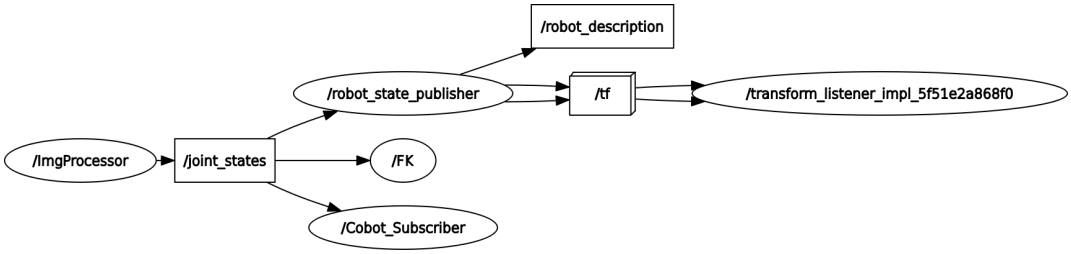


Figure 3: This is the same ROS 2 node graph as Lab3. The `/ImgProcessor` node publishes joint angles to `/joint_states`, which are used by `/robot_state_publisher` for visualization, `/FK` for verification, and `/Cobot_Subscriber` to send commands to the real robot. Transforms are broadcast via `/tf` from the `/robot_description` model.

Conclusion

In this lab, I extended the vision-based path planning pipeline from Lab 3 to handle curved paths. By reusing the same URDF, image processing, and ROS 2 setup, I was able to build a full system that detects a curved path from an image, converts it into real-world coordinates, and calculates the joint angles needed for the robot to follow it. The robot's end effector successfully followed the curved path in both simulation and the real world while maintaining a downward orientation throughout the motion. This lab gave me a better understanding of how interpolation resolution and timing affect path smoothness, and how important it is to tune those for accurate physical execution.

References

[1] Course Materials

[2] ROS2 Tutorials - ROS2 Humble For Beginners

[3] MyCobot Pro 600-Official Website

[4] MoveIt2 and ROS2 Control: How to Configure & Fix Motion Planning Issues with Setup Assistant

[5] OpenCV Course - Full Tutorial with Python