Final Project Report

Ethan Senatore & Milad Mesbahi & Dea Rrozhani & Eric Wang

MEAM 5200: Introduction to Robotics

(June 2, 2025)

1 Introduction

The MEAM 5200 final project afforded us a phenomenal opportunity to explore the simulation-to-hardware divide for robotic manipulators. Each step in solving the pick-and-place problem incentivized us to weave together concepts we have learned over the course of the semester. Our approach ultimately fell short when testing on hardware, which resulted in a more than lack luster competition performance. Despite the bitter performance, we learned about the flaws in our approach and what showed promise. Our report highlights the theory necessary to implement our approach, a systematic breakdown of our closed-loop solution, and an analysis on what we learned.

2 Methods

In this section, we will discuss the methodology used and theoretical framework that supported our approach for the final project. First and foremost, we will highlight the theory we have already covered in previous labs in order to establish the foundation of our approach. As always, *Robotic Modeling and Control*[1] acted as our main source of theory. We first begin by discussing the derivation of the forward kinematics of the robot. Before conducting any scans with the onboard camera, we established the frame of the camera in the world coordinate system. We did this first by establishing the pose of the end effector T_0^e in the world frame. Then, using the software provided to us by the TA's, we calculated the camera's pose in the end-effector's frame T_e^C. To get the pose of the camera in the world frame, we simply took the dot product between the two:

$$T_0^C = T_0^e T_e^C \tag{1}$$

Next, we needed to accurately detect and correct the poses scanned by the camera. We found that the rotation matrix of each cube was encoded with information regarding the true pose of the block. We conducted our initial scans with the following pose:

$$\underset{Blue}{T_0^C} = \begin{bmatrix} 1 & 0 & 0 & 0.532m \\ 0 & -1 & 0 & 0.1562m \\ 0 & 0 & -1 & 0.5215m \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \underset{Red}{T_0^C} = \begin{bmatrix} 1 & 0 & 0 & 0.532m \\ 0 & -1 & 0 & -0.1562m \\ 0 & 0 & -1 & 0.5215m \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2}$$

which each equated to the configurations:

$$\underset{Blue}{q} = \begin{bmatrix} \frac{\pi}{11} & 0 & 0 & \frac{-\pi}{2} & 0 & \frac{\pi}{2} & \frac{\pi}{4} \end{bmatrix} \qquad \underset{Red}{q} = \begin{bmatrix} \frac{-\pi}{11} & 0 & 0 & \frac{-\pi}{2} & 0 & \frac{\pi}{2} & \frac{\pi}{4} \end{bmatrix} \tag{3}$$

With this scanning pose, we guaranteed that one of the three columns of the block's rotation matrix was always:

$$r_i \approx \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

After identifying the column of interest, we could easily adjust the orientation in order to align the z-axes. The orientation of the block was arbitrary in a physical sense, but was a prominent issue we ran into in developing our solution. In order to properly grasp each block, we determined that a new *pseudopose* with the block's z-axis aligned with the camera's z-axis was required. This was done to avoid any awkward or dangerous poses for the arm. We accomplished this by first detecting which of the three columns of the block's pose rotation matrix was aligned with the camera's z-axis. Upon doing so, we rotated the matrix by $\pm\frac{\pi}{2}$ depending on whether the block's x or y-axis was aligned with the camera's z-axis. If the z-axes were aligned then a simple check and correction of whether they faced the same direction was completed. Algorithm 1 captures our approach in more detail. Table 1 captures all possible scenarios and adjustments we would potentially make.

Column	Parallel Axes?	Rotation Made
0	True	$R_{y,\frac{\pi}{2}}$
0	False	$R_{y,\frac{-\pi}{2}}$
1	True	$R_{x,\frac{-\pi}{2}}$
1	False	$R_{x,\frac{\pi}{2}}$
2	True	I
2	False	$R_{y,\pi}$

Table 1: Angle adjustment scenarios and rotations

Following the adjustment of the block's pose, we further verified the validity of the pose by checking if a solution could be reached using our previously defined inverse kinematics function. We chose to use the pseudoinverse method along with a value of 0.30 for our learning rate (α). If our inverse kinematic algorithm failed to converge to a solution, we concluded that the most likely cause of error was the final wrist joint of the arm being pushed beyond its limits. To account for this flaw, we simply conducted a rotation of $\frac{\pi}{2}$ about the z-axis of the pose, until a valid solution was found.

In anticipation of potential noise being encountered during hardware testing, we added additional fail-safes. The first of which includes an averaging of fifty scans in order to combat noise. It is important to note that this decision contributed to our failure and we will discuss why in our Section 4. Next, we added a re-scan system, whereby the arm was positioned above and slightly behind a block in order to conduct an additional scan of its April tag. We found that an offset of [-0.025m, 0m, 0.1m] was a comfortable position above the block to conduct a re-scan. This was done to better calculate the pose of a block, as a better positioned scan led to a more accurate pose. To summarize, we first took an overall scan over the static block area to scan all four blocks. Next, we approach the first of the detected blocks and performed a rescan at a closer position. We noticed that even at closer positions, the camera occasionally captured other blocks. To ensure the correct block was scanned, we added id-verification. Once a rescan was completed, we lowered the gripper and grasped the block. We then raised the arm, first to its rescan position, and then the

neutral position as referenced in Eqn. 2 & 3. We then positioned the arm in its neutral dropping configuration, which was hard coded in order to save time. The hard coded positions can be seen below.

$$\underset{Blue}{q} = \begin{bmatrix} -0.1286 & 0.07215 & -0.13995 & -1.53771 & 0.01006 & 1.60915 & 0.51682 \end{bmatrix}$$

$$\underset{Red}{q} = \begin{bmatrix} 0.15668 & 0.07189 & 0.11041 & -1.53771 & -0.00792 & 1.60917 & 1.05251 \end{bmatrix}$$

What remained was to successfully drop each block in the correct position. We picked an arbitrary drop-off location with an initial position resting atop the drop-off table. Following a drop, we would reset to the neutral drop-off position and conduct an April tag scan. If the block-id matched that of the block-id we initially grasped then a successful drop was made, and 0.05m was added to the z-position of the drop-off position. If not, we then prompted the system to restart its pick-up algorithm after fully iterating through each of the detected blocks. A more detailed description of this process can be seen in Algorithm 2.

This overall process was repeated until each block was dropped off. An overview of our approach can be found in Algorithm 3. In an ideal world, all four static blocks would be picked up and dropped in the correct positions. As we will discuss, the simulation-to-hardware bridge was difficult to cross, and our approach proved to be unsuccessful.

Dynamic Blocks

The dynamic block task required real-time detection, grasping, and drop-off of blocks from a rotating platform. Given the significant challenges encountered in accurately determining block poses during the static block phase, we adopted a robust and simplified methodology for dynamic blocks that relied on a hardcoded sweeping strategy.

Initially, we attempted to apply the same pose detection and adjustment pipeline used for static blocks. The process involved:

- Detecting block poses using AprilTags and converting them to world coordinates.
- Correcting the pose by aligning the block's z-axis with the camera's z-axis (Algorithm 1).
- Verifying grasp feasibility using the pseudoinverse inverse kinematics (IK) solver.

However, this approach proved to be quite error-prone for dynamic blocks. The primary challenges included:

- **Pose Adjustment Complexity:** Blocks on the turntable exhibited significant motion, making pose detection noisy and requiring multiple rescans for accuracy.
- **Time Sensitivity:** The rotating table introduced a temporal constraint that limited the feasibility of iterative pose adjustment and IK validation.
- **Simulation-to-Hardware Gap:** While pose alignment succeeded in simulation, real-world testing revealed discrepancies in calibration and system latencies.

These difficulties as well as the time constraint we were facing led us to adopt a more deterministic approach based on pre-defined sweeping motions.

To ensure reliable block pickups, we implemented a hardcoded sweeping strategy that eliminated the need for real-time pose adjustments. The arm performed a series of pre-programmed motions that covered the dynamic turntable area systematically. The methodology is as follows:

1. **Initialization:** The robot arm was moved to a pre-defined "home position" near the rotating table to establish a stable starting configuration:

$$q_{\text{home}} = [0.5586, 1.3334, 0.8116, -0.7220, 0.6709, 1.0898, -1.0656] \tag{4}$$

2. **Sweeping Motion:** The arm executed a sweeping trajectory defined by start, intermediate, and end positions. The joint configurations for these positions were hardcoded as:

$$q_{\text{sweep\_start}} = [0.6547, 1.0217, 0.7461, -1.1245, 0.5413, 1.7202, -1.1701]$$
$$q_{\text{sweep\_end}} = [0.8206, 1.0217, 0.7461, -1.1245, 0.5413, 1.7202, -1.1701]$$

   These positions ensured complete coverage of the table surface while maintaining a safe clearance above the blocks.

3. **Grasp Attempt:** After completing the sweep, the gripper closed to attempt a block grasp:

$$\text{Gripper Command: } \texttt{exec\_gripper\_cmd(0.03, 60)} \tag{5}$$

   A successful grasp was inferred by checking the gripper's finger positions. If the total finger distance was less than 0.025 m, the grasp was deemed unsuccessful, and the sweep was adjusted or repeated.

4. **Adjustment Sweep:** If no block was successfully grasped during the initial sweep, the arm executed an adjusted sweep:

$$q_{\text{sweep\_adjust}} = [0.8556, 1.0217, 0.7461, -1.1245, 0.5413, 1.7202, -1.1701] \tag{6}$$

5. **Drop-off:** Upon detecting a successful grasp, the arm moved back to the dynamic home position and called the drop-off function to place the block at the designated location.

This methodology prioritized reliability and practicality over precision. The deterministic nature of the sweeping strategy reduced computational overhead and mitigated the impact of noisy detections, providing a straightforward yet effective solution for the dynamic task. While further refinements may optimize its efficiency, and the simulation-to-hardware gap was yet to be seen, the hardcoded sweeps demonstrated a reliable method for dynamic block pickup within the context of strategy for the competition.

3 Evaluation

3.1 Static Blocks

Throughout our initial testing phase, we adopted the basic goal of successfully grasping and placing a single block. We took advantage of both the simulation and the hardware environment to iterate and improve our approach. Our simulation testing was fairly straightforward as it enabled us to test our approach at a high level. The lack of noise, quick repeatability of testing, and flexible environment were incredibly helpful. However, issues within the environment did present themselves which we discuss in further detail in Section 4.

Initially, we did not make note of any quantitative data regarding our tests as we felt that the minute details of our performance paled in comparison to the overall success of the approach. In essence, we were very pleased if we successfully achieved our goals, regardless of the time taken and the precision with which our placements were performed.

Our initial goal with simulation testing was to successfully grasp and place a single block. This proved to be quite a daunting task, but once completed, it acted as a catalyst to expand our approach to all four blocks. Figures 1 & 2 capture our approach in action as we successfully grasped and placed a block.

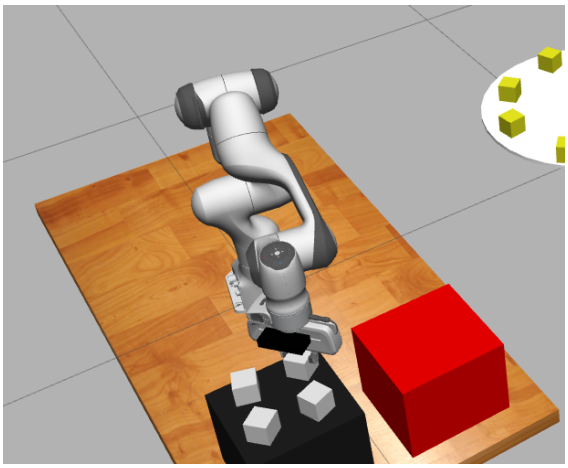

Figure 1: Grasping the block

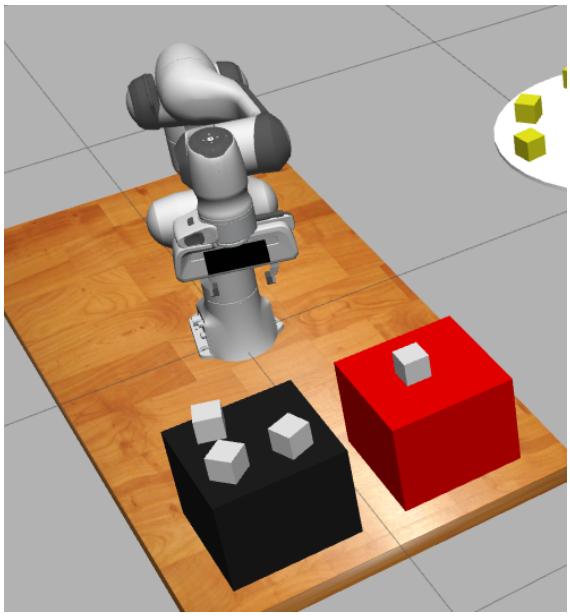

Figure 2: Successful placing of the block

Once we had accomplished our first goal, we extrapolated our approach to all four blocks and repeated the same simulation test. Figures 3 & 4 capture our closed-loop system that successfully placed the second and fourth blocks. Following this result, we begun to capture data that measured the performance of our approach. Table 2 shows the measurements taken over five simulation tests. We observed a 100% grasp rate, an average pickup time of 14.8s, and an average drop-off time of 9.6s. Overall, on average, to stack four blocks took 4min 36s. We felt this value was a bit inflated due to the time it took for the ROS control package to process the arm commands. Over all simulation tests we ran, including those in which we did not track performance metrics, the grasping and

placement of blocks were incredibly precise. As a result, we neglected to track positional metrics measuring how well a block was grasped or placed.

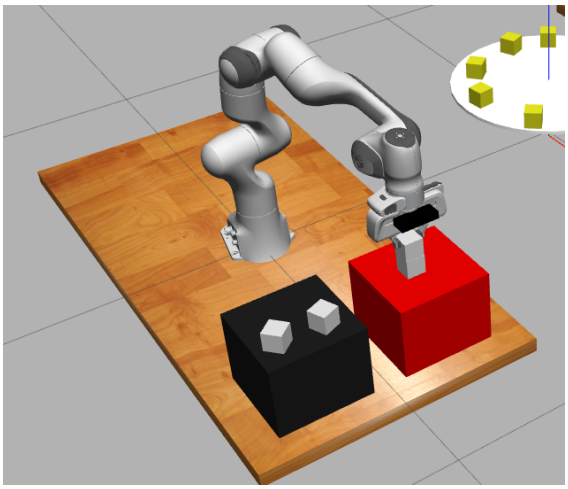

Figure 3: Stacking two blocks

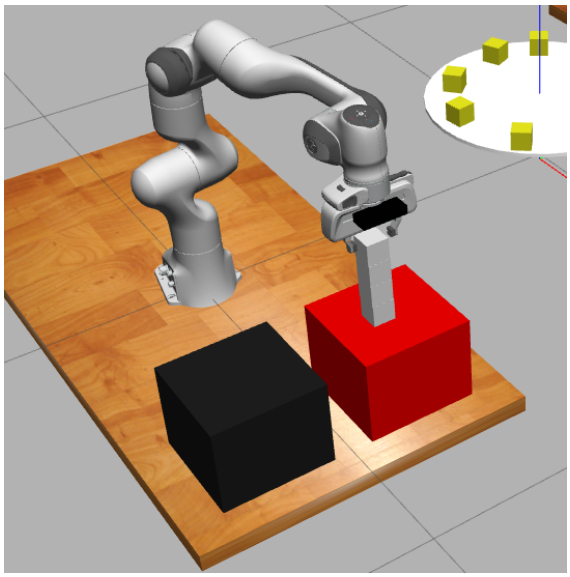

Figure 4: Stacking four blocks

Metric	Measurement
Grasp Success Rate	100%
Average Time of Scan	5.3s
Average Time of Re-Scan	1.1s
Average Time per Pickup	14.8s
Average Time per Drop-Off	9.6s
Average Total Time	4min 36s

Table 2: Simulation Performance Metrics

Testing on hardware proved to present more challenges than we had initially anticipated. First and foremost, the camera had a different FOV and struggled with noise. Secondly, the manipulator required us to tinker with the positional offsets of scanning, grasping, and dropping blocks. Due to time constraints, we did not capture any quantitative data regarding our static approach. Instead, we conducted tests on our approach and made on-the-fly adjustments in order to observe how the robot behaved. Figure 5 shows the robot after having successfully placed a single block; however, this was one of the only solid results that we achieved throughout the testing and the competition. Despite the lack of success on hardware, our observations assisted us in improving our approach to be more adaptable and robust. Some of the notable observations we made were as follows:

1. **Camera Noise**: The hardware camera proved to be noisy resulting in the pose scanned being inaccurate. Additionally, it took around one second for the camera to refocus and adjust after the manipulator was moved. This affected the quality of the scan as the camera did not have enough time to adjust to new lighting and refocus.

2. **Positional Discrepancies**: Perhaps due to the camera noise or for minor mechanical discrepancies, the gripper seemed to always position itself slightly above where the grasping position was supposed to be. We measured that it was roughly 0.02m off in the z-direction.

3. **Speed Discrepancies**: The one positive observation we made was that our approach ran incredibly fast on hardware in comparison to the simulation.

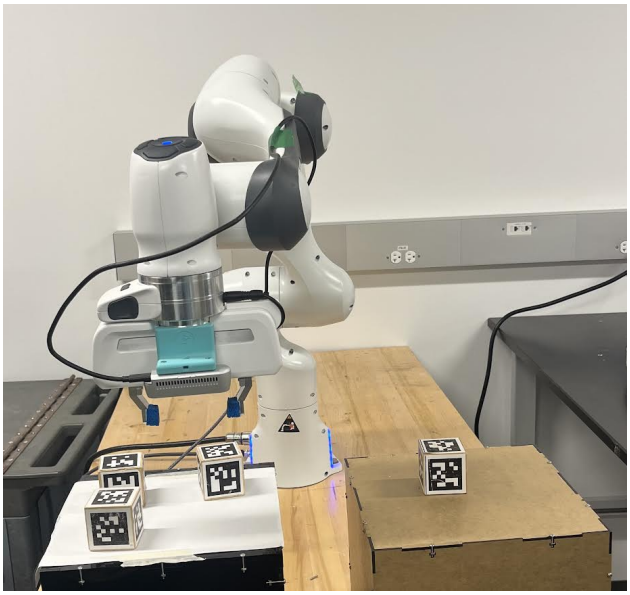

Figure 5: Hardware Test

A less pertinent observation we made was the lack of difference between being on the blue or red robot. All that had to be done to adjust was mirroring the hard-coded positions to scan and place the blocks.

The ultimate evaluation came in the form of competition day. Unfortunately, in both matches we competed in our system completely failed as we were unable to scan or act upon any blocks. We identified the main failure point as the orientation adjustment algorithm that we had settled on. In Section 4 we provide a comprehensive analysis of why it failed, what led us to such a failure, and how we would have done things differently.

3.2 Dynamic Blocks

For dynamic blocks, the transition to a hardcoded sweeping strategy demonstrates a practical trade-off between precision and reliability. Given the real-time constraints of the rotating table and inaccuracies in pose detection, our initial pose adjustment method proved impractical. The sweeping strategy, while simplified, ensured coverage of the turntable and reduced computational overhead.

Due to the issues with the implementation for static blocks and the problem with scanning on competition day, we were unable to test these new changes on hardware. However, our adjusted strategy proved to be more efficient simulation when attempting to pickup one dynamic block.

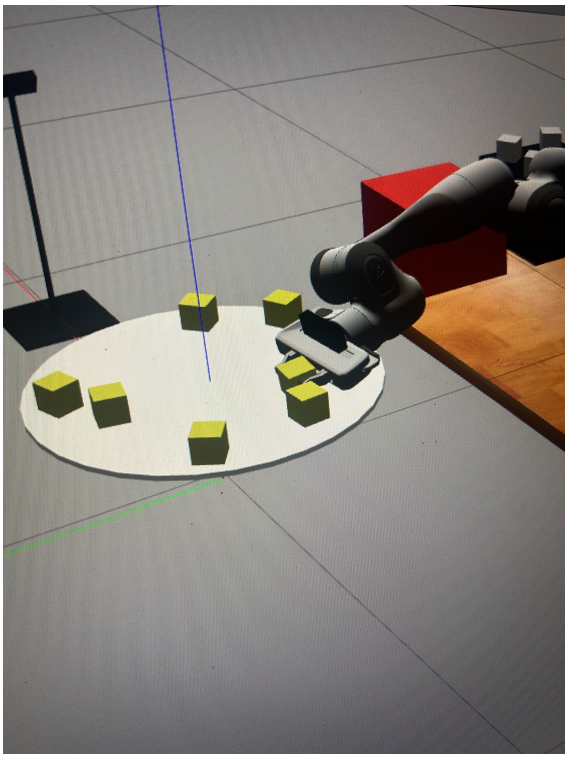

Figure 6: Beginning of Sweep

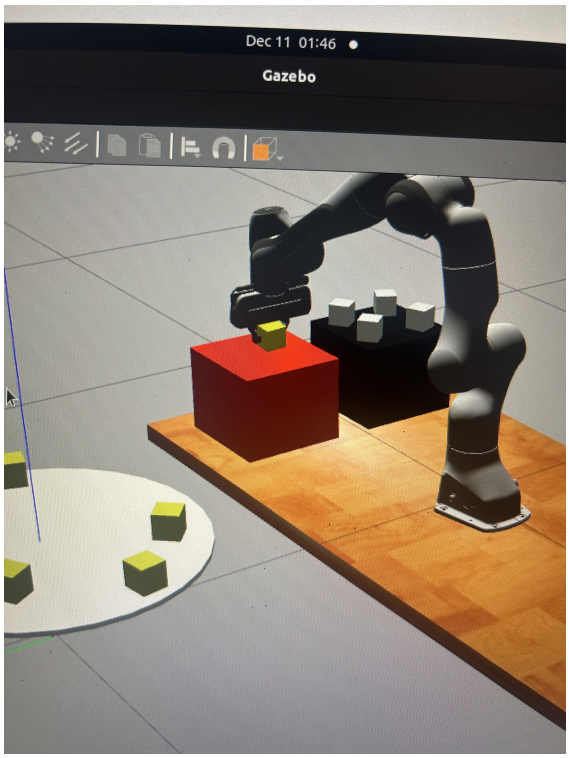

Figure 7: Dynamic Drop-off

The key performance metrics we looked into for the dynamic blocks were grasp success rate and time efficiency. All experiments were conducted in a simulated environment due to time constraints and the complexity of deploying the solution on hardware. The simulation replicated real-world conditions, including block motion on a rotating turntable, variable orientations, and possible misalignments.

The results of the dynamic block evaluation are summarized in Table 3. Each metric provides insight into the system's capabilities, highlighting both its strengths and limitations. The table summarizes results of running the dynamic block simulation 7 separate times from a fresh starting point. The grasp success rate is an indication of how successful the arm was in picking up a block once making contact. The average number of sweeps indicates how many times the arm had to re-run the sweeping trajectory due to not successfully picking up a block or completely missing the target.

Metric	**Result**
Grasp Success Rate	71.4%
Average Time per Pickup	14.2 seconds
Average number of Sweeps	1.43
Grasping Duration	2.5 seconds
Return-to-Drop-Off Duration	6.7 seconds

Table 3: Evaluation Metrics for Dynamic Block - Full Table

We also wanted to evaluate the robustness of the dynamic block strategy under scenarios where fewer blocks were available on the table, mimicking a mid-to-end-game situation where the opposing team might have already removed several blocks. This was necessary to indicate how well the system could locate, adjust, and grasp blocks in more sparse environments, where the blocks are less likely to be within easy reach.

We conducted seven independent trials of the dynamic block strategy in this environment, maintaining consistency with our more block-rich environment, measuring the same metrics. The results are summarized in Tabl e4 .

Metric	**Result**
Grasp Success Rate	51.14%
Average Time per Pickup	14.8 seconds
Average number of Sweeps	2.71
Grasping Duration	2.5 seconds
Return-to-Drop-Off Duration	6.7 seconds

Table 4: Evaluation Metrics for Dynamic Block - Blocks Removed

4 Analysis

4.1 Static Block

The simulation environment offered us the chance to test and improve our approach at any point throughout our development period. However, we were hampered by two glaring issues. The first was the slow speed at which the simulation ran. We found that our simulation ran at between 0.15 and 0.31 real-time speed. This made testing quick fixes particularly slow. The second issue was the simulation-to-hardware divide. Despite being able to successfully stack four blocks in simulation, we were unable to replicate those results on hardware. As was mentioned in Section 3, noise and slight positional offsets acted as roadblocks in our development. What became apparent was that the simulation environment essentially became futile, and all that was left was to overcome issues that only presented themselves on hardware. Due to the limited time, and perhaps our haphazard testing approach, we were unable to sort out the main issues we encountered.

Ultimately, what caused our lackluster competition performance was the orientation adjustment algorithm. This, coupled with the averaging of poses over multiple scans, plagued our system and caused our robot to be ineffective. What led us to the decision to both average scans and adjust the orientation of the blocks? The noise that presented itself during testing proved to be a major annoyance in our approach. We heavily relied on our scans to be accurate in order to correctly adjust the pose so that the gripper could safely grasp each block. Why did it fail in testing? We found that conducting an orientation adjustment at each iteration of a scan slowed down our approach far too much. Instead, we should have first averaged the scans and then performed an adjustment. In this order, we would have not only ensured a better scanning accuracy, but also sped up our approach.

Future work could further refine this process and look to simplify some of the fail-safes. Furthermore, additional testing would allow for minute improvements that may unlock this approach and prove it to be successful. We do feel that, although we had failed on hardware, our promising

results in simulation hinted at a solid approach. Despite the failure, we were proud of what we had accomplished and felt as if we were one or two developmental iterations away from success.

4.2 Dynamic Block Results

The evaluation of the dynamic block strategy focused on its performance in simulated environments, examining two distinct scenarios: a standard environment with plentiful blocks and a sparse environment mimicking mid-to-end game conditions where fewer blocks remained on the table. The results demonstrated both the strengths and limitations of the adopted sweeping strategy, shedding light on areas for future improvement.

In the standard dynamic block environment, the strategy achieved a grasp success rate of 71.4%, with an average pickup time of 14.2 seconds per block. The average number of sweeps required to secure a block was 1.43, highlighting the efficiency of the pre-defined sweeping motion in environments where blocks were abundant. The time taken for grasping and returning to the drop-off location remained consistent at 2.5 and 6.7 seconds, respectively. This consistency in handling times underscores the reliability of the robot's motion planning and execution capabilities for grasping and drop-off once making accurate contact with a block.

In the sparse block environment, designed to simulate mid-to-end game conditions, the grasp success rate dropped to 51.14%, reflecting the increased difficulty of locating and securing blocks when fewer were present. The average time per pickup increased slightly to 14.8 seconds, while the number of sweeps required rose to 2.71. Despite this increase, the system maintained stable grasping and drop-off durations, indicating that the arm's handling mechanics were robust across varying conditions.

The results reveal several key observations. First, the success rate variability highlights the dependency of the sweeping strategy on block density. While the strategy is highly effective in the standard, beginning game environment, its performance diminishes in sparse scenarios due to the reduced likelihood of blocks intersecting the sweeping path. Second, the consistent handling times across all scenarios demonstrate the strength of the robot's motion control and the deterministic nature of the sweeping approach. Third, the increase in sweeps in sparse conditions points to inefficiencies in block localization when their density decreases, suggesting a need for improved adaptability.

Despite its strengths, especially from the standpoint of practicality, the sweeping strategy's limitations became evident in the sparse environment. The hardcoded nature of the sweeps restricts the system's flexibility to dynamically adapt to varying block distributions. It relies too heavily on fixed distributions and block paths. Furthermore, the reliance on gripper feedback for grasp validation, while effective in standard scenarios, proved inadequate in edge cases where blocks were partially grasped or slipped during transport. These limitations emphasize the importance of integrating dynamic perception capabilities to enhance adaptability and performance.

In conclusion, the dynamic block strategy successfully balanced simplicity and effectiveness in high-density scenarios for the minimum requirements of the competition environment, providing reliable pickups with minimal computational demands. However, its diminished performance in sparse environments underscores the need for enhancements in perception and adaptability. Future work could focus on integrating real-time vision systems and dynamic path planning to complement the sweeping strategy, enabling the robot to excel in more complex and varied conditions. In the future, we also need to test the dynamic block strategy on hardware to confirm its performance under real-world conditions. Many of the strengths and limitations observed in simulation would

likely translate to hardware, albeit with some parameter tuning, such as adjusting the sweep positions. The strengths of avoiding challenges like varying lighting conditions or perception noise would remain. However, the limitations, such as the lack of adaptability and lower success rates in sparse block environments, would also persist. Nonetheless, we regret not being able to explore these aspects further and believe more work in this area could provide valuable insights.

5 Conclusion

Albert Einstein once said, "Failure is success in progress." Although we did not achieve a desirable competition result, the development process and what we had learned proved invaluable. The final project offered us a chance to use an iterative approach to solve a complex engineering problem. At each step of the development cycle, we observed improvements that in turn motivated us to continue working. We had hoped to have accomplished more, but we are proud of what we produced. This project gave us a unique perspective and a greater degree of appreciation for the engineers that work on complex projects that are put to use in real life.

6 Appendix

Algorithm 1 Orientation Correction Algorithm

Input: *pose* (4x4 transfomation matrix of the pose of a block)
Output: *newPose* (4x4 transfomation matrix of the corrected pose)
$R \leftarrow$ 3x3 rotation matrix within the 4x4 pose
$T \leftarrow$ 1x3 translation matrix within the 4x4 pose
for column in R **do**
  if column $\approx [0, 0, 1]^T$ **then**
    flip = -1
    $idx \leftarrow i$
  else if column $\approx [0, 0, -1]^T$ **then**
    flip = 1
    $idx \leftarrow i$
  end if
end for
$\theta = \frac{pi}{2} \cdot flip$
if $R[i] = 0$ **then**
  $R = RR_{y,\theta}$
else if $R[i] = 1$ **then**
  $R = RR_{x,-\theta}$
else
  $R = RR_{y,\frac{\pi}{2}}$
end if
$pose \leftarrow [R|T]$
Return *pose*
=0

Algorithm 2 Drop-off Algorithm

Input: id (block id)
Move to neutral drop position
Move to drop position
Release block and reposition in neutral drop position
$detectedID \leftarrow$ Conduct a re-scan and assign id detected
if $id = detectedID$ **then**
  $T_{drop}[z] \leftarrow T_{drop}[z] + 0.05$
else
  $restart \leftarrow True$
end if
=0

Algorithm 3 Static Block Algorithm

Move to neutral scanning position
$blocks \leftarrow$ Scan using Alg.4
for block in $blocks$ **do**
  Approach block
  $pose \leftarrow$ conduct rescan using Alg.4
  Pick up block
  Move to neutral drop position
  Drop-off block
  Move to neutral drop position
  $checkID \leftarrow$Conduct rescan using Alg.4
  if block not in $checkID$ **then**
    $restart \leftarrow$ True
  end if
end for
=0

Algorithm 4 Scanning Algorithm

```
poses ← {}
for i in range(50) do
  blocks ← scan blocks in FOV of camera
  for block in blocks do
    id ← blocks[block][1]
    pose ← adjusted orientation using Alg. 1
    if id in poses then
      poses[id] ← pose[id] + pose
    else
      poses[id] ← pose
    end if
  end for
end for
Return poses
=0
```

Algorithm 5 Dynamic Block Handling Algorithm

Require: Predefined joint configurations for sweeping positions: dynamic_home_position, dynamic_sweep_start, dynamic_sweep_end, dynamic_sweep_adjust.

Ensure: Blocks are picked up and dropped off reliably.

```
Move to dynamic_home_position.
Open gripper to maximum width.
Set max_attempts ← 10, attempt ← 0.
while attempt < max_attempts do
  Increment attempt ← attempt + 1.
  Print: "Attempt attempt to pick up a dynamic block...".
  Move to dynamic_sweep_start.
  Move to dynamic_sweep_end.
  Wait for 6 seconds to allow blocks to stabilize.
  Close gripper to grasp a block.
  Retrieve gripper_state and calculate grip_distance.
  if grip_distance ≥ 0.025 then
    Print: "No block grasped. Adjusting...".
    Move to dynamic_sweep_adjust.
    Continue to the next iteration of the loop.
  end if
  Print: "Block grasped! Moving to drop-off location...".
  Move back to dynamic_home_position.
  Execute drop-off routine to place the block.
end while
Print: "Dynamic pickup process completed.".
=0
```

References

[1] Hutchinson Seth Vidyasagar M. Spong, Mark W. *Robot Modeling and Control (2nd Edition)*. John Wiley Sons, 2020.