Final Project: Pick and Place Challenge Report

Reem Azar, Karin Dyer, Konstantinos Gkatzonis, Saibernard Yogendran

Scope

The scope of this project is to combine the concepts and methodology learned in this course to enable the 7 DOF PANDA arm (Figure 1) to pick up static and dynamic (moving) blocks and place them in a desired goal position in a stack using a Geometric Based Inverse Kinematics (IK) solver. The code was tested first in simulation to confirm its correct operation and then in hardware to examine how additional sources of noise may impact its performance.

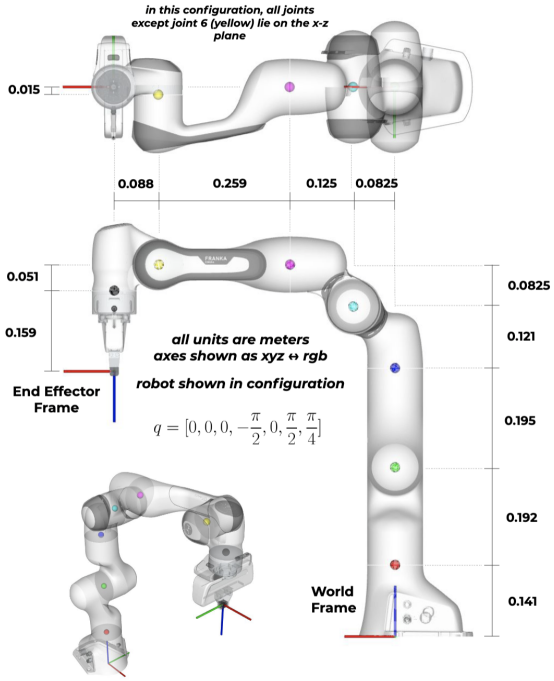


Figure 1: Schematic of PANDA arm

Method

Our strategy considered separate approaches for retrieving static and dynamic blocks and placing them in the goal location. The geometric IK solver we used, constraints the 5th joint of the robot making the robot's DOF to 6. To avoid collisions with the environment, we used a custom planner for static blocks and an advanced version of it for the dynamic blocks. As the environment was known from the beginning, the custom planner we designed for the static blocks ensured that the second joint angle did not go beyond $+\frac{\pi}{2}$, an angle which would bend the arm in such a configuration that it could collide with the obstacles and ground.

Our approach for picking up dynamic blocks utilized an advanced custom planner to have the robot detect a dynamic block and use inverse kinematics to confirm the block is reachable in the desired end-effector orientation. The robot will then approach in a way that guarantees its retrieval while pushing other blocks off the table without colliding with the environment. This dynamic planner also had a restriction for the second joint angle to stay below + $\frac{\pi}{2}$ to avoid hitting the table.

Static Approach

The end effector was set to a predetermined position (q = [-0.07421933, -0.10978325, 0.29795374, -1.78864852, 0.05834795, 1.66149177, 0.97243972]) to allow the camera mounted on the end effector to detect all four static blocks in the camera's view. The camera detects the blocks by calling command *detector.get_detections()* from the detector class which provides the name and pose of all the blocks with respect to the camera frame. This value is stored in a variable named *target_pose*. The command *detector.get_H_ee_camera()* obtains the transformation matrix from the camera to the end effector.

Tc_e (camera with respect to end effector) = $\begin{bmatrix} 0 & -1 & 0 & 0.05 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & -0.06 \\ 0 & 0 & 0 & 0 \end{bmatrix}$

The pose from the camera frame is then converted to the end effector frame and stored in a variable named *target_pose_e* by multiplying *H_ee_camera* × *target_pose.*

To find the pose of the blocks with respect to the world frame, the FK class is called, and the transformation matrix *T0e* (the output of the FK class) is multiplied with *target_pose_e*. This pose is stored in a variable named *target_pose_w* as a dictionary. The rotation matrix and the translation matrix are then extracted and provided as inputs to the IK function. The IK geometric solver provides multiple solutions for reaching the static block. The solver filters the solutions to provide the least path to approach the block and ensure the second joint does not go beyond + $\frac{\pi}{2}$.

When this solution is passed as an input to the function *arm.safe_move_to_position()*, we observed that the robot was hitting and pushing the target static block while reaching the target position. To ensure the end effector did not hit other static blocks, we programmed the robot to adopt a sequential approach. First, the robot will open the gripper using the command *arm.exec_gripper_cmd(10, 50)*, with the first argument being position (10 = fully open, 0 = fully closed) and the second argument being force (50N). The robot will actuate the calculated joint angles for joints 1, 3, and 5, and then move the calculated angles for joints 2, 4, 6, and 7.

To adjust the height where the end effector grabs the block, we created a hyper parameter for tuning the IK input translation matrix, *ik_input['t'][2] = ik_input['t'][2] - 0.012* where 't' is the translation matrix of the *target_pose_w*. This was tuned in hardware to ensure that the end effector grabs the block at 0.012m below the center position for guaranteed retrieval. This sequence is executed after reaching the target static block. This sequential approach ensures that the robot approaches and grabs the target static block efficiently and effectively. The end effector is then programmed to give a tap to the block by closing

the gripper to 0.052 m from the open position. This step ensures the block is oriented perfectly at the center of the end effector, and with each tip of the gripper positioned at the center of the block's opposing flat sides. The robot then grabs the block using the command *arm.exec_gripper_cmd(0, 50)* and moves to the *above_goal_ position* (q = [0.053, 0.119, -0.3349, -1.383, 0.067, 1.528, 0.5026]). From there, the robot descends to the goal location and places the block.

There are certain cases where we observed the robot dropping the block on its way to the goal location. When grabbing the next block, the updated goal location affected the stack quality (as the first block was not there). To ensure the robot knows if it drops a block, a sanity check is performed when the robot reaches the *above_goal_position*. If the gripper state position detected is 0, the robot goes back to the starting position, detects the remaining static blocks and proceeds again, else it goes to the goal position. This condition ensures the robot stacks the remaining blocks perfectly even if a block is dropped. We also observed cases where the robot tries to grab dropped blocks that landed below the goal platform (blue platform / red platform) since the block is in the camera's view. Approaching a dropped block resulted in collision with the workspace. To avoid this, a simple condition was given for the robot to detect and approach blocks only when the z height with respect to the camera is below 0.39 m. We initially observed that the static blocks are at a height (z) of 0.36 m from the camera during its predetermined position. This condition is included in our custom planner to avoid collision.

The goal position is updated in the z direction (0.05m), with each iteration allowing the following blocks to be stacked vertically. After stacking each block, the end effector moves back to the predetermined position to detect the remaining blocks. This is to ensure the positions of the blocks remain updated as the surrounding block's pose might be changed due to external factors.

Dynamic Approach

After stacking the static blocks, the robot moves to a predetermined position for detecting the dynamic blocks. This position is set so the camera's view is restricted to 1/2 of the turntable as we observed our IK solver provided no solution for blocks detected in the center of the turntable, outside our robot's workspace.

After moving to the predetermined position, the command *detector.get_detections()* is called and the name and pose of blocks in the camera's view are obtained and the pose is converted to the world frame. Among all the blocks detected, unreachable blocks are filtered out, and the block with the lowest number (e.g *cube_dynamic9* - dynamic blocks are numbered from 9 onwards) is set as the target and the robot utilizes the advanced planner to reach it. The advanced planner uses a unique approach to grab the dynamic blocks (Figure 7) using a funneling technique. To avoid collision with obstacles and the workspace, the IK solution picks the smallest joint 1 angle proposed in the IK solutions. This approach consists of a sweeping motion from the front to the back of the table to reach the target position. This sweeping motion also prevents collisions that would occur were the robot to descend down vertically to the target position (where the back side of the end effector could collide with the tops of the blocks passing under it).

Similar to the static approach, to adjust the height where the end effector grabs the block, we created a hyper parameter to tune the IK input translation matrix, *ik_input['t'][2] = ik_input['t'][2] - 0.012*. This

matrix was tuned in hardware to ensure that the end effector grabs the block at 0.012m below the center position for guaranteed retrieval. The robot is programmed to pause for 3 seconds while the motion of the table moves the dynamic block to the end effector and orients the block perfectly for grasping. Similar to the static approach, we also observed that the robot tried to grab blocks which were dropped below the dynamic table as the block remained in the camera's view. To avoid collisions with the environment, a simple condition was added in our advanced planner to detect and approach blocks only if the height (z) with respect to the camera is less than 0.39 m. We initially observed that the dynamic blocks on the turntable are at a height (z) of 0.36 m from the camera during its predetermined position. Another condition was written such that if a block is detected outside the workspace, the robot will target the next numbered block until one is found within it.

As the team color (red and blue) impacts the orientation of the robot's workspace, we programmed our method to be applicable for both colors. This was done by programming with the blue robot workspace, then adding a red robot workspace condition by reflecting specific joint angles. The sign (+/-) of joints 1, 3, and 5 were switched to run the program in the red robot workspace.

Evaluations

We extensively evaluated our strategy by conducting comprehensive experiments both in simulation and hardware. We conducted multiple tests to validate the robot's performance and the robustness of our program. Tests were first conducted in simulation, then validated in hardware.

Camera and IK Tests

While testing the IK solution for the end effector to reach a block position, we observed a discrepancy between the simulation and hardware. In the simulation, the camera is mounted directly to the end effector. However, during hardware testing we observed that the camera is mounted to a plate that is mounted to the end effector. The mounting plate had a thickness of 5mm, which resulted in our robot colliding with the turntable during the dynamic approach. To accommodate for this discrepancy, we adjusted the hyper parameter *ik_input['t'][2] = ik_input['t'][2] - 0.012*.

Hardware testing also revealed a complication with the camera. In our initial trials, we positioned the camera at an angle to capture all four static blocks within the camera's view. In the initial static block capture position, the camera mounted on the end effector was not in parallel with the x-y world plane. This position resulted in difficulty detecting all static blocks due to the perspectives in the camera's view increasing noise. Further testing showed that keeping the camera parallel to the x-y world plane resulted in successful detection of all four static blocks.

It is worth noting that during our program development we observed that when multiple blocks (both static and dynamic) are in the camera's view, the detected AprilTag number of the block determines the order by which the list is created in the *target_pose_w*. Blocks with a lower AprilTag number will be selected first.

Static Block Tests

Grasping static blocks was first tested in simulation (Figure 2).

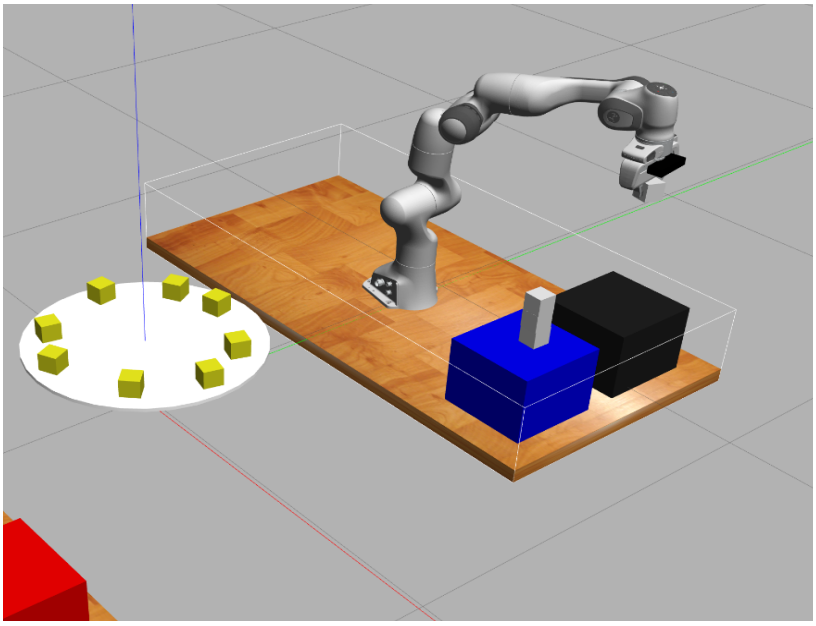
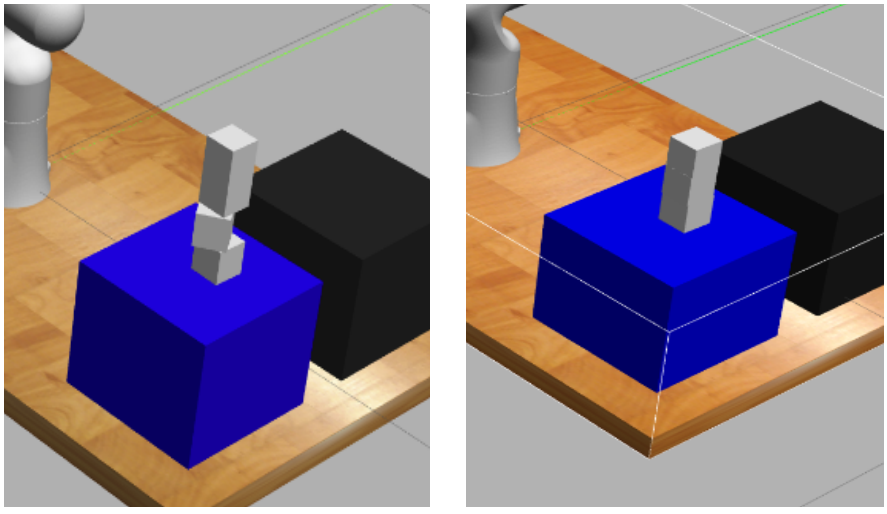


Figure 2: Static block retrieval in simulation

To investigate the impact of block orientation on the stacked block tower, we compared simulations of the end effector directly grasping the block with simulations of the end effector first tapping the block to orient, then grasping the block (Figure 3).



a) End effector grasping directly b) End effector tapping then grasping

Figure 3: Comparison of end effector grasp method and resulting block tower

Seen in Figure 3, tapping the block before grasping resulted in a more secure block tower for stacking.

To evaluate the condition where the robot abandons static blocks below the platform (0.36m), *cube_static_1* was placed below the platform and *cube_static_2* was placed on the static platform. After detecting both blocks, the robot successfully skipped block 1 and set block 2 as the target position.

Dynamic Block Tests

Our program for retrieving dynamic blocks was first tested in simulation (Figure 4).

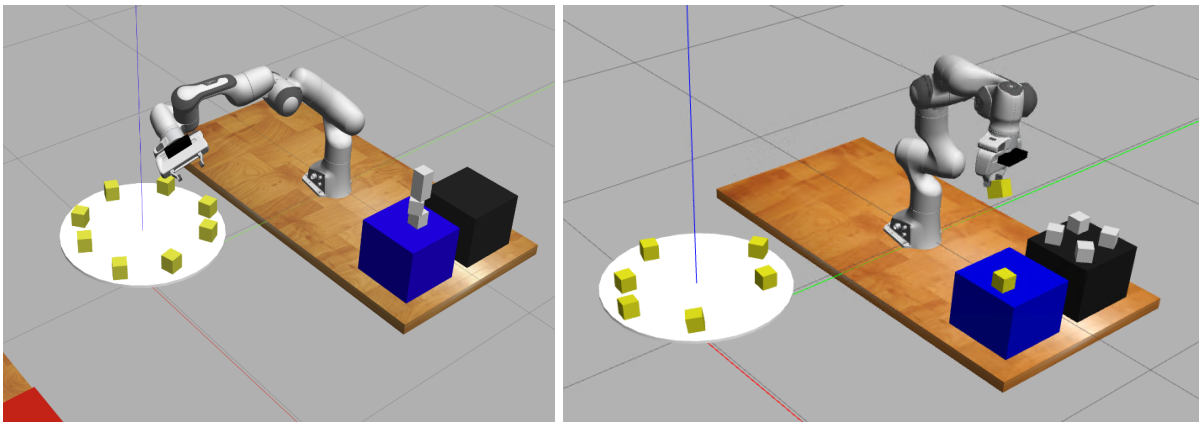


Figure 4: Dynamic block testing in simulation

Similar to the approach used for stacking static blocks, we found that tapping the block prior to grasping resulted in a more stable block tower and incorporated the same method in our dynamic approach.

For grabbing dynamic blocks, we observed a discrepancy between the simulation conditions and hardware. In simulation, the friction between the block and the table was high enough to deflect the end effector from the desired funneling position (Figure 5). To validate the funneling approach, all dynamic block testing had to be done in hardware (Figure 6).

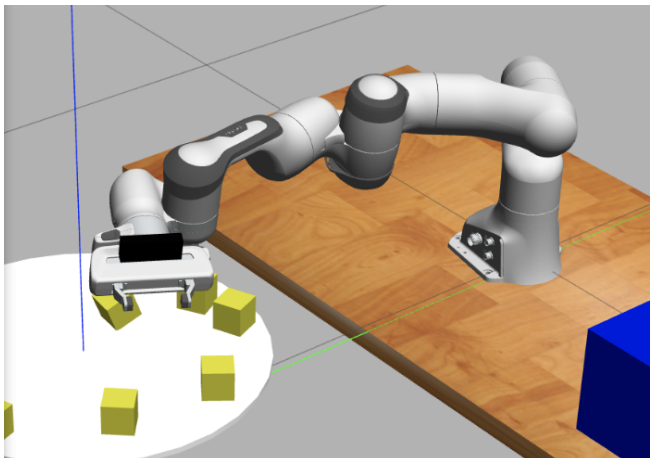


Figure 5: End effector deflection in simulation

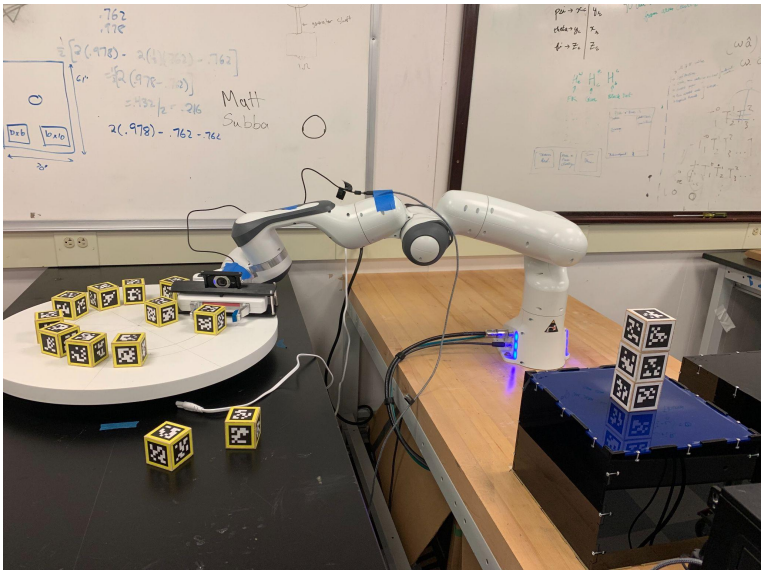


Figure 6: Dynamic block testing in hardware

While developing the dynamic block retrieval method, we initially planned to position the end effector on the outer edge of the turntable and move inwards towards the center of the table. Testing this method resulted in only a 2/5 success rate. Our alternative approach was to sweep the end effector from the back of the table (in the audience view) to the front, keeping an equidistant radius from the center of the table (Figure 7). Testing this method resulted in a 4/5 success rate. The sweeping method also knocked other dynamic blocks off the turntable through the process of funneling, decreasing the number of dynamic blocks available for our opponents (and ourselves), giving us a competitive advantage.

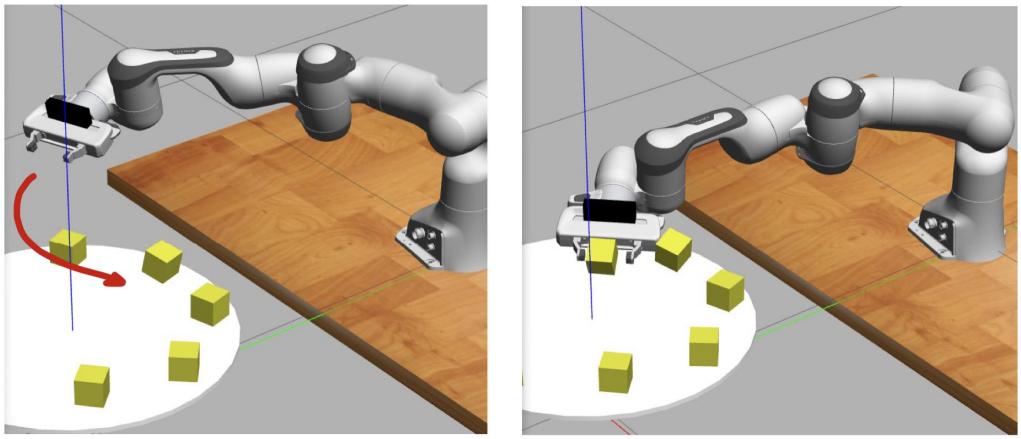
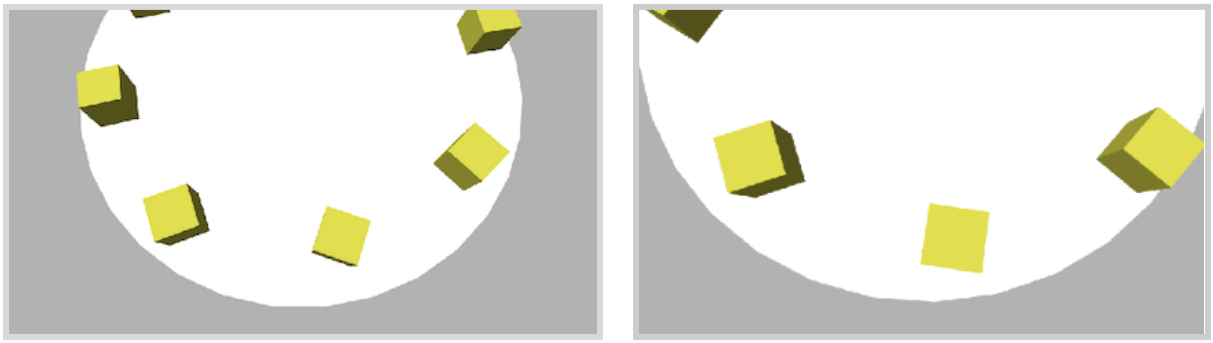


Figure 7: Robot position on turntable for sweeping approach

During hardware testing, we also observed that, in addition to knocking other dynamic blocks off the turntable, this sweeping approach cleared the blocks from a slice of its circular surface, which subsequently rotated to our opponents side for them to find it empty. The blocks kept piling onto our arm

without ever reaching the opponent's. This resulted in them having to wait for a dynamic block until after we had already retrieved ours - wasting their time and giving us an advantage.

In picking the area of the turntable in which we wanted to detect blocks, we started with around 3/4 of the turntable in the camera's view. However, we observed that the end effector would detect blocks near the center of the table and provide no IK solutions as those areas were outside the robot's workspace. The blocks at the center of the table were unreachable to us and would remain unreachable throughout the turntable's rotation. To avoid wasting time processing these blocks and solving IK for them, we adjusted the camera's perspective to capture 1/2 of the turntable to ensure most of the blocks detected are within the robot's workspace. Implementing the 1/2 view by adjusting our joint configuration yielded successful detection of only blocks within the robot's reach. The 3/4 and 1/2 camera views are shown in Figure 8.



Camera view for 3/4 of turntable | Camera view for 1/2 of turntable

Figure 8: Comparing Camera View

To evaluate the condition where the robot abandons dynamic blocks below the platform (0.36m), *cube_dynamic_9* was placed below the turntable and *cube_dynamic_10* was placed on the turntable. After detecting both blocks, the robot successfully skipped block 9 and set block 10 as the target position.

Block Stacking Tests

Scoring of the competition depends on the type of block and the altitude of the block (distance in mm from the center of the block to the goal platform) where $Points = Block\ Value \times Altitude$. Static blocks are valued at 10 while dynamic blocks are valued at 20. To maximize our points scored, we aimed to stack all static and dynamic blocks on the goal platform.

In our strategy, along with tapping the block to orient it properly prior to grabbing, we adjusted the goal position upwards in the z direction each time a block was placed so that the robot would position it directly above the previous one. While this worked in simulation, in hardware we observed cases where dynamic blocks were stacked out of line with the previous ones in the x-y plane due to a misaligned end-effector grasp (i.e. friction causing block to slip from gripper).

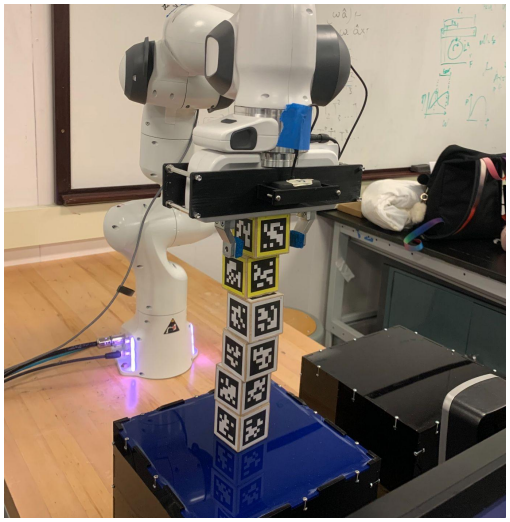


Figure 9: Hardware testing of stacking block tower

As seen in Figure 9, the block tower is stacked unevenly. We attempted to increase the gripper force however the tips of the gripper did not have enough friction to avoid slipping and were also naturally out of line because of the aforementioned noise in the block coordinates the camera gave us. We also considered placing each dynamic block down and re-orienting the end effector grasp, however this approach would take up precious time and (again due to noise) better results were not guaranteed. After further hardware testing, we decided that the uneven stacking of dynamic blocks had no significant effect on the stability of the tower. As misaligned grips were only present with the dynamic blocks (which would be stacked last) we chose to risk uneven stacking for the reward of time saved and algorithmic simplicity.

Strategy Tests

During hardware testing, we observed cases where the robot dropped a block due to lack of friction between the gripper and the block as well as an improper (non-centered) grip position. To address this, we added a check when the robot reaches the *above_goal_location* to ensure that the block is still being held. If the gripper force is equal to 0N, the robot will stop, position the camera to scan for the dropped block, and attempt to pick up the target block again.

In hardware testing, the sequential approach (which is discussed in greater detail in the Static Approach Method section) was the safest method for avoiding collisions and saving time. As time is an important factor in the competition, we also investigated the impact of joint 7 actuation on retrieval time. We tested two actuation sequences. The first sequence involved actuating joints 1, 3, 5 followed by actuating joints 2, 4, 6, then 7 and the second sequence involved actuating joints 1, 3, 5 followed by actuating joints 2, 4, 6, and 7. We found that the second sequence saved 22 seconds (12% of the competition round time, pre 5 minute extension) picking up 4 static blocks compared to the first.

To further develop our competition strategy, we recorded the time taken to retrieve certain numbers of static blocks. The average times are shown in Table 1.

Table 1: Average time taken to retrieve static blocks

Avg time to retrieve 3 static blocks (s)	Avg time to retrieve 4 static blocks (s)
80	114

These times informed our competition strategy. To maximize the points scored, we calculated the total points for three strategies. It is important to note that since $Points = Block\,Value \times Altitude$, stacking dynamic blocks last (at a higher altitude) will result in a higher score.

Strategy 1 was to retrieve 4 static blocks and 2 dynamic blocks, strategy 2 was to retrieve 4 static blocks and 1 dynamic block, and strategy 3 was to retrieve 3 static blocks and 2 dynamic blocks. The point breakdown for strategy 1, 2 and 3 are shown in Table 2, Table 3, and Table 4 respectively.

Table 2: Point Breakdown for Strategy 1

Block Points	**Altitude Points**	**Total Points**
10	25	250
10	75	750
10	125	1250
10	175	1750
20	225	4500
20	275	5500
Final Score		14000

Table 3: Point Breakdown for Strategy 2

Block Points	**Altitude Points**	**Total Points**
10	25	250
10	75	750
10	125	1250
10	175	1750
20	225	4500
Final Score		8500

Table 4: Point Breakdown for Strategy 3

Block Points	Altitude Points	Total Points
10	25	250
10	75	750
10	125	1250
20	175	3500
20	225	4500
Final Score		10250

Analysis

In the first two 5 minute rounds, we implemented strategy 1 (Table 2) successfully and won both rounds. For the remaining qualifying rounds, we observed each competing team's strategy to inform our strategy for the semi-finals. From our observations of our semi-final opponent, we found that strategy 1 was sufficient to win (Table 2).

We observed that our final round opponent stacked only 5 blocks before the robot stopped moving. This informed our strategy for the final round and we kept strategy 1 (Table 2) as it stacks 6 blocks instead of 5 (post time extension). This ensured we would score higher than our opponent and win the competition.

One specific challenge was observed during the competition: light was being reflected off the static block platform and interfered with the camera's block detection abilities, ending up with the 4th block being completely missed. This occurred in two rounds in which the robot simply moved on to the dynamic blocks after placing the third static block as we had designed it to do if no static blocks were detected.

Upon further analysis of our competition performance and strategies, we would have liked to incorporate some additional conditions to improve the robustness of our program:

- In the beginning of a round, after retrieving 3 static blocks, the robot could detect the dynamic blocks. If there are more than 5 dynamic blocks on the turntable, the robot should retrieve dynamic blocks before retrieving the last static block.
- To improve block stacking, after placing the 4th block, the robot could position the camera in the updated goal position (in the z direction) and view the tower from the top in the x-y world frame. The robot could scan and measure the x and y coordinates of the tower. If an offset from the expected x and y coordinates was detected, the robot could calculate the error and place the block in such a way as to correct the offset and balance the stack.

- To improve our strategy, we could add a condition for the robot to scan the opponent's block tower after we had stacked 3 static blocks or when the time reached 2 minutes and 30 seconds (whichever happened first). If the opponent's tower was taller than ours, the robot could begin dynamic block retrieval to catch up points-wise.

Summary of Lessons Learned

1. Joint actuation sequence was integral for avoiding collisions. When actuating all joints at once, the robot was hitting and pushing the target static block while reaching the target position. Through further testing we learned that actuating joints 1, 3, and 5, followed by joints 2, 4, 6, and 7 was the safest method for avoiding collisions.

2. Hardware testing revealed a complication with the camera. In our initial hardware tests, we positioned the camera at an angle to capture all four static blocks within the camera's view. In the initial static block capture position, the camera mounted on the end effector was not in parallel with the x-y world plane. This position resulted in difficulty detecting all static blocks due to the perspectives in the camera's view increasing noise. Further testing showed that keeping the camera parallel to the x-y world plane resulted in successful detection of all four static blocks.

3. One specific challenge was observed during the competition: light was being reflected off the static block platform and interfered with the camera's block detection abilities, ending up with the 4th block being completely missed. This occurred in two rounds in which the robot simply moved on to the dynamic blocks after placing the third static block as we had designed it to do if no static blocks were detected. We learned that light reflection and the resulting noise is unavoidable in certain environments and that the end effector must move closer to the static block platform during detection.

4. We observed cases where the robot tries to grab dropped blocks that landed below the goal platform (blue platform / red platform) and the turntable since the block is in the camera's view. Approaching a dropped block resulted in collision with the workspace. We learned to avoid this by adding a simple condition for the robot to detect and approach blocks only when the z height with respect to the camera is below 0.39 m.

Conclusion

We are very satisfied with our implementation of coding and the methodology learned in this course to accomplish the tasks required by the competition. Rigorous testing in simulation and hardware allowed us to be confident in the robustness of our robot performance and feel ready – even optimistic! – for the competition. Though there are additional features that could improve the robustness of our robot performance, we were able to apply well-designed strategies to beat our opponents and come out first.