MEAM520: Final Report

Yiming Huang, Yunshuang Li, Shaoting Peng, Yifan Wang

1 Problem Formulation

The objective of this project is to develop a strategy for successfully retrieving and stacking as many blocks as possible in a 1 vs. 1 competition scenario. Considering the competitive nature of the problem and that obstacles in the environment are mostly static, predictable, and/or detectable, we choose to prioritize speed over precision, dynamic over static. In the following sections, we will present a 6-DOF IK solver-based control strategy that is capable of alternating between precise detection-based and fast detection-free grabbing of blocks.

2 Methods

In this section, we will first introduce some basic yet fundamental control and detection methods that will be used throughout our project. Then we will walk through our implementations of static and dynamic block grasping methods in detail. After that, we will demonstrate our overall competition-oriented strategy for block grasping and placing based on these methods.

2.1 Basic Methods

2.1.1 Position Control: Inverse Kinematics

In this challenge, the overall environment within the robot workspace is relatively simple and stable, and there are no requirements for changing velocity or acceleration of the robot arm at any point. Therefore, we formulate a strategy that uses only position control for conducting the task of grasping and placing blocks. The strategy consists of executing movements to various position and orientation settings for the end-effector. To derive joint configurations for these corresponding settings, we utilize the 6-DOF of freedom Panda Arm IK Solver from Lab 2, where we assume that the joint angle of Joint 5 is fixed to zero. We chose to trade off one redundant degree of freedom of the robot arm (the z-axis of Joint 5 and Joint 7 are always colinear) for faster and more precise joint configuration calculations that would not encounter local minima issues.

The execution of robot movement based on joint configurations is achieved through the provided function `safe_move_to_position()`. Forward kinematics of the Panda Arm from Lab 1, though not used for executing the actual strategy, was mainly applied for debugging and validation of IK Solver calculations.

2.1.2 Gripper Control: Grasp and Release

The state of the gripper was controlled using the provided `exec_gripper_cmd` function which directs the gripper to close to a width of d meters, and apply a force of F newtons. For our strategy, the gripper was set to have two possible states:

- Grasping State: $d_{set} = 0.045$ m, $F_g = 50$ N
- Released State: $d_{open} = 0.08$ m, $F_r = 0$ N

where d_{set} is chosen to be slightly smaller than the width of the blocks and d_{open} is the maximum width of the open gripper.

2.1.3 April Tag Detection and Pre-processing

The robot arm is equipped with a camera near the end effector that is capable of detecting April Tags attached to the surfaces of the blocks. The detection data provides us with the block type (static or dynamic), block ID, and block pose in the camera frame, denoted as H^{cam}_{block}. Since the relative pose of the camera to the end effector is static, we are also given the constant homogeneous transformation matrix corresponding to the camera pose written in the end effector frame, H^{ee}_{cam}. Considering that the IK solver performs calculations based on an input orientation and position written in the robot base frame, we pre-process the detection results to obtain the block

pose written in the robot base frame, H^{base}_{block}:

$$H^{base}_{block} = H^{ee}_{base} H^{ee}_{cam} H^{cam}_{block}$$

where H^{e}_{base} represents a predetermined end effector pose at the state where the detection was conducted written in robot base frame.

2.2 Static Block Grasp

There are two key aspects to successfully grasping the static blocks:

1. Accurately determining the target position and orientation for grasping the static blocks from the detection results.
2. Designing a collision free trajectory for retrieving the static blocks.

In the following sections, we will detail our strategy choices for each of the two aspects above. In particular, we encountered notable issues in the first aspect of accurately determining grasping orientation and positions. Hence, we will elaborate on both the initial failed strategy and the our final successful strategy below.

2.2.1 Grasping from Above

In our approach, we choose to let the end effector grasp the static blocks from above, a choice that requires smaller degrees of rotation overall in the joints of the robot arm and has a relatively lower likelihood for collision.

In order to determine the target top-down grasping orientations and positions, we first move the end effector to a fixed position roughly above the center of the static block table at a height where all four static blocks are within the robot's field of vision, with the positive direction of the z-axis of the end effector parallel to the negative direction of the z-axis of the world frame (see Figure 1). We denote the end effector orientation matrix and position in the robot base frame at this point to be R_{static} and t_{static}, respectively:

$$R_{static} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{bmatrix}$$

$$t_{static} = \begin{bmatrix} 0.542 & \pm 0.169 & 0.500 \end{bmatrix}$$

At this position, the robot arm performs one single round of detections, ideally detecting all four of the static blocks. From the detection results, we complete the calculation of the target orientations and positions for grasping each of the static blocks and then command the robot arm to sequentially execute the retrieval trajectories respectively. In particular, our approach requires the robot to grasp the static blocks from above, implying that the z-axis of the target end-effector frame will be in the negative direction of the world z-axis.

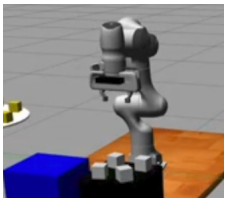
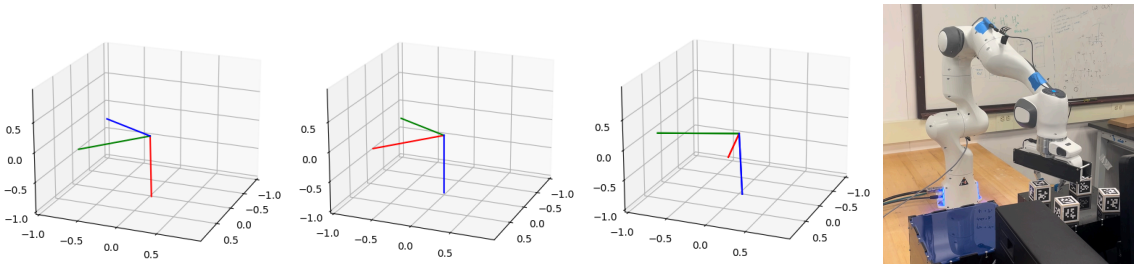


Figure 1: Static Detection Position

The detected position of the static block, written in the world frame, can be directly used as the target position of the end effector. On the other hand, considering that the z-axis of the detected static block frame, written in the world frame, is not necessarily in the negative direction z-axis of the world frame, we cannot directly use the detected block orientation as the target grasping orientation of the end effector. However, since the blocks are placed on a flat surface that is parallel to the world x-y plane, it is guaranteed that one of the axes of the block frame is parallel to the world z-axis. Hence, we can determine the target orientation of the end effector by aligning the x, y axes of the target end effector frame to the remaining two axes in the block frame that are not parallel to the world z-axis (while maintaining the right-hand rule).

In our initial failed strategy, the axis parallel to the world z-axis was determined to be the axis with the smallest Euclidean distance to `np.array([0,0,1])` or `np.array([0,0,-1])`. The remaining two axes were directly used to define the x, y axes for the target frame according to the right-hand rule. Using the IK solver on the calculated target position and target orientation, we are able to obtain an



(a) Detected Block (b) Theoretical Target (c) IK Calculated Target (d) Hardware Failure

Figure 2: Inconsistencies in Initial Orientation Frame Calculations

executable grasping configuration on the robot arm. There were no issues in simulation for this approach, but on hardware we found that the configurations produced by the IK solver were inconsistent with the theoretical target configuration, causing the static block grasp to fail.

We recorded the detection data on hardware and analyzed the original data as well as the calculated target orientation. We found that the detection error on hardware was a lot larger than that in simulation, causing the axes of the detected block frames to not be strictly orthogonal to one another. This created potential issues in the IK solver, causing the produced orientation and position to be inaccurate. In Figure 2, we visualize, for a sample hardware detection result, the inconsistencies between the orientation of the detected block frame, the theoretical target frame, and the IK - calculated target frame, with origins all set at the origin of the world frame (x,y,z axes marked in red, green, and blue, respectively).

Building upon the observations above, we decided that instead of directly defining the x, y axes of the target frame with detection data, which could cause the target frame axes to not be orthogonal, we could calculate the target frame by applying a rotation along the world z-axis to a default valid coordinate frame (here we select the end effector frame at the static detection position). Denote the orientation of the end effector frame at the static detection position in the world frame as R_{static}, the orientation of the target frame in the world frame as R_{target}, and the rotation matrix as $R_{z,\alpha}$ where α is the rotation angle along the world z-axis, then we have:

$$R_{target} = R_{z,\alpha} \cdot R_{static}$$

$$R_{z,\alpha} = \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

For efficient correction of the orientation, we define the rotation angle α as the smallest angle between the x-axis of the end effector frame and the two axes of the detected block frame that lie in the world $x-y$ plane. Denote the unit vector representing the x-axis of the end effector frame in world frame as u and the set of unit vectors representing the axes of the detected block frame that lie in the world $x-y$ plane as S, then we have:

$$\alpha = \min_{v \in S}(\arctan \frac{u_x v_y - u_y v_x}{u_x v_x + u_y v_y})$$

This approach was executed successfully both in simulation and on hardware, hence selected as our final strategy for calculating the target grasping position and orientation of static blocks.

2.2.2 Trajectory Planning with Intermediate Stop Points

The position control strategy that we are using directs the end effector smoothly from point to point in the robot workspace. However, the trajectory between a target grasping point and the robot starting point planned and executed through `safe_move_to_position` does not take into account any external obstacle avoidance. The main collision issues during the static block grasping and placing sequence that were identified during trial

and error simulations that used no intermediate stop points include:

- Issue 1: Collision between the end effector and the target block during grasping due to the end effector simultaneously reaching down and correcting its orientation.
- Issue 2: Collision between a grasped target block and the block tower during block placing.
- Issue 3: Collision between the end effector and the block tower when the end effector moves to grasp the next static block.

Based on these observations, we determined that at least three intermediate stop points are necessary, one between the starting point and the target grasping point (Issue 1), one between the target grasping point and the target placing point (Issue 2), and one between the previous target placing point and next starting point (Issue 3). Denote the calculated target grasping position and orientation matrix for ith static block in the robot base frame as $R^{(i)}_{sgrasp}$ and $t^{(i)}_{sgrasp}$, denote the target placement position and orientation matrix for static block i in the robot base frame as $R^{(i)}_{splace}$ and $t^{(i)}_{splace}$. The static block grasping and placing sequence with intermediate stop points can be represented as follows:

1. Start: R_{static}, t_{static}
2. Intermediate Stop 1: Point above the target block where the end effector orientation matches the target grasping orientation.
   $R^{(i)}_{sgrasp}, t^{(i)}_{sgrasp} + \begin{bmatrix}0 & 0 & 0.03 + 0.05 * i\end{bmatrix}$
3. Target grasping point: $R^{(i)}_{sgrasp}, t^{(i)}_{sgrasp}$
4. Intermediate Stop 2: R_{static}, t_{static}
5. Target placing point: $R^{(i)}_{splace}, t^{(i)}_{splace}$
6. Intermediate Stop 3: Stop point above the block placing point.
   $R^{(i)}_{splace}, t^{(i)}_{splace} + \begin{bmatrix}0 & 0 & 0.05\end{bmatrix}$

This sequence is repeated back to back for each static block. A visualization of this sequence on hardware will be presented later in the evaluation section.

2.3 Dynamic Block Grasp

2.3.1 Grasp without detection

Even though detection is an advanced method to grasp dynamic blocks, and in theory should be more accurate and smart, we found that there are still limitations for stable detecting and grasping due to inevitable noise in real camera sensor and the difficulty of estimating time intervals between detection and execution. Hence, we decided to develop a detection-free dynamic block grasping strategy where the robot arm is directed to predetermined key positions to wait for blocks upon the turntable. As the robot waits, the gripper opens and closes to fish for passing blocks. A gripper state analysis is conducted repeatedly to identify when a block has been grasped and trigger the robot arm to move the grasped block to a predetermined position on the block tower. The whole workflow of this method is demonstrated in Figure 3 and described below for the ith dynamic block:

1. Preparation state: Right at the side of the turntable
   $$R_{prep} = \begin{bmatrix}\frac{\sqrt{2}}{2} & 0 & \frac{\sqrt{2}}{2} \\ 0 & -1 & 0 \\ \frac{\sqrt{2}}{2} & 0 & -\frac{\sqrt{2}}{2}\end{bmatrix}$$
   $t_{prep} = \begin{bmatrix}0 & \pm 0.75 & 0.4\end{bmatrix}$
2. <Transition 1> Move along the y-axis to get closer to the center of the turntable while maintaining the same orientation.
   $R_{t1} = R_{prep}, t_{t1} = \begin{bmatrix}0 & \pm 0.9 & 0.4\end{bmatrix}$
3. Grasp state: Wait at the position above, continuously opening and closing the gripper until we detect that an object has been grasped.
4. <Transition 2>: Move grasped block back to the preparation state at the side of the turntable to ensure collision-free movement.
   $R_{t2} = R_{prep}, t_{t2} = t_{prep}$
5. Final state: Move grasped block to predetermined target placement point defined by $R^{(i)}_{dplace}, t^{(i)}_{dplace}$

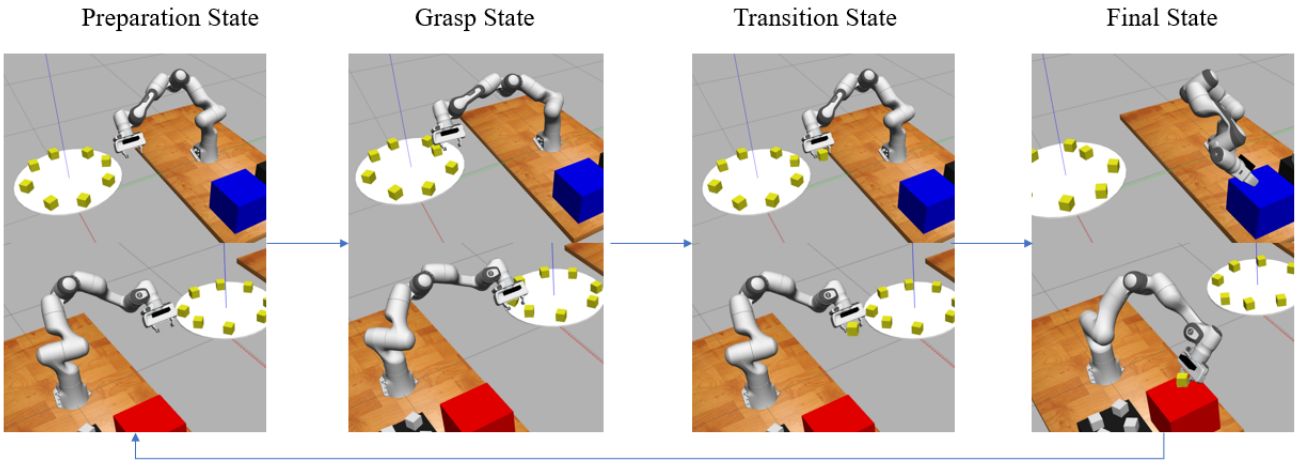


Figure 3: The three different states shown sequentially in simulation.

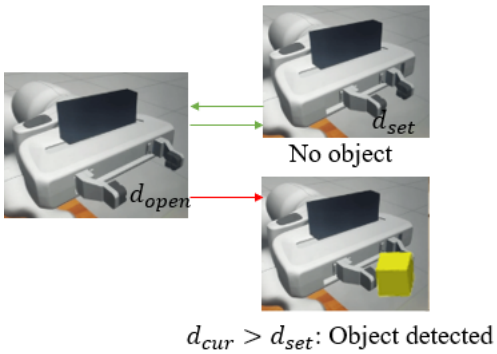


Figure 4: Gripper State Analysis

The key for successfully executing this workflow is a robust gripper state analysis algorithm. We've set the gripper to loop between released state and grasping state with a random time delay after reaching the released state in each iteration determined with `htime.sleep(1.7+np.random.randn()/3)`. As shown in Figure 4, at the grasping state of each iteration we perform a gripper state analysis by comparing the actual gripper width d_{cur} obtained through the function `get_gripper_state` with the theoretical width of the gripper at its grasping state, d_{set}. Since the theoretical width of the gripper is slightly smaller than the width of the blocks, if $d_{cur} \leq d_{set}$ then there can't be a stably grasped block between the gripper fingers. On the other hand if $d_{cur} > d_{set}$, we can deduce that the gripper was unable to close to its set width, implying that an object has been successfully grasped and that we can transition to the final state. Note that it is also possible to perform the gripper state analysis based on the gripper force. However, during testing in simulation we found that the force was always 0 regardless of the settings applied through `exec_gripper_cmd`. Hence, for consistency between simulation and hardware we chose to only evaluate the gripper width.

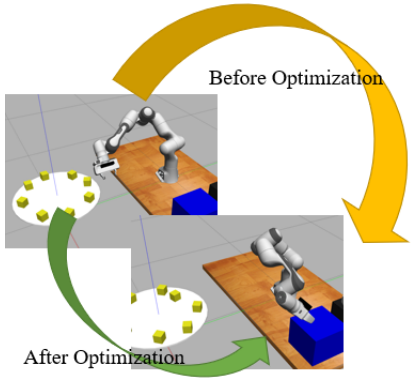


Figure 5: Trick 1–The movement between preparation state and final state before and after using least l_2 norm selection.

In order to improve the adaptability of the detection-free strategy to changes in the environment, we implemented two additional tricks. One trick we used is to pick the IK calculated target configuration closest to current target configuration in terms of Euclidean distance. Usually there are many inverse kinematic solutions for one position, which will lead to different paths between positions, some more optimal than others. Since time is quite limited in the challenge,

we need to pick the shortest path for execution. As shown in Figure 5, without this optimization the default first IK solution will lead the robot arm to rotate clockwise instead of counterclockwise, which will cost much more time and bring more risks of dropping the block accidentally.

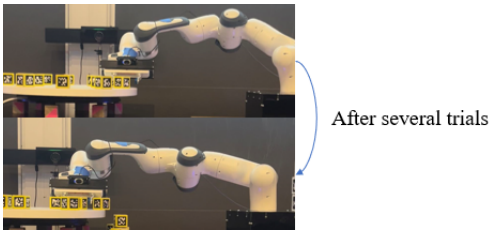


Figure 6: Trick 2–Change preparation state according to the changing environment.

The other trick is that we adapt the waiting position of the gripper on the turntable according to the changing environment (see Figure 6). Our method pushes some of the dynamic blocks towards the center of the turntable and we expect our opponents to cause similar changes. Hence, we keep track of the number of failed grasping trials at the initial waiting state, and direct the robot arm to move parallel to the y-axis closer to the center of the turntable once the number of failed grasping trials exceeds 5. This allows us to avoid wasting time waiting at the outer rim of the turntable when there is a lack of blocks and reach for blocks that have been moved closer to the center.

2.3.2 Grasp with detection

In this method, we use camera to detect dynamic blocks' positions, and move the end-effector downward to grasp the detected block. This method can achieve good performance on simulation while with certain drawbacks, which will be discussed in 'Analysis' section. Here we mainly focus on the introduction to methodology. The whole process is illustrated in Figure 7 on the next page.

1. Move the end-effector above the turntable's edge.

$$R_{prep} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{bmatrix}, t_{prep} = \begin{bmatrix} 0 & \pm 0.75 & 0.4 \end{bmatrix}$$

2. Keep detecting AprilTags using camera. Assume that we project the detected block and end-effector onto the turntable plane, projected block center is point B, projected end-effector center is point E and the turntable's center is point O as illustrated in Figure 8. Then, calculate the angle $\angle BOE$ using `np.atan2()`.

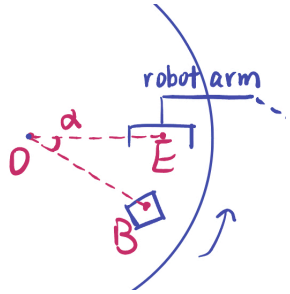


Figure 8: Top view of end effector and block to form ∠BOE

3. When the angle is between 0.17 and 0.19, we will move the end-effector to $[0, B[1] +$ `offset`$, 0.233]$ and execute grasping. Unlike the detection-free method, here we only grasp once and then lift the end-effector, regardless of whether the block is grasped. $B[1]$ is the y-coordinate of the detected block, and `offset` (0.012 for blue and -0.012 for red) is an approximation for the block's shift in y-axis during the movement of the robot arm, which is selected by conducting many experiments in simulation and hardware.

4. Move the end-effector on the top of the predefined dynamic block stack and release the gripper, where i is the current number of dynamic blocks. Finally lift the end-effector and move to the start position for next static or dynamic block according to our strategy.

2.4 Strategy

2.4.1 Grasping Sequence

There are two kinds of blocks in this project: the white static blocks placed on a stationary platform and the yellow dynamic blocks placed on a rotating turntable. We determined our grasping sequence based on the following considerations:

1. Each dynamic block is worth 20 points and static block is worth 10 points, while dynamic

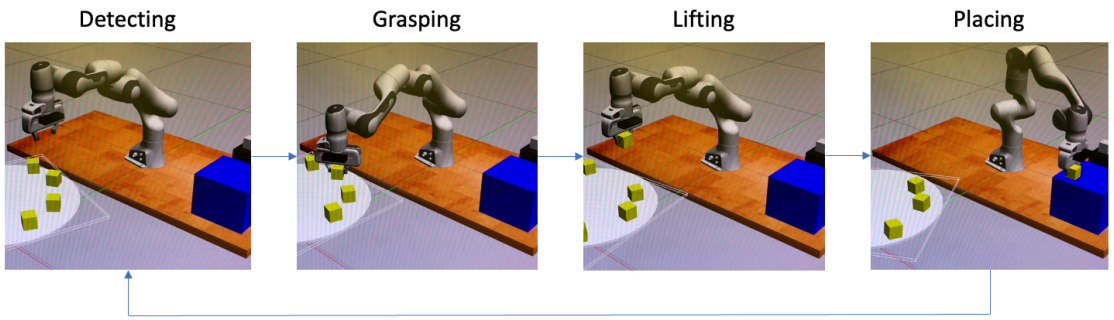


Figure 7: The whole process for dynamic grasping with detection

blocks can be more difficult to get than static blocks.

2. Our opponent may take destructive actions on the dynamic blocks, making grasping dynamic blocks even more challenging.

3. We noticed that many teams are able to grasp static blocks successfully, so the key factor for success in competition is the number of dynamic blocks.

We have tested on hardware that grasping all the four static blocks can take 1 minute 50 seconds, and grasping one dynamic block can take nearly 30 seconds, so our strategy is to grasp two dynamic blocks first earlier than our opponent when the environment is still relatively stable, then moving to the static blocks, which should take almost 3 minutes in total. If we advance to knock-out, we will maintain our strategy for the first 3 minutes and try to grasp as many dynamic blocks as possible in the additional 2 minutes.

2.4.2 Dynamic grasping method

We developed two dynamic grasping methods: with detection and without detection. For the competition, we chose the method without detection because of the following reasons:

1. Early dynamic grasping: Based on our grasping sequence, we are less likely to suffer from the potential destructive actions by our opponent, so the position of dynamic blocks are more predictable and thus easy to grasp without perception.

2. Collision avoidance: We are aware that if the end-effector hits the block from top, it will be considered as a collision and the robot arm will be halted. There is a higher risk for this issue with the method using detection due to camera detection noise.

3. Better performance on hardware: Although the method with detection works better in simulation, the other method works better on hardware, due to the differences between simulation and hardware which will be discussed more in the Analysis section. The numerical comparison results and visualization can be found in the Evaluation section.

2.4.3 Block Placing

Considering the nondeterministic nature of the grasped pose for dynamic blocks in our strategy, we chose a conservative placing strategy: we stack the dynamic blocks on the top left corner and the static blocks on the bottom right corner. In this way, we can avoid toppling issues when the stack is too high and collision issues when the two stacks are too close. The placement positions and orientations executed by the robot arm are predetermined for the ith static and jth dynamic block as follows:

$$R^{(i)}_{splace} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{bmatrix}$$

$$t^{(i)}_{splace} = \begin{bmatrix} 0.6245 & \pm 0.1065 & 0.255 + i * 0.05 \end{bmatrix}$$

$$R^{(j)}_{dplace} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{bmatrix}$$

$$t^{(j)}_{dplace} = \begin{bmatrix} 0.4995 & \pm 0.2315 & 0.255 + j * 0.05 \end{bmatrix}$$

3 Evaluation

3.1 Simulation Results

3.1.1 Static blocks

The grasping of static blocks achieved successful performance in simulation with the optimized strategy, resulting in a stable four-story static block tower for all trials. As demonstrated in section 2.2.1., the chosen optimized static strategy is more robust to detection noises that were more prominent on hardware than in simulation.

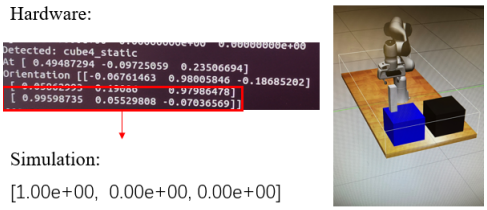


Figure 9: Orientation evaluation for static blocks.

3.1.2 Dynamic blocks using detection

In simulation, we tested this method several rounds using different threshold settings and grasping angles:

1. Angle vs. distance as threshold: As introduced before, we set a threshold for the detected block's position (i.e. if the distance between block and end-effector is smaller than a threshold, we will move to grasp), and we can measure this threshold either using $\angle BOE$ (refer to section 2.3.2) or distance between their y-coordinates.

2. Direct downward vs. 45° inclined grasping: In addition to the direct downward grasping, we also tried to grasp with the end-effector 45° inclined to the turntable which could help avoid potential collisions.

We tried to grasp 4 dynamic blocks each round, and performed 10 rounds in total. The numbers of blocks successfully grasped are listed in Table 1 for four different settings. Simulation results indicate that there were no potential safety issues for this strategy and that an angle-based threshold with a downward grasp has the best performance overall. However, on hardware this method rarely succeeds for any of the simulation-tested settings, and thus was not used in the competition. The potential reasons are explained in the 'Analysis' section and we only present hardware results for the detection-free method in the following evaluation sections.

Method	success/total (success rate)
Angle + Downward grasp	34/40 (85.0%)
Angle + 45° grasp	32/40 (80.0%)
Dist + Downward grasp	27/40 (67.5%)
Dist + 45° grasp	26/40 (65.0%)

Table 1: Different dynamic grasping with detection method accuracy

3.1.3 Dynamic blocks without detection

In contrast to the detection-based method, although the detection-free method worked well on hardware, we were unable to obtain successful results in simulation. As shown in Figure 10, the friction between the blocks, the table and the gripper was set to be quite large in the simulation, which caused the turntable to get stuck when the robot arm comes into contact with blocks as it attempts to slide to the waiting position on the turntable as well as when it attempts to grasp the moving blocks from the side. However, after many trials in simulation, we concluded that despite the grasping failures the movement of the robot was safe and stable over all, indicating that it was plausible to test the strategy on hardware.

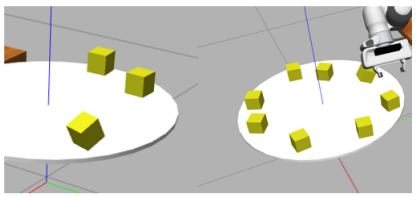


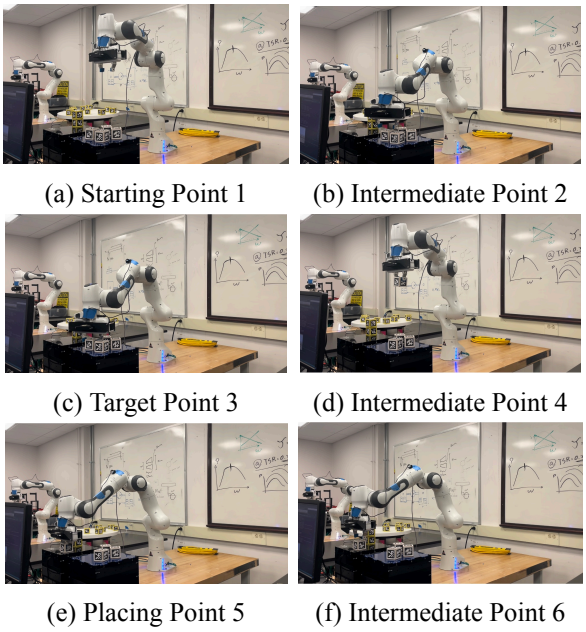
Figure 10: Great frictions shown in the simulation.

3.2 Hardware Results

3.2.1 Static Block Grasp

In order to better demonstrate the whole process of the static grasp, we intercepted a few key frames in

the video recording of successful hardware results, each representing a stage in the defined static grasping sequence. As shown in Figure 11, the robotic arm first moves to point 1, which is directly above the static blocks platform, where the robotic arm detects the position of static blocks via the camera. According to the detection results, the robotic arm descends to a height close to the block and adjust the orientation to be parallel to the side of the block at point 2, and then continues to slowly descend to point 3 and complete the grab. After grabbing, the robotic arm moves up to point 4 and then to placement point 5. After placing, the robot arm moves up a certain distance to point 6 to avoid hitting the placed block.



(a) Starting Point 1 (b) Intermediate Point 2

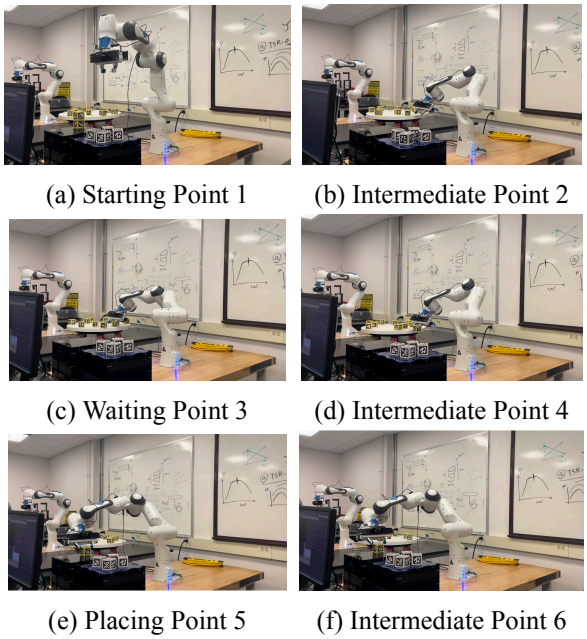
(c) Target Point 3 (d) Intermediate Point 4

(e) Placing Point 5 (f) Intermediate Point 6

Figure 11: Static Grasping Trajectory

3.2.2 Dynamic Block Grasp

In order to better demonstrate the whole process of dynamic grasp, we also intercepted a few key frames in the video recording of successful hardware results, corresponding to each stage in the detection-free dynamic grasp sequence. As shown in Figure 12, the robotic arm first moves from the starting point 1 to an intermediate point 2 outside the turntable, then translates to the waiting point 3 above the turntable, where the robotic arm waits to grab blocks. After detecting a successful grasp, the robotic arm translates the block out of the turntable to intermediate point 4, and then moves smoothly to point 5 above the platform for placement. After placing, the robot arm moves up a certain distance to point 6 to avoid hitting the placed block.



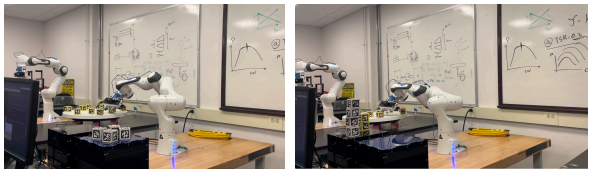
(a) Starting Point 1 (b) Intermediate Point 2

(c) Waiting Point 3 (d) Intermediate Point 4

(e) Placing Point 5 (f) Intermediate Point 6

Figure 12: Dynamic Grasping Trajectory

Two positions of the grasping In the competition setting, there would exist two areas of dynamic block placements on the turntable: low-difficulty outer circle and high-difficulty inner circle. Shown in Figure 13, to realize the grasping of both areas' blocks, we choose two settings for the waiting point 3 and adapt between the two according to the success rate of our grasps (detailed in section 2.3.1.)



(a) Outer Blocks Grasp (b) Inner Blocks Grasp

Figure 13: Different settings of Waiting Point 3

3.2.3 Competition

Based on the hardware results, our competition strategy is to first, before the opponent disturbs the arrangement of the dynamic blocks, grab two dynamic

blocks on the outer circle of the turntable and build them into a pile. Then, grab the static blocks and build another stack of four static blocks. Finally, grab the dynamic blocks in the inner circle and build on the previously dynamic blocks in the remaining time. Unfortunately, we didn't achieve successful performance for the competition, only stacking all the static blocks at our best. The following are issues that we encountered in the competition:

1. Gripper state analysis failed: As shown in Figure 14, even though the gripper successfully grasped a dynamic block, it didn't move to the final state, which means that the gripper state analysis was not functioning accurately. In retrospect, we found that the threshold we had chosen for determining a successful grasp $d_{cur} > d_{set}$ was not sufficient. In simulation, there was no force applied regardless of the setting, so the gripper would attempt to move precisely to the set grasping width d_{set} and we were able to threshold using it. On hardware, the actual force was applied causing the gripper to attempt to close completely, which in turn produced smaller values for d_{cur} then anticipated. In order for the gripper state analysis to work properly, we later reduced the threshold to $d_{cur} > d'$ where $d' = 0.25$ m. Hardware results were successful after this adjustment and are presented below.

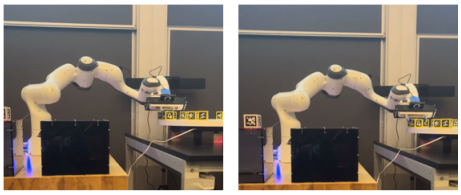


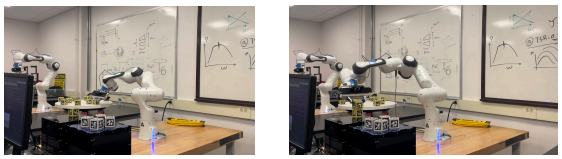
Figure 14: Gripper opening and closing without moving to the next state

2. Only 3 static blocks are detected for blue team: As shown in Figure 15, the blue team robot arm only detected 3 static blocks. We assume that this could be due to a coincidental lighting condition or reflection of the blocks on the platform. Another reason could be that the field of view of the camera is not sufficient. In retrospect, we adjusted the position for detecting static blocks to be slightly higher for a larger field of view. From this issue we could tell that the setting for detecting static blocks may not work robustly enough for varying environments (lab vs. competition).

Figure 15: 3 static blocks for blue (left), 4 static blocks for red (right)

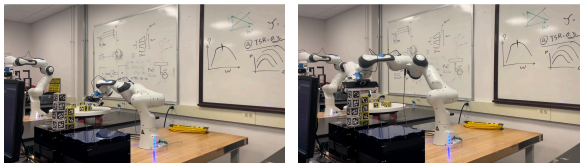
To validate our optimizations based on the issues found in the competition, we ran additional trials in the robot lab using the adjusted code and were able to achieve a steady and successful grasping and placing sequence for both static and dynamic blocks (shown in Figures 16-18).

(a) Dynamic Block 1 Grasp (b) Dynamic Block 1 Place



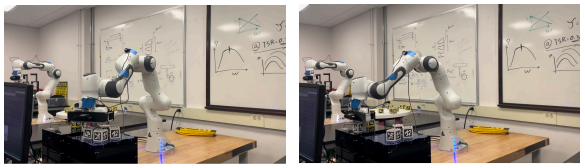
(c) Dynamic Block 2 Grasp (d) Dynamic Block 2 Place

Figure 16: Dynamic block grasping of two blocks prior to static block grasping

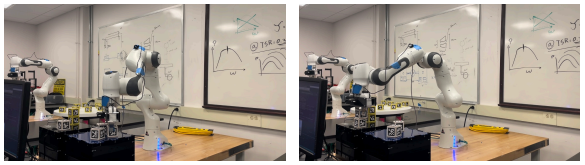


(a) Dynamic Block 3 Grasp (b) Dynamic Block 3 Place

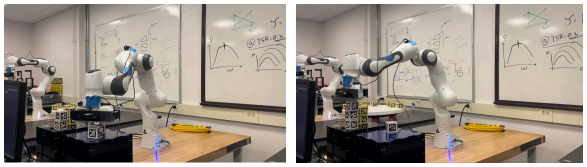
Figure 17: Dynamic block grasping after static block grasping



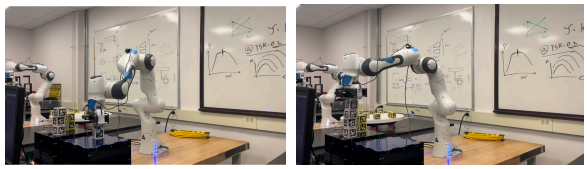
(a) Grasp of Static Block 1 (b) Place of Static Block 1



(c) Grasp of Static Block 2 (d) Place of Static Block 2



(e) Grasp of Static Block 3 (f) Place of Static Block 3



(g) Grasp of Static Block 4 (h) Place of Static Block 4

Figure 18: Static block grasping

4 Analysis

4.1 Simulation & Hardware Differences

During our tests in both simulation and hardware, we discovered many different behaviors:

1. **Grasp force and position**: Recall that in the competition we were able to achieve opening and closing gripper in the desired position, but failed to detect successful grasps. This was mainly caused by the inconsistency between applied forces in simulation and on hardware, where simulated forces are always zero. Large non-zero force settings on hardware caused the gripper to forcibly close further than expected even in the case where a block is present, leading to failed gripper state analysis.

2. **Camera Noise**: In simulation, the camera is ideal and produces relatively accurate detection, whilst on hardware, there is considerable noise in the detections. In particular, the detected block frames are not completely orthogonal, which caused inconsistent IK results for our initial static strategy. After adjusting the static strategy to not directly use the noisy detected block frames as input for our IK solver, we were able to solve this issue. However, the noise still causes issues such as collision for the detection-based dynamic grasping method which requires high precision. So we chose to implement and use a detection-free method for dynamic grasping in the end.

3. **Camera Field of View**: The field of view of the camera between simulation and hardware are also different. With the theoretical same camera position and orientation, the camera on hardware appears to have a smaller perception field than that in simulation, which means some static blocks can't be detected on hardware even if they can be detected in simulation. To solve this, we lifted the original static detection position to detect from a higher position.

4. **Dynamic Block Kinematics**: In simulation, our detection-free dynamic grasping method never succeeded, but it had a high success rate on hardware. This was because the unrealistically large friction settings in the simulation made it impossible to push the blocks on the turntable when the robot arm attempted to move to a waiting position or grasp the blocks from the side. We discovered that this issue was not present on hardware, and thus were motivated to use the detection-free dynamic grasping method.

4.2 Potential Improvements

1. **Combine the two dynamic methods**: We can detect the target dynamic block's position in advance, and translate horizontally to a waiting position on the the turntable from the outside to avoid potential collision based on careful calculation of target block's trajectory as well as extensive tests on simulation and hardware. Ideally, we can save more time than waiting blindly for blocks to come into the end-effector and have a more reliable result.

2. **Be more concise with the y-offset in dynamic grasping**: Recall that in the dynamic grasping method introduced in section 2.3, when the camera detects the block, the end-effector will move to $[0, B[1] + \texttt{offset}, 0.233]$, where $B[1]$ is the y-coordinate of the detected block, and `offset` is set to 0.012 for blue and -0.012 for red. Ideally we want to compensate for the block's shift in y direction because of the turntable's movement, but ± 0.012 is just an empirical number through simulation and hardware experiments and it's set to a static number. In fact, with different distance to the turntable's center, this y-offset should also be different, as illustrated by the following figure:

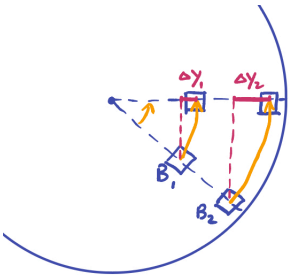


Figure 19: Top view of two blocks moving on the turntable

As we can see, the further the block is to the turntable's center, the shift in y direction is larger ($\Delta y_1 > \Delta y_2$). To solve this problem, we may make this y-offset a function of block's y-coordinate. For example, we can change the original $B[1] + \texttt{offset}$ into $1.05 \times B[1]$, which can be more closer to our target position.

3. **Combine the two stacks**: Currently we stack the dynamic and static blocks separately. One better improvement is to place them on the same stack. But this method requires our algorithm to be more accurate on grasping and placing the blocks, for every small placing error may accumulate and result in the collapse of tower.

4. **Better trajectory planning**: Based on our measurements, the duration of the robot arm's movement is not linear to its moving distance, even a single lift in 0.05 meters can take more than 3 seconds. Hence, more experimentation on selecting a more compact and optimal set of intermediate points could improve the efficiency of the executed trajectories. In addition, using a potential field planner instead of an IK solver, though less robust in terms of escaping local minima, may also generate more efficient trajectories.

5 Lessons Learned

1. **Be wary of changing code that has been well-tested on hardware.** One of the main reasons for our failure in the competition is that our method used for detection-free dynamic block grasping couldn't be tested well in simulation. However, after the last lab, during which we coincidentally found parameters that worked, we chose to change the criterion based on our experience and theoretical intuitions in our final adjustments, which ended up not working.

2. **Observe and optimize based on realistic failed data.** The idealized simulation environment made it difficult to identify and debug key flaws in our initial theoretical approaches that were not robust enough to deal with various inconsistencies between hardware and software. Through observing and recording data of failures gathered during lab sessions and the competition, we were able to find and fix small nuances in our code that were causing huge issues.

3. **Teamwork.** A full strategy can be difficult to devise and even more difficult to realize from scratch. However, through brainstorming and pair-programming all together as a team, we were able to produce multiple optimized iterations of various different approaches. In addition, by dividing and conquering tasks between team members, we were also able to complete milestones with higher efficiency. The combination of our ideas and efforts ultimately helped us push the project forward steadily.

Acknowledgements We thank Professor Cynthia Sung and all the TAs for helpful discussions and the wonderful semester.