Pick-and-Place Challenge

Yanqian Wang, Jinyuan Zhang, Angelina Zhang, Lintao Zheng
{yqwang12, jinyuanz, angelzxy, zer1tle}@seas.upenn.edu

I. Scope

This project addresses the MEAM 5200 Pick-and-Place Challenge, in which a Franka Emika Panda robot manipulator autonomously acquire blocks from a shared environment and stack them on a designated goal platform to maximize the team's score. The competition involves two teams (Red and Blue) operating simultaneously in a shared workspace, with blocks categorized into two types: static blocks positioned on a stationary platform and dynamic blocks placed on a rotating turntable at the world frame origin.

The detailed experimental setup is shown in Fig 1.

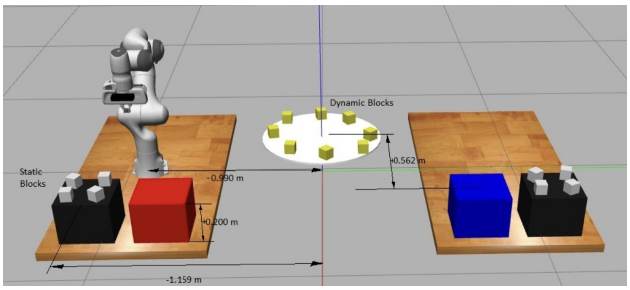


Fig. 1: Experimental environment in Gazebo simulation.

The primary objective is to develop a robust and reliable robotic manipulation system capable of:

1) Detecting and localizing blocks using AprilTag-based fiducials with an end-effector-mounted camera
2) Computing feasible grasp poses for both stationary and moving objects
3) Executing pick-and-place operations to stack blocks on the goal platform
4) Maximizing the cumulative score based on the formula: Points = Value × Altitude, where static blocks have a value of 10 and dynamic blocks have a value of 20

Our team's strategy prioritizes rapidly acquiring and stacking four static blocks first, leveraging our highly stable grasping pipeline to establish a reliable foundation. Following the static block phase, the system transitions to collecting as many dynamic blocks as possible from the rotating turntable, stacking them atop the static blocks to maximize altitude-based scoring. We are honored to have achieved first place in the final competition.

This report first presents the methods employed for perception, grasp planning, and motion control of the Panda manipulator. We then evaluate the system performance through both Gazebo simulation and hardware testing. Subsequently, we analyze the advantages and limitations of our approach, along with potential improvements for future work. Finally, we discuss the lessons learned from this project.

II. Method

A. Initialization

Our pick-and-place system consists of four main components: **block detection**, **gripper alignment with block**, **static block pick and place**, and **dynamic block pick and place**. We leverage the provided `ArmController` for low-level Panda arm control and the `ObjectDetector` for AprilTag-based

block detection. In addition, forward kinematics (FK) and inverse kinematics (IK) modules implemented in previous labs are initialized for pose transformation and motion planning. For inverse kinematics, we employ the **pseudo-inverse** method, which provides fast and reliable solutions and is well suited for the real-time pick-and-place task. A self-defined `GraspPrimitive` class encapsulates high-level grasping and placing behaviors, including pose calculation, grasp execution, placement strategies, and gripper status estimation. Before executing the pipeline, each team (red or blue) is assigned a predefined start pose in the robot base frame. This start pose is chosen to ensure that the end-effector-mounted camera provides a clear field of view covering all four static blocks on the initial platform. Since all blocks are placed on a known, flat tabletop, the camera initialization is task-driven and does not require precise depth alignment. Furthermore, distinct and adjustable camera calibration offsets are defined for each team to compensate for systematic perception bias in the camera frame.

B. Block Detection

Prior to AprilTag-based block detection, the teaching staff performed camera calibration to correct for lens distortion, as illustrated in Fig. 2, and to reduce systematic geometric errors in the image formation process. While this calibration step significantly improves detection accuracy, we observed that this procedure introduces a small but consistent offset in the estimated block poses. As a result, additional offset compensation is required in practice to ensure reliable block localization during manipulation.

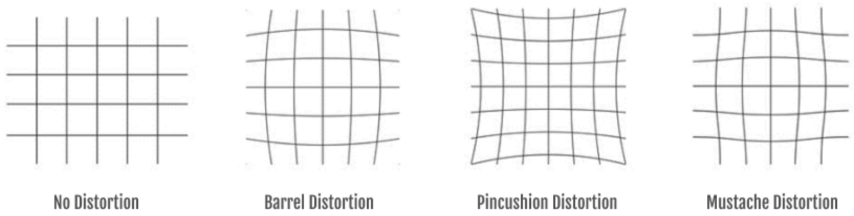


Fig. 2: Camera Distortion [1]

Block detection is performed using AprilTag pose estimation provided by the `ObjectDetector`. AprilTags are planar fiducial markers with known geometry, allowing the 6-DoF pose of a tag to be recovered by detecting its corner features in the image and solving a Perspective-n-Point (PnP) problem. In this project, all six faces of the cubic block are covered with AprilTags, which are used for obtain the pose of each detected block expressed in the camera coordinate frame and block category.

Each detected block pose is initially represented as a homogeneous transformation $\mathbf{T}_{cube}^{camera}$, which denotes the pose of the detected block expressed in the camera coordinate frame. To express the block pose in the robot base frame, a chain of rigid-body transformations is applied. Let $\mathbf{T}_{camera}^{ee}$ denote the transformation from the camera frame to the end-effector frame, and let $\mathbf{T}_{ee}^{base}$ denote the transformation from the end-effector frame to the robot base frame, obtained via forward kinematics. The block pose in the base frame is computed as:

$$\mathbf{T}_{cube}^{base} = \mathbf{T}_{ee}^{base}\mathbf{T}_{camera}^{ee}\mathbf{T}_{cube}^{camera}, \tag{1}$$

Despite the initial camera calibration, noticeable offsets were observed in the estimated block positions along all three axes. To compensate for residual biases in the image plane, translational offsets in the x and y directions are applied in the camera frame before transforming the detected poses to the robot base frame. These offsets are kept fixed within a single task execution, as the calibration bias remains consistent throughout a single run. For the z-axis, depth estimates from the camera were found to be significantly noisier, even with filtering applied. Since all static blocks lie on a planar tabletop with a known height, the z-coordinate of each static block pose in the base frame is therefore clamped to a

fixed value of $0.235\,\mathrm{m}$, corresponding to the tabletop height. This design choice improves robustness and avoids the need for repeated manual tuning of depth offsets.

Detected blocks are categorized into static and dynamic blocks based on their AprilTag labels. For static blocks only, distance-based sanity checks are applied to discard detections that are unrealistically far from the camera or the robot base. This prevents the system from mistakenly selecting blocks that have fallen onto the ground and are no longer on the tabletop. In addition, small blocks detected within the target platform region are explicitly filtered out during the static block selection stage. This avoids unintended re-grasping of blocks that are already placed on the target platform but may still be visible from the start position.

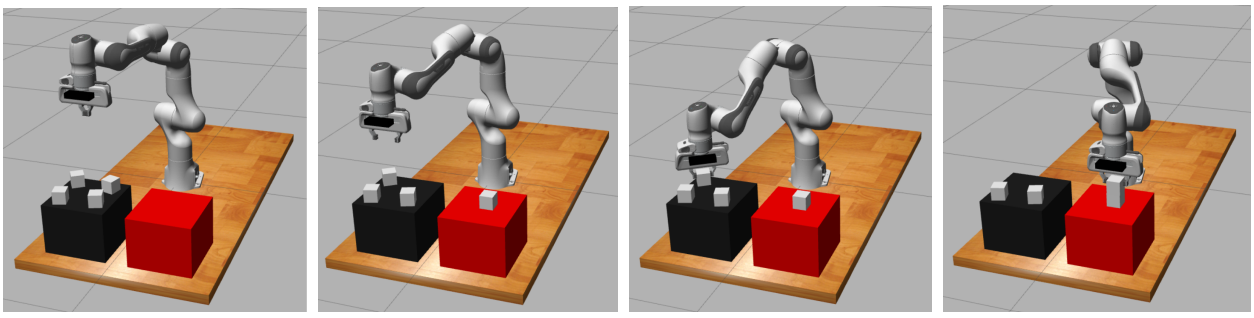
C. Gripper Alignment with block

After detecting the position and orientation of a block, it is not sufficient to directly move the gripper to the detected pose. To ensure reliable grasping and stable placement, the gripper must be aligned such that its grasping direction is parallel to one of the block's edges. Improper alignment may result in unsuccessful grasps or unstable placements, where a block is placed at an incorrect orientation and subsequently tips over due to a raised center of mass.

To achieve robust alignment, we explicitly analyze the detected orientation of each block. The block orientation is represented as a rotation matrix, from which the three orthogonal axis vectors of the block frame are extracted. For each axis vector, we examine its third component, which corresponds to its alignment with the global vertical (z) direction. The axis whose z-component has the smallest magnitude is selected as the most horizontal axis of the block. This axis is therefore chosen as the target alignment direction for the gripper. Once the most horizontal axis is identified, its orientation in the horizontal plane is calculated by taking the arctangent of its projected components onto the xy plane. Based on this angle, a desired gripper orientation is constructed by forming the corresponding rotation matrix, which is then used as the 3×3 rotational component of the homogeneous transformation for grasp execution.

This strategy is necessary because blocks are not always placed in an ideal orientation. Due to variations in placement and sensing noise, the axes detected via AprilTag pose estimation may not align consistently with the expected block frame, for example, the detected x-axis may point partially upward, and the detected z-axis may not be perfectly aligned with the global vertical direction. By selecting the most horizontal axis rather than relying on a fixed tag axis, the proposed method provides a more robust and reliable gripper alignment across a wide range of block orientations.

D. Static Block Pick and Place



(a) Start pose. (b) Fine detection pose. (c) Grasp pose. (d) Stack pose.

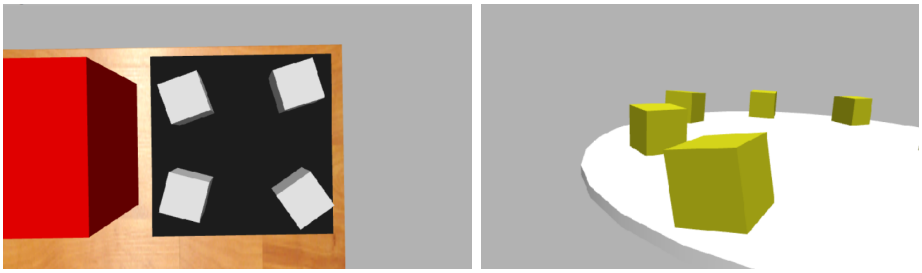
Fig. 3: Key pose for static block pick and place in simulation.

The static block pick-and-place procedure follows an iterative detection, grasping, and placement pipeline. In each iteration, the robot detects all static blocks at the start pose (Fig. 3a), a calibrated

configuration that ensures full visibility of the initial platform (Fig. 4a). Blocks that have already been placed on the target platform are filtered out to avoid redundant manipulation. To improve grasp accuracy, a two-stage perception strategy is employed. First, a coarse block pose is obtained from the start pose. Next, the robot moves to a pose located $15\,\mathrm{cm}$ above the selected block and performs fine detection from close range (Fig. 3b), which significantly reduces perception noise and improves pose estimation accuracy. During the grasping phase, the gripper uses the refined pose to grasp the block while adapting to its orientation (Fig. 3c).

Upon a successful grasp, the placement height along the z-axis is determined by the number of blocks already stacked on the target platform. Inverse kinematics is then solved to compute a stacking configuration, referred to as the stack pose (Fig. 3d). In practice, the placement accuracy is dominated by the quality of the inverse kinematics solution. We observed consistently accurate and stable IK solutions, with placement failures occurring almost exclusively when the preceding grasp was unstable. Since grasp stability is explicitly ensured by the perception and alignment strategies described earlier, we assume that each stacking operation succeeds once a block is successfully grasped. This assumption was found to be more robust than earlier approaches that relied on perceptually estimating the height of the stacked blocks, which were more sensitive to sensing noise.

An observe pose was initially introduced to improve estimation of the stacked block height after each placement. However, as the system evolved toward a success number based stacking height computation, this perception-based role became unnecessary. In the final design, it is retained solely as a high-clearance intermediate configuration to avoid collisions between the robot and previously stacked blocks during arm motion. After placing a block, the robot retreats to the observe pose, which is positioned sufficiently high above the target platform to ensure safe clearance before transitioning to the next iteration. If either the grasping or placement step fails, the end-effector returns to the start pose and resumes detection in the next iteration. This recovery mechanism allows the system to handle perception errors and occasional grasp failures without manual intervention. Once four static blocks have been successfully placed, or the maximum number of attempts has been exceeded, the system will forward to the pick-and-place phase for dynamic blocks.



(a) Top down view of static initial platform. (b) Horizontal view of dynamic blocks.

Fig. 4: Camera view at static start pose and dynamic prepare pose.

E. Dynamic Block Pick and Place

Dynamic blocks are dispersed on a rotating turntable, making precise perception and real-time tracking challenging. Instead of continuously estimating block motion, we adopt a robust **horizontal sweeping grasp strategy** based on predefined joint-space trajectories. Unlike top-down grasping with vertical motion, this approach performs a horizontal grasp by moving the end-effector laterally toward the block. This strategy is more robust under motion uncertainty, since the block can still be enclosed by the gripper without explicit prediction of its instantaneous position. We further observe that the robot arm approaches a near-singular configuration during horizontal grasping of dynamic blocks. Therefore, using fixed joint-space postures is more stable and reliable than repeatedly solving inverse kinematics for dynamic targets.

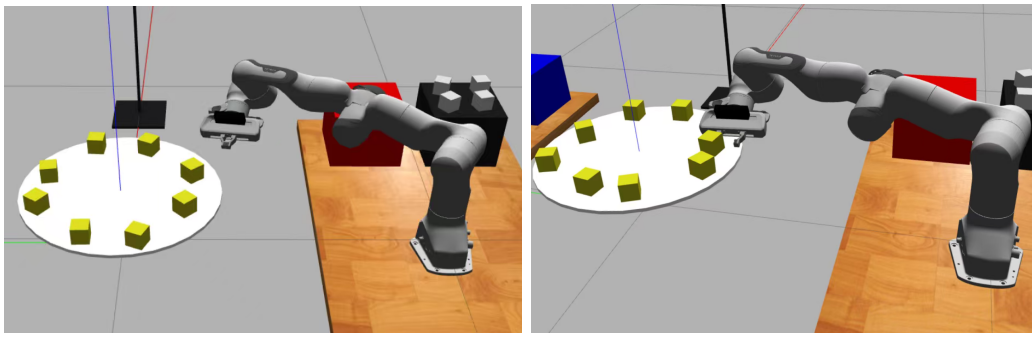
For each team (red and blue), two sets of joint-space grasping trajectories are manually tuned and hard-coded: an **edge sweep** and a **middle sweep**. Each sweep is defined by a triple of joint configurations consisting of a *prepare pose*, a *grasp pose*, and a *lift pose*. Together, these poses specify a horizontal sweeping motion designed to intercept dynamic blocks on the rotating turntable.

The edge sweep is used by default. Dynamic block detection is performed at the corresponding prepare pose, which orients the camera to observe approximately half of the rotating turntable (Fig. 4b). If a detected block lies within a predefined range in the x-y plane, it is considered graspable. The gripper then sweeps from the prepare pose toward the grasp pose (Fig. 5), remains stationary for a fixed duration of 8 seconds to allow blocks to pass due to platform rotation, and then closes to attempt a grasp. If the grasp fails, the arm returns to the prepare pose and repeats the same motion.

Repeated failures can significantly alter the block distribution on the turntable, often pushing blocks toward the center. To account for this effect, an adaptive fallback strategy is employed. After a predefined number of consecutive failed attempts, the controller switches from the edge sweep to the middle sweep. In this mode, a separate set of prepare, grasp, and lift poses is used, with the grasp pose pointing toward the center of the turntable. This orientation reduces the likelihood that blocks slide into the gripper during the sweep. Consequently, the waiting time at the grasp pose is reduced to 2 seconds, improving efficiency while maintaining robust grasping behavior.

If a dynamic block is successfully grasped, the end-effector first moves to a predefined lift pose above the target platform (Fig. 6b), followed by a transition to a IK Computed stack pose (Fig. 6c). Introducing an intermediate lift pose is vital because it ensures that IK can be solved every time. Additionally, it reduces the risk of collisions with previously stacked blocks that may occur when moving directly from the grasp pose to the stack pose, and prevents the robot from remaining near the turntable during inverse kinematics computation. The stacking height is determined based on the number of blocks already placed, and inverse kinematics is then solved to compute a stacking pose for final placement.

During placement, we explicitly specify an `input_rotation_matrix` to fix the end-effector orientation. This constraint ensures that the gripper remains parallel to the platform surface while placing dynamic blocks. By maintaining a consistent placement orientation, the risk of collision with previously placed blocks is reduced, leading to more stable and reliable stacking behavior.



(a) Prepare Pose. (b) Grasp Pose.

Fig. 5: Key pose for dynamic block picking.

III. Evaluation

We conducted qualitative and quantitative tests in simulation and on hardware to evaluate the the pick-and-place pipeline with an emphasis on perception reliability, grasp success, stacking stability, motion safety, and dynamic grasp feasibility.

A. Simulation Evaluation

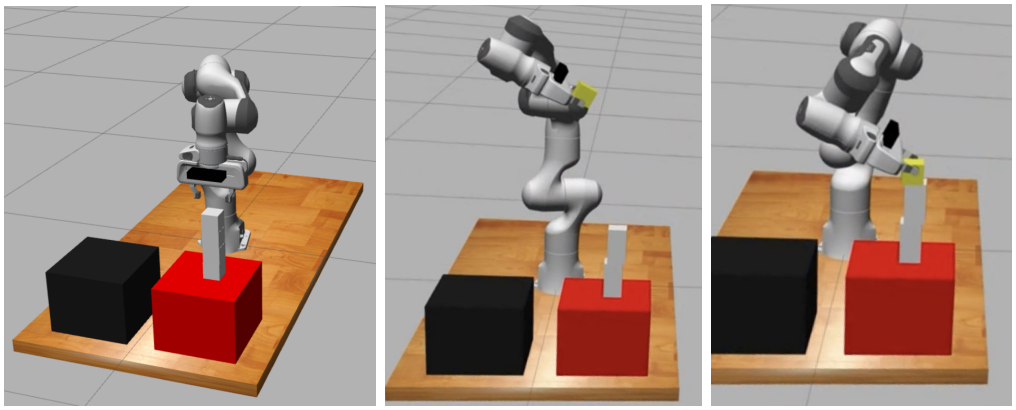
1) Qualitative Observations: **Test 1: Visibility and Perception of All Static Blocks.** From the observe pose, all four static blocks were consistently detected, confirming full camera coverage of the platform. The pose readouts in simulation (noise-free) verified that tag-based estimation was stable, ensuring that downstream errors originate from our algorithm rather than perception issues.

Test 2: Grasping and Stacking Orientation for Static Blocks. We verified that the gripper aligns with the two flat faces of each block before grasping, preventing diagonal grasps and reducing yaw drift in the tower. In simulation, alignment was consistently correct, and stacked blocks remained nearly vertical with minimal misalignment (See Fig. 6b). This indicates that our yaw-alignment logic and IK solver provided reliable grasps and placements.

Test 3: Visibility, Safety, and Gripper Alignment for Dynamic Blocks. At the prepare pose, the camera maintained visibility of dynamic blocks on the left half of the turntable, satisfying the requirement that the robot should "see before grasp." While transitioning to grasp pose, the arm cleared the turntable safely with no collisions. At the grasp pose, the gripper was parallel to the tangent of the turntable path, ensuring that the gripper aperture faced the direction of incoming blocks.

Test 4: Passive Capture of a Dynamic Block. The gripper will stay open at grasp pose for 8 seconds, allowing the rotating block to translate into the gripper. Due to an unrealistically large friction coefficient ($\mu = 4$) in the Gazebo environment, significant resistance is created that prevents the block from sliding naturally into the gripper. Out of five trials, only three resulted in successful capture. This illustrates a clear limitation imposed by simulation physics rather than algorithmic design. It also demonstrates that our pipeline is sensitive to realistic sliding behavior, which is expected to improve on hardware.

Test 5: Full Static & Dynamic Blocks End-to-End Pipeline. We executed the detection-grasp-stack pipeline to evaluate cumulative effects such as error propagation. Across trials, the tower remained fairly straight and stable with a little misalignment, and no collisions occurred during transitions between waypoints. This demonstrated that the pipeline maintains robustness over multiple sequential manipulations.



(a) Result of static blocks. (b) Lift pose before stacking. (c) IK computed stack pose.

Fig. 6: Static pick and place result (left) and key pose for dynamic block stacking (middle and right).

2) Quantitative Results: Table I summarizes the quantitative results for both static and dynamic tests. For static blocks, we report total completion time, grasp orientation correctness, success rate, number of blocks stacked, and stacking misalignment. For dynamic blocks, we report number of successful captures in 2 minutes, stacking misalignment, and success rate. The static tests exhibit perfect grasp success and stacking accuracy across all trials, with an average runtime of 145.57 s. This confirms the reliability of the static pipeline under noise-free conditions. The dynamic results reveal a clear limitation: the simulated high friction significantly reduces the feasibility of passive grasping. Nevertheless, it can stack around 2

blocks in 2 minutes, demonstrating that we can move to hardware testing and is expected to improve on hardware where the physical friction coefficient is realistic.

TABLE I: Quantitative Results for Static and Dynamic Simulation Tests

Category	Time (s) ↓	# Stacked ↑	Stack Alignment	Success Rate ↑
Static Trial 1	145.13	4	Aligned	100%
Static Trial 2	145.17	4	Slightly misaligned	100%
Static Trial 3	147.10	4	Aligned	100%
Static Trial 4	146.31	4	Aligned	100%
Static Trial 5	144.80	4	Slightly misaligned	100%
Static (Avg.)	145.7	4	Sometimes	100%
Dynamic Trial 1	120	1	Slightly misaligned	100%
Dynamic Trial 2	120	0	Aligned	0%
Dynamic Trial 3	120	2	Slightly misaligned	100%
Dynamic Trial 4	120	1	Slightly misaligned	100%
Dynamic Trial 4	120	0	Aligned	0%
Dynamic (Avg.)	120	0–2 per 2 min	Sometimes	60%

B. Hardware Evaluation

Once the simulation pipeline was reliable, we moved these same tests to the real robot. The procedures were identical, but now we were evaluating how well our system handled perception noise, calibration differences, and real-world uncertainty.

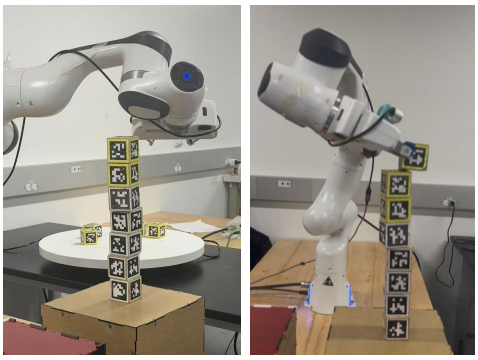
We checked that all static blocks were detected at the observe position. This revealed early on that hardware perception misses blocks more frequently than simulation. We also tested grasping and stacking orientation but now with camera noise. This allowed us to tune thresholds and introduce camera offsets. We watched for micro-collisions and compliance effects that could destabilize the tower. Finally, we attempted stacking all blocks sequentially.

1) Static Blocks on Hardware: For the static blocks, the hardware behavior closely matched simulation. In multiple runs the robot successfully detected, grasped, and stacked all four static blocks into a straight tower. The resulting tower (bottom four blocks in Fig. 7a) showed only slight misalignment, indicating that the Z–height placement logic based on count of successful grabbed block transferred well from simulation to hardware. A key difference from simulation, however, was the need to carefully tune the camera offset in camera frame. Small errors in this offset produced systematic grasping biases, leading to fingertip contact on the block corners. Through repeated trials we manually adjusted the offset parameters until the gripper consistently closed around the flat faces of the blocks. This highlights that camera extrinsics are a dominant source of error in the real system.

2) Dynamic Blocks on Hardware: Joint-space configurations that worked in simulation did not directly transfer to hardware. We therefore used the guiding mode on the Panda arm to further tuned the prepare, grasp, and stack poses for the dynamic sequence so that the sweeping motion of the gripper covered the region where dynamic blocks typically appear. With these adjustments, the robot successfully stacked three dynamic blocks on top of the four static ones, forming a seven-block tower (Fig. 7a). The tower remained upright but showed a small amount of tilt, reflecting accumulated orientation error and the fact that the dynamic blocks entered the gripper with slightly noisier poses than the static ones.

An important limitation observed on hardware is that the sweeping motion of the arm sometimes pushes dynamic blocks either inward or outward on the turntable instead of capturing them. To probe robustness, we manually displaced some dynamic blocks away from the nominal grasp radius and then allowed the robot to attempt a grasp. In several cases the robot was still able to capture the block, demonstrating some tolerance to radius errors; in other cases the block was nudged further away and missed. The most illustrative failure is shown in Fig. 7b. After stacking four static and three dynamic blocks, the robot barely captured an eighth block in a “weird” pose. This happened because the gripper is set to close after

8 seconds at grasp pose, and the gripper closed before the block conforms to the shape of the gripper as the turntable turns. The block had significant tilt relative to the tower axis when it left the gripper. When this block was placed on top, its center of mass was far from the tower centerline, causing the entire stack to tip over. This failure mode reveals an important limitation of our current pipeline: we do not explicitly check the quality of the estimated block pose or predict tower stability before stacking.



(a) 7-block tower (b) Failure case

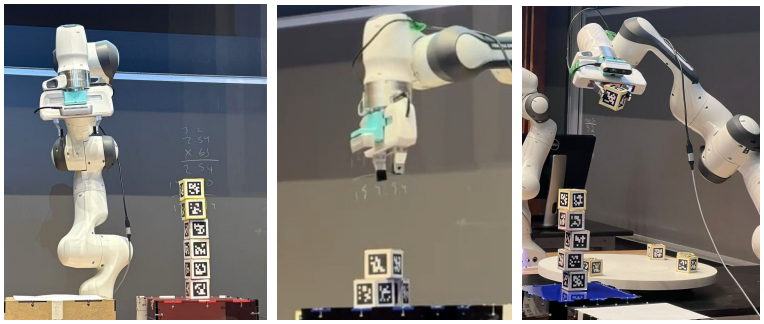
Fig. 7: Hardware stacking: successful 7-block tower and failure case during 8th dynamic block stacking.

C. Competition Results

Our final system won the First Place in the MEAM 5200 Pick-and-Place Competition.

1) Best Performance: Our best match resulted in stacking 4 static blocks + 2 dynamic blocks (Fig. 7a), for a total of 14,000 points. In this run, the static stack was perfectly aligned and stable, and 2 dynamic blocks were captured using the sweep-based grasping motion. The tower remained upright for the entire match with a slight misalignment between static dynamic stack, demonstrating that our pipeline is capable of stacking both static and dynamic blocks.

2) Lowest Performance: Our lowest-scoring match yielded only 1,250 points, consisting of two static blocks stacked at the first altitude level and one static block placed at the second altitude level. This underperformance was caused by camera offset issues on the blue-team robot. Because we did not manage to fully tune the camera extrinsics during the limited hardware time, the robot frequently grasped static blocks off-center. As a result, the second static block collapsed to the base level. The misaligned stacked tower in Fig. 8c also reflects this issue. This failure directly exposed a limitation in our stacking logic, which will be discussed in Analysis.



(a) Highest score (b) Lowest score (c) Misaligned

Fig. 8: Competition results on red and blue teams.

IV. Analysis

In general, our strategy performed well in both simulation and real-world experiments, though some discrepancies were observed between the two environments. In the analysis section, we will discuss the general methods analysis for both static and dynamic block grasping strategies, examine the performance discrepancies between simulation, hardware testing, and competition, analyze the error and failure modes encountered during execution, investigate future improvement, and provide a quantitative summary of our results. Our results are summarized in Tab II, which reports the maximum number of blocks stacked.

TABLE II: Maximum Number of Stacked Blocks

	Static	Dynamic
	Max Stacked Blocks ↑	Max Stacked Blocks ↑
Simulation	4	2
Hardware	4	3
Competition	4	2

A. General Methods Analysis

1) ***Static Block Grasping Strategy Analysis****:* From Table I, all the static block pick-and-place tasks achieved perfect performance, successfully grasping all four blocks in every run due to appropriate start pose, accurate pose detection, precise gripper alignment, and robust placement decisions.

Start position: Accurate block detection requires that all static blocks are fully visible within the camera's field of view. Although the camera viewpoint was tuned in simulation, the same pose did not always provide complete visibility in the real-world setup. The camera start pose was therefore re-adjusted on the physical system to ensure consistent block visibility.

Block Pose Detection: Residual camera offsets in the real-world setup required careful handling. While applying offsets in the camera frame should theoretically suffice with proper calibration, we found this approach inconsistent, which led to a misaligned tower in Fig. 8c. Offsets sometimes varied between blocks in the same run. In contrast, applying offsets directly in the robot base frame, as other teams successfully did, provided more stable corrections. This inconsistency likely stems from focal length miscalibration or minor camera misalignment (translational shifts or pitch errors), which introduce perception biases that manifest unpredictably in the camera frame. Add offsets in base frame offer improved robustness for future system refinement.

The Gripper and Block Alignment: 1. After detecting the block position, accurate gripper–block alignment is critical for reliable grasping and placement. Initially, block orientation was estimated by assuming the first two columns of the rotation matrix corresponded to the block's x and y axes. However, this assumption does not always hold, as the detected axes may be inconsistent, leading to incorrect orientation estimation. Without explicit alignment constraints, diagonal approaches may occur, leading to misaligned placements, shifted centers of mass, and increased instability as the stack height grows.

To address this, we adopt a "most horizontal" axis strategy, which identifies the block axis closest to the horizontal plane and aligns the gripper accordingly. This approach improves robustness when the detected z-axis deviates from the global vertical, significantly enhancing gripper block alignment reliability.

2. We also considered quaternions instead of rotation matrices to mitigate gimbal lock issues from Euler-angle representations. Quaternion orientation is singularity-free and numerically stable for 3D rotations, reducing discontinuities and abrupt changes during grasp planning. However, our alignment operates on axis directions extracted from the rotation matrix and does not rely on Euler-angle decomposition, so

gimbal-lock risk is already low. Thus, while quaternions offer theoretical advantages, the "most horizontal" axis approach was sufficient for robust, stable gripper alignment in our system.

Placement Position and Height: Block placement is the most critical pipeline stage. Initially, we estimated placement height from the highest stacked block's surface, or platform height plus half-block height (with safety margin) if empty. However, this approach was unreliable beyond the first layer due to limited depth-camera z-axis accuracy introducing significant noise. We considered two alternatives: (1) classify the number of stacked blocks by detecting which height interval the top surface falls into, then assign the corresponding height; (2) assume each placement succeeds and compute height from the count of successfully placed blocks. We chose (2) because (1) is more sensitive to perception noise and prone to misclassification under large depth errors. Since all placements succeeded consistently in our experiments (See Fig. 7a), this assumption proved reliable. However, dynamic manipulation experiments revealed its limitations. Fig. 9 shows an example dropping a block from an inaccurate stack height after one block fell down. Future work will evaluate whether interval-based height estimation improves robustness under more challenging conditions.

*2) **Dynamic Block Grasping Strategy Analysis**:* Our dynamic grasping strategy achieved moderate success but exhibited inconsistent performance: not all grasp attempts succeeded, and stacked blocks were occasionally misaligned. We analyze the contributing factors at each pipeline stage below. Dynamic block grasping logic follows the flow of: Prepare Position $\rightarrow$ Detect Dynamic Blocks $\rightarrow$ Execute Grasp $\rightarrow$ Move to Stack Position $\rightarrow$ Place Block. We analyze our strategy along this logic flow.

Prepare Position: We fine-tuned a preparation pose that positions the arm extended over the turntable with the gripper angled downward and rotated inward, ready to intercept incoming blocks. It was selected for several reasons: (1) it provides a safe configuration that avoids collisions with the turntable by positioning the gripper in an "elbow-down" orientation and maintaining a small clearance height above the platform to prevent contact, (2) it positions the end-effector camera with a clear field of view to detect dynamic blocks, and (3) It pre-configures the arm in a close orientation for grasping, minimizing joint motion to reach the target, and serves as a fallback pose if grasping fails.

Detect Dynamic Blocks: Our detection strategy employs a simple approach: selecting the block with the Y-coordinate closest to the origin, which corresponds to the block nearest to the turntable edge. This method offers several advantages: (1) it requires minimal computation compared to trajectory prediction or visual servoing approaches, and (2) blocks at the edge have more predictable arrival times at the grasp position. However, this approach has limitations: it assumes blocks distributed across the turntable and does not account for scenarios in which opponent robots push blocks toward the center. In such cases, no blocks may be detected at the edge, causing repeated failed attempts.To address this, a fallback mechanism is used: after a set number of consecutive failures, the system switches to a middle-position strategy, reorienting the gripper toward the turntable center to capture blocks in the central region.

Execute Grasp: Upon detecting a block, the system executes a fine-tuned grasp pose via safe motion. This pose was calibrated through iterative testing, considering arm approach trajectory and block motion, so the gripper trajectory intersects the block's expected path. The gripper is inclined relative to the block: in the block x/y-z cross-sectional plane, the gripper z-axis forms an angle with the block z-axis, allowing interception as the block rotates into position. This provides built-in redundancy: with minor positioning errors, the open gripper aperture acts as a capture zone, letting the block slide into the gripper. Compared to top-down grasping (e.g., static grasping), this simple sweep-based approach works is more robust for moving blocks by tolerating positional variation along the travel direction. It works very well on hardware as shown in Fig. 7a where we successfully grasped and stacked 3 dynamic blocks.

Move to Stack Position: After grasping the block, the arm lifts up horizontally to an intermediate lift pose. However, solving IK from the lift pose to the stack pose was unreliable, since the lift configuration

is a poor seed for IK convergence. To address this, we added a stack preparation position: a hand-tuned intermediate configuration at higher elevation that provides a reliable IK seed for the final placement pose at each stacking height (See Fig. 6b). Now IK can be solved consistently and fast with normally less than 10 iterations every time.

Place Block: For stable placement, we used the robot's guiding mode to record a reference joint configuration that preserves grasp orientation while allowing only translation in $\mathbb{R}^3$ and yaw rotation. This $\mathrm{SE}(2)$ constraint removes roll and pitch variations that could destabilize the stack. We applied FK to the recorded configuration to obtain the rotation matrix. For stacking, x and y are hardcoded to the platform center, while z is computed from the current stack height. These coordinates, combined with the rotation matrix, define the target pose for IK solving. We chose this method because fixing the rotation matrix and varying only xyz improves placement stability and reduces IK time. This method ensured very straight static blocks stacking (See Fig. 8a) as long as camera offset is well tuned. However, this method assumes each placement succeeds: if a block fails to stack, the system still increments the stack count, so later z calculations become incorrect, causing cascading failures. Fig. 9 shows such a case. A vision-based feedback mechanism could detect the actual stack height in real time to address this limitation.

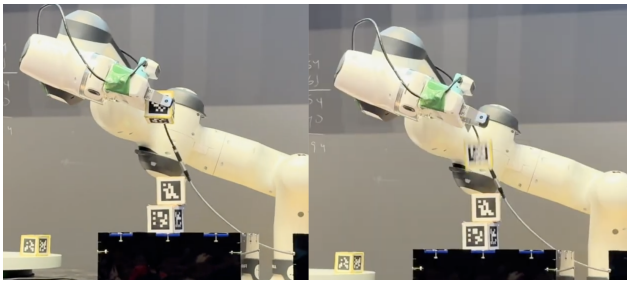


Fig. 9: Failure scenario resulting from an inaccurate stack height

B. Simulation vs. Hardware Performance Discrepancy Analysis

We observed several simulation-to-reality gaps, due to: **(1) Camera–End-Effector Extrinsic Misalignment:** In simulation, static blocks are stacked straight and stable as shown in Table I. Despite intrinsic calibration, the assumed camera-to-robot extrinsic pose differed from the physical setup. Small mounting translation/rotation errors caused a consistent planar offset when projecting detected block poses into the robot base frame, so the end-effector missed the true center. In competition run on the blue team, camera offset was not well-tuned, which caused a tilted tower in Fig. 8c. **(2) Depth Estimation Error:** Single-view detection produced inaccurate z-axis measurements and unreliable height estimates. We enforced a fixed z-height using the known table surface height. **(3) Field of View Discrepancy:** Simulated and real cameras had different FOV at the same height, so some blocks were detected only in simulation. This required re-tuning detection positions on hardware. **(4) Configuration Inconsistency:** Pre-tuned joint configurations from simulation did not always transfer to hardware due to robot dynamics and environment differences. We resolved this via iterative physical testing and refinement. **(5) Joint Limit Handling:** Simulation used soft joint-limit tolerance, while hardware enforced strict physical limits. Near-boundary configurations that worked in simulation sometimes failed on hardware. **(6) Friction Coefficient Difference:** Friction between blocks and the turntable was higher in simulation than in reality. This made blocks harder to grasp in simulation under our method, yielding fewer missed grasps than on hardware, where blocks shifted more easily.

We also observed discrepancies between lab testing and the competition environment: **(1) Lighting Condition Variability:** Venue lighting differed from our lab, affecting AprilTag detection. We implemented two-phase detection: coarse detection from the start position, then fine detection at close range, improving robustness under variable lighting. **(2) Opponent Interference:** Competition included a simultaneous opponent robot. Our dynamic grasping assumed blocks stayed at the turntable edge, but opponents could push them inward. We added a fallback middle-position mode.

V. Lessons Learned

A. Insights Gained

We learned the system behaves very differently on real hardware than in simulation. Poses tuned in simulation often required retuning on the physical robot: small differences in kinematics, collision geometry, and joint limits made some simulated poses unsafe or unreachable on hardware. Perception also differed. Simulation provides noise-free extrinsics and detections, while on hardware AprilTag poses were biased by a nonzero camera offset. Since the offset was applied in the camera frame, small calibration errors produced noticeable grasp errors. Tuning this offset was one of the most difficult tasks, emphasizing the need for a proper camera-extrinsic calibration routine. In our lowest-scoring run, we found stacking should not rely only on an internal grasp count. Because the tower can collapse or shift, the count may not match the true height, leading to incorrect placement altitude. Overall, simulation is only a starting point: hardware needs larger margins, robustness, and careful pose/perception tuning.

B. Benefits and Drawbacks of Our Pipeline

Static grasping and stacking were highly reliable. The gripper consistently contacted the two flat faces of each block, producing stable grasps and well-aligned towers. Placement height was accurate, and all four static blocks stacked with minimal misalignment in both simulation and hardware. For dynamic blocks, our sweep-based method worked well after hardware tuning. The robot reliably captured blocks as they passed through a fixed radial region, avoiding motion prediction and requiring mainly robust joint configurations and a sweeping trajectory. The main weakness was dynamic stacking uncertainty. We did not verify block pose/orientation before stacking, so the robot stacked even if the block was tilted or off-center, reducing tower stability (e.g., the poorly oriented eighth block caused collapse). The sweep is also sensitive to radius changes: blocks shifted inward/outward can be missed. Skewed grasps and rotational errors further weakened stacking stability.

C. Future Improvements

Static stacking was precise and robust, and the sweep strategy enabled reliable dynamic grasps with low complexity. However, dynamic stacking remains fragile due to pose uncertainty and missing grasp-quality checks, highlighting the need to handle real-world sensing and grasp variability.

Possible improvements:

- Evaluate grasp quality before stacking; regrasp if orientation is poor.
- Improve stacking feedback:
  - detect the top block position before placement;
  - estimate tower height from perception instead of an internal counter;
- Estimate block centers more accurately for symmetric, stable grasps.
- Adapt prepare/grasp poses using detected block positions rather than fixed waypoints.
- Add limited timing prediction to better align with turntable motion.
- Implement automatic camera-offset calibration.

REFERENCES

[1] Apriorit, "How to fix image distortions using opencv," https://www.apriorit.com/dev-blog/ai-fix-image-distortions-using-opencv, 2023, accessed: 2025-12-18.