Lab 6 Report

Dohun Jeong, Marsh Ma

dohunj2, yuxuanm4

TA: Chuyuan Tao

Section: Monday 9AM

Submitted: May 10, 2023

Introduction

In this lab, we integrate previous works in ROS, forward and inverse kinematics, and vision-based block detection to accomplish a more complex pick-and-place task. Specifically, we use the same marker detection algorithm from lab 5 to detect the centroid and orientation of multi-colored sticks (comprised of three colored blocks), and use inverse kinematics from lab 4 to stack blocks one by one into two leaning towers that meet at the top level to create a bridge-like structure.

Methodology

Stick Angle Detection

For each center block (colored purple), find the closest yellow block (which is attached to the left of the purple block) and the closest green block (attached to the right of the purple block). To prevent missed marker detection from failing the orientation calculation, we have a three step process:

1. For every purple marker, check the distance between it and the closest yellow marker to make sure that the blocks are separated by a reasonable distance, which was between 20 mm and 28 mm (24mm nominal) in our case. Then calculate the angle using arcsin using the following formula:

   $$yawAngle = arcsin((PurpleY - YellowY)/(PurpleX - YellowX)$$

2. If no yellow marker was found within a reasonable range, look for green marker and use arcsin to calculate the angle.

   $$yawAngle = arcsin((GreenY - PurpleY)/(GreenX - PurpleX)$$

3. If neither markers were within a reasonable range, take the closest yellow and green marker from the purple marker, check if they're within a reasonable range (44 mm to 52 mm) to determine the distance.

   $$yawAngle = arcsin((GreenY - YellowY)/(GreenX - YellowX)$$

4. Otherwise assume 0 degree yaw angle.

Stick Rotation

Once the yaw angle of the block is determined, we need to rotate the block to the correct orientation, which in our case was 90 degrees since we were building the bridge along the world Y-axis. Based on our inverse kinematics solution from lab 4, we control the desired yaw angle based on Joint 6 of the robot. Joint 6 has a angular displacement range of -170 to 170 degrees, meaning a naive implementation will reach safe joint angle limits and cause an error.

We first determine the minimum angular displacement required to turn the block into correct orientation using the following equation:

$$minimumAngle = (target\_yawW - start\_yawW + 180)\ \%\ 360 - 180$$

This formula returns the angular range to be between -180 and 180 degrees. This avoids the scenario such as the robot trying to turn the block by 350 degrees when it can simply turn 10 degrees in the other direction. Once we determine this angle, we divide this minimumAngle by 2 such that the block is picked up at -minimumAngle/2 and is placed at +minimumAngle/2 to stay within -170 and 170 degrees. We also observed that to increase the workspace for our given tower location, we need to favor positive angles, so we add a 45 degree offset as well so that the robot picks up the block at -minimumAngle/2+45 and places it at +minimumAngle/2+45.

Rotate and Move Block Function

Given the stick rotation algorithm described above, we create a rotate and move block subroutine that picks up the block at an angle, and simultaneously rotate and move the block to the target position. To avoid colliding with the tower, we set waypoints 5cm above the target location and approach the goal location top down. However, for the penultimate block, which needs to touch the block from the other location, we found that inaccuracies in block pick up causes the blocks to touch each other on its way down, causing the entire tower to collapse. We therefore modify the motion such that the penultimate block slides in diagonally rather than from the top.

Stick Offset Calculation

- Theoretical stick offset calculations between each layer and the compromises made to make the bridge stand in the real world.

Theoretically, we can use the Tower of Lire formula to calculate how far the blocks can extend from the previous block given the number of levels we want to achieve. This is given in the table below:

Levels	Offset expressed as fraction	Offset expressed as block length (Assume 3-block stick with 25mm cube blocks)
Top Level	0	0mm
Level 5	+- 1/2	37.5mm
Level 4	+- 3/4	56.25mm
Level 3	+- 11/12	68.75mm
Level 2	+- 25/24	78.125mm
Bottom Level	+- 137/120	85.625mm

We initially generated block stack target positions based on the theoretical calculations, but quickly realized that there isn't enough margin for error with the given offset. In the end, we settled on the following offset that was hardcoded in our script:

Levels	Offset expressed as fraction	Offset expressed as block length (Assume 3-block stick with 25mm cube blocks)
Top Level	0	0mm
Level 5	+- 1/2	38 mm
Level 4	+- 3/4	54 mm
Level 3	+- 11/12	68 mm
Level 2	+- 25/24	78 mm
Bottom Level	+- 137/120	83 mm

The top level offset was ever so slightly more than the theoretical offset to prevent block positioning error from knocking down the whole tower. The rest of the offsets were kept to be less than the theoretical offset to allow for a more stable and robust construction.

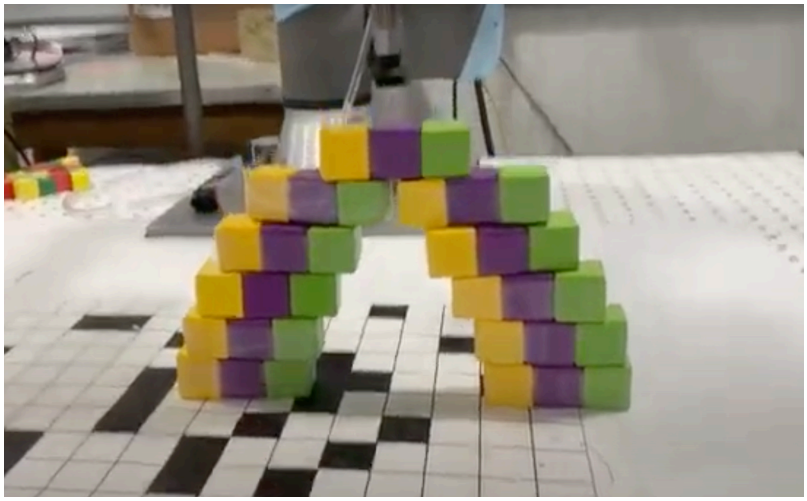
Therefore, given a home position of (0.27m, 0.26m) in world coordinate, we built the target position array top down, starting with height 6*25mm = 150mm in world z-coordinate, and building the tower downwards with the offsets stated in the table above

$$targetX = homeX$$
$$targetY = homeY + yOffset, homeY - yOffset$$

This array of target positions were reversed so that we can stack bottom up.

Data and Results

Our 6-level bridge is two towers of 5-level towers touching each other with a single block connecting the two towers from above. Therefore, the theoretical length of the bridge is 85.625*2 + 75 = 246.25mm. However, with our compromise offsets, the tower should have been 83*2 +75 = 241mm. In the end, we managed to stack a six-level tower that was 232mm long.



To improve the results, we first need a better camera calibration algorithm. Our current camera calibration assumes a fixed z-height and that the image coordinate and the world coordinate can be modeled as a homogeneous transformation. This means that it assumes the image plane and the world coordinate XY plane are coplanar. However, in reality, the camera is not pointing perfectly down at the table. Poor calibration leads to poor inverse kinematic solution that doesn't pick up the centroid of the block even with perfect marker detection.

Even if the camera was pointing perfectly down at the table, the radial distortion of the camera makes blocks at the fringes of the image to have larger error. Also, because the image isn't an orthogona projection, blocks farther from the camera center have not only their top their side surfaces visible in the image, throwing off the centroid detection.

We also need a more robust angle detection algorithm that can track the sticks. A more robust algorithm such as RANSAC (that finds the most likely slope given three block centroids) can reject out an outlier that can exist when blocks are tightly packed together. One alternative solution is to move the block to a special "observation zone" far away from the rest of the blocks, put it down, observe the orientation, and then place it on the tower. We actually successfully implemented this algorithm, but opted not to use it, since it took way longer and required a part of our limited workspace to be empty.

The biggest hardware limitation was the limited workspace it could operate in. We often hit inverse kinematics solution error because when we laid out our eleven blocks, the robot sometimes couldn't reach the farthest block to pick it up.

Conclusion

In this lab, we used the block moving subroutine, forward kinematics, inverse kinematics, and marker detection code from lab 2 to 5 to create a bridge based on two leaning towers. Using the stick angle detection and rotation features that we described in this report, we successfully implemented an algorithm that can stack a 6 level bridge that was 232mm in length.