BlockBuilder: Autonomous Color Block Selection and Structured Arrangement

Kshitij Bhat, Manyung Hon, Juan Muerto, McKenzie Young

***Abstract*— This paper presents the design and implementation of an autonomous robotic system for color-based block picking and structured placement, utilizing a seven-degree-of-freedom Franka Emika robotic arm. The system integrates advanced perception and manipulation techniques to detect, classify, and grasp colored cubes from a designated picking area and assemble them into organized rows based on predefined color patterns. The perception pipeline employs a depth camera for accurate block localization and color classification, while a robot-mounted camera monitors the assembly process. Color patterns are extracted automatically from pixel art images using image processing tools, which convert visual designs into actionable building instructions for the robot. The system demonstrates robust integration of hardware and software components to perform precise, repeatable manipulation tasks with minimal human intervention. Experimental evaluations confirm the effectiveness of the proposed method in constructing color walls accurately and efficiently, even in dynamic environments. Our results demonstrate the potential of combining computer vision, robotic manipulation, and automated planning for creative and industrial applications, particularly in tasks requiring precision and repeatability.**

I. INTRODUCTION

Manipulation and perception are the foundations of robotics research and industry applications. These two systems, in combination or separately, are used in nearly every robotics use case. Using both of these subsystems effectively is critical to a successful robotics deployment. One such case where this melding is particularly important is work involving picking and placing, that is, the action of picking an item and placing it elsewhere.

In this paper, we propose a two-step system, one that combines key components from manipulation and perception topics to properly demonstrate understanding of the fundamentals of robotic systems. Our proposal is a system for selecting certain colors from a pool of colored cubes (two by two inches) and constructing rows of the desired colors. Although a seemingly simple task, this work tests our understanding of the fundamental tenets of robotics by requiring us to implement cube detection, color classification, obstacle detection, path planning, cube grasping, and cube placement.

Our system is composed of a seven degree-of-freedom Franka Emika robotic arm with a gripper as an end effector. There is also a camera mounted to the end effector so as to be able to classify and identify blocks.

The remainder of this report is organized into four additional sections. Section II introduces other work being done in the area of autonomous manipulation, Section III introduces the system in more detail, Section IV evaluates the success of our system, and, finally, Section V identifies areas for improvement and concludes the paper.

The authors are with the Robotics Institute at Carnegie Mellon University

II. RELATED WORK

In this section, we provide an overview of relevant work on autonomous robots for color sorting, bin picking, and block stacking. These tasks are essential to the success of our pixel-artist robot, and they have useful applications in many industries, including logistics and manufacturing.

A. Color Sorting

Bhapkar et al. [1] introduced a new automation system referred to as the Color Sorting Pick-up and Drop Robot Arm. This 6 DOF system categorizes an object by color using a TCS3200 color sensor, grasps and lifts the object, and drops the object into a box corresponding to its color category. Communication between the color sensor and the servo motors that govern the robot arm's movement was handled by an Arduino Uno. Lim Jie et al. [2] designed and implemented a similar system which integrated a wireless control interface, the Blynk app, allowing the user to give explicit commands or enter the "Automatic Color Sorting Mode". Both of these systems rely on being able to drop objects into a desired bin, rather than placing the objects in a specified location or pattern.

B. Bin Picking and Block Stacking

Macias et al. [3] developed a 6 DOF vision-guided robotic system to pick up blocks from a random pile and stack them in a tower. The system incorporated a wrist-mounted webcam and utilized predefined 2D symbols known as markers to identify objects. A unique set of 9 markers was affixed to each block; the central marker determines the block's position, and the 8 support markers allow the robot to estimate the block's orientation. The system assigns a cost to each block as a measure of how difficult it is to remove from the pile. This cost function determines the order in which the blocks are picked up and placed. Chen et al. [4] offer a different approach to block stacking. Their research focuses on two types of robots working in tandem; the "handling robot" picks up blocks using a 2-finger gripper and stacks them neatly on a "logistics robot" to be transported. The handling robot incorporates a 2-phase approach to object localization: grasping localization and in-hand localization. Initially, the robot locates and grasps a target block. It then repositions itself into a predetermined state to measure the in-hand pose of the block. This information is used to guide the robot to place the block on the logistics robot.

III. METHODOLOGY

The methodology outlines the system hardware setup and forward process workflow for the robotic picking and placing task. The workspace is divided into a picking area, where colored cubes are detected and classified, and a building area, where rows of cubes are constructed. A RealSense camera mounted on the Franka Emika robotic arm identifies block poses and colors. The robot follows a forward process: detecting and grasping blocks in the picking area, navigating to the building area, and placing them in the desired arrangement.

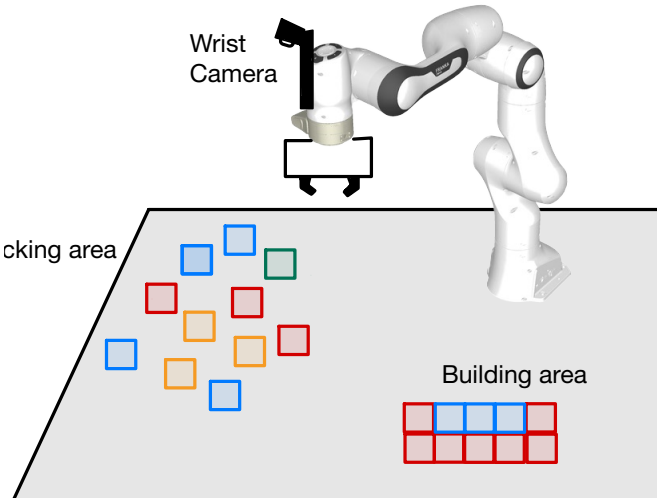

Fig. 1. Hardware Setup

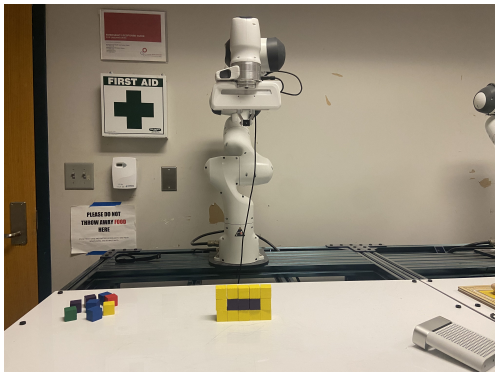

Fig. 2. Photo of hardware setup

A. Hardware Setup

The system is configured as shown in Fig. 1, with the workspace divided into two areas: the picking area and the building area. The picking area serves as the location where colored cubes are initially placed, while the building area is designated for constructing rows of cubes based on the desired color sequence. This separation ensures a clear workflow for the robotic system and minimizes interference between tasks.

A RealSense camera is mounted on the end effector of the seven degree-of-freedom Franka Emika robotic arm. This depth camera plays a critical role in detecting blocks within the picking area, classifying their colors, and calculating their precise poses for accurate manipulation.

B. Forward Process

The forward process workflow begins with the robot identifying and selecting a block from the picking area using its mounted depth camera. Once a block is detected, its color is classified, and its pose is calculated to determine an optimal grasping strategy. The Franka Emika arm then uses its gripper to securely pick up the block.

After successfully grasping a block, the robot navigates to the building area while avoiding any obstacles in its path. The block is then placed in its designated position within the color wall based on the desired arrangement. This process repeats iteratively until all rows in the building area are completed according to specification.

By using the end-effector-mounted depth camera for detection and classification the system ensures robust performance throughout the picking and placing tasks. This setup not only demonstrates effective integration of manipulation and perception subsystems but also highlights their importance in achieving precise and reliable robotic operations.

C. Color Pattern Design and Extraction

The artwork is first designed using online pixel art tools such as Piskel, then exported as a PNG image. Using the Pillow and webcolors libraries, the image is processed row by row, analyzing pixel data from top to bottom. However, since the robot constructs the pixel-art wall from the bottom row upward, the extracted list of rows must be flipped vertically to match the build order. This ensures that the robot starts with the correct bottom row and progresses upward as it places each block.

During processing, transparent pixels are ignored, and visible pixels are mapped to their nearest CSS3 color name using webcolors. These names are then further simplified and remapped to a smaller, predefined set of colors to match the physical brick palette available to the robot. To reduce redundancy, consecutive duplicate colors within each row are compressed. Additionally, only visually unique rows are retained to avoid unnecessary repetition in the placement process. The final result is a clean and structured list of row instructions that the robot uses during the stacking process.

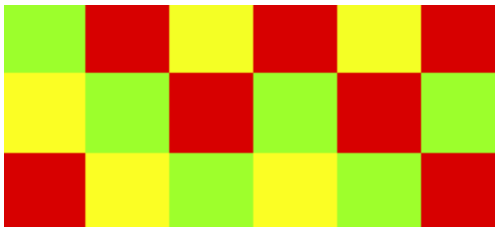

Fig. 3. Designed Pixel Art

```
['red', 'yellow', 'green', 'yellow', 'green', 'red']
['yellow', 'green', 'red', 'green', 'red', 'green']
['green', 'red', 'yellow', 'red', 'yellow', 'red']
```

Fig. 4. Analyzed Output

D. Perception

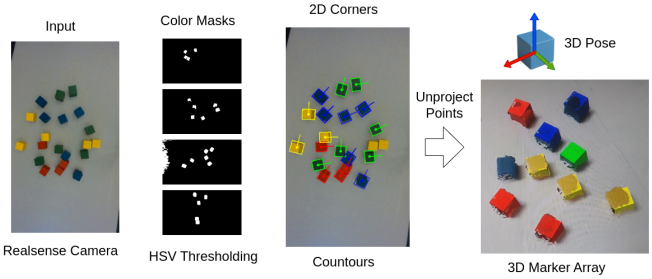

Fig. 5. Perception Node

Figure 5 illustrates our robot perception pipeline for block manipulation, beginning with a RealSense camera capturing an image of colored blocks scattered in the picking area in any position and orientation. The system processes this through HSV thresholding to create color masks that isolate different colored blocks. Next, 2D corners are identified with contours drawn around each detected block. Finally, these 2D points are unprojected into 3D space to calculate the poses, enabling the Franka Emika robotic arm to accurately identify and grasp blocks of specific colors from the picking area.

E. Block Selection and Grasping

Given the list of blocks that is provided by the perception subsystem, the planning system searches for a block that matches the ”desired” color that the placement plan requires. Using the ROS message provided by perception, the planning subsystem searches for the desired block by comparing the RGB values of each reported block against a list of hard coded thresholds.

After finding a valid block, the planner then calculates the pose necessary for the end effector to grasp the block. Subsequently, the planning subsystem also calculates a pose that is relatively close to the block to avoid collisions.

Once it has calculated these poses the planning system performs the following:

- Opens the gripper.
- Sends the manipulator to the ”close” position.
- Sends the manipulator to the ”pick” position.
- Closes the gripper.
- Sends the manipulator to the ”close” position.
- If it fails to pick, repeat the steps from the block finding step.
- Otherwise, send the manipulator to the ”neutral” position to prepare for placement.

By following this plan, we are able to separate the picking and placing routines to allow for optimal parallel development, despite perhaps not being the most optimal routine overall.

In order to increase robustness, our team has added a ”repick” routine. That is, when a pick is ”unsuccessful”, the system will send the manipulator back into its ”observation” pose, and attempt to find a block of the desired color. This gives the system some leeway in cases where the block has been removed, or the perception system provides a slightly inaccurate block position.

We also have worked towards increasing the robustness of the block position gathering. Initially, the planning subsystem was only getting a single iteration of the block positions from the perception system. Rather than relying on perfect reporting on every perception message, we had the planning subsystem sample the block positions reported by perception and then average those positions when the positions fall within a certain range.

F. Block Placement and Stacking

Once the robot has picked the desired block and returned to the neutral position, it must place that block in its corresponding position in the building area. The robot builds the color wall row-by-row in the building area, moving from left to right. The position of the first block, which is the leftmost block in the bottom row of the wall, is hard-coded into the placement program. Each following position is calculated according to two counter variables, ”row” and ”col”. ”row” updates the original position’s y-coordinate to determine the desired distance to the right of the first block, while ”col” updates the z-coordinate to determine the distance above the first block. ”row” resets every time a single row is completed, while ”col” does not reset until the entire wall has been built.

As in the picking routine, the placing routine also calculates a pose that is close to the desired placement pose to avoid collisions. Once the close and placement poses for a given block are calculated, the system executes the following:

- Send the manipulator to the ”close” position.
- Send the manipulator to the ”place” position.
- Open the gripper.
- Send the manipulator to the ”close” position.
- Send the manipulator to the ”neutral” position.

At this point, the system is ready to pick another block and restart the forward process.

G. Reset Process

Once the robot has completed its forward process, it is necessary to reset the environment in order to set ourselves up for cyclical testing. In order to do so, we will reuse much of the existing routines (block selection and placement).

Since we are already aware of where the blocks should be, we will send the manipulator to the most recently placed block, and pick it (using the procedure described previously). Since we have the position, there is no need to search for a block based on a desired color. Initially, we would send the manipulator back to its ”neutral” position and simply open the gripper. This is, of course, not ideal as it requires human intervention to catch the falling block and place it back in our picking space.

Because of this, we transitioned towards a more robust reset routine as follows.

Before we pick up a block, we must first search our picking space for free spaces where we can place a block from our wall. We must do this first, as once we have a block picked up, the block itself obfuscates our wrist camera. Therefore, we send the manipulator to its observation pose and determine where there are blocks currently. Once we have our list of blocks, we perform a simple algorithm:

- For each visible block.
  - Generate a candidate position (certain X/Y away from the visible block).
  - For every other visible block, check whether this candidate position would cause a collision.
- If there are no collisions, return the candidate position
- Otherwise, start over again with the next visible block.

Once we have a candidate position, it is a simple matter of picking the block and then placing it in this candidate position (reversing the forward process).

IV. Evaluation

Despite the various challenges faced during development and testing, our system achieved moderate success in autonomously constructing the colored wall. One of the key observations was that block orientation played a significant role in picking success. When blocks were consistently oriented in the same direction (i.e., no rotation), the robotic arm demonstrated a relatively high success rate in grasping and relocating blocks. This trend is evident in Figure 6, which shows picking performance across different block colors.

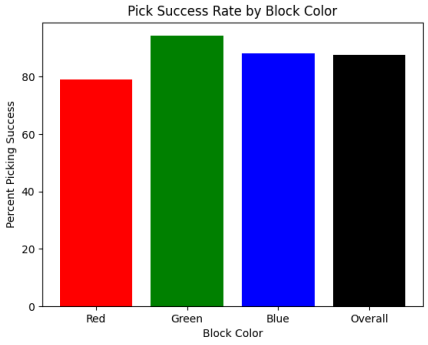

Fig. 6. Bar graph comparing picking success by color

In addition to color classification performance, we observed spatial variability in picking success based on the block's position within the picking area. As shown in Figures 7 and 8, blocks located near the center of the observation region were picked more reliably, while those closer to the edges of the workspace had a significantly higher failure rate. This positional sensitivity is likely due to increased visual distortion, depth inaccuracies, and less favorable viewing angles from the wrist-mounted camera when targeting blocks at the periphery. The effect was observed in both horizontal (X-axis) and vertical (Y-axis) directions.

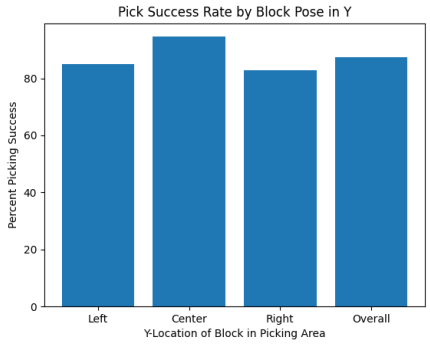

Fig. 7. Bar graph comparing picking success by block position in Y

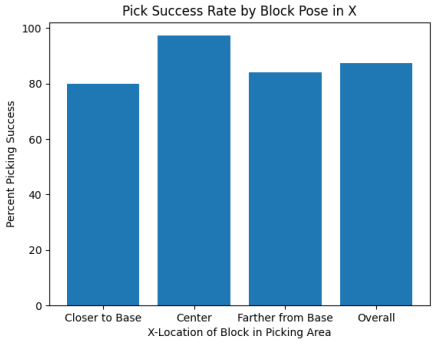

Fig. 8. Bar graph comparing picking success by block position in X

To mitigate these challenges, we implemented a "re-pick" routine. This routine allowed the robot to return to its observation pose and attempt a new grasp in cases where the initial attempt failed. While this mechanism introduced slight delays, it significantly increased the overall success rate by allowing recovery from minor perception or grasping inaccuracies.

However, our system still faces challenges in achieving precise grasp alignment with the true center of the block. Even slight deviations in the pick location propagate to the placement stage, often leading to misalignment when blocks are stacked. These inconsistencies can cause several types of structural issues: blocks may clip adjacent ones during placement, fail to sit flush with the block beneath, or, in more severe cases, topple due to imbalanced support.

These placement-related issues became more noticeable as we increased the size and complexity of the wall configurations. In wider configurations (i.e., more columns), the system experienced a higher rate of block clipping, potentially due to the accumulation of horizontal misalignments over successive placements. Interestingly, increasing the number of rows did not significantly affect the clipping rate, suggesting that horizontal tolerance plays a more critical role in structural consistency than vertical growth.

Conversely, tall and narrow wall configurations, such as those with a single column and many rows, were particularly prone to toppling failures. As the height of the structure increased, the reduced surface area available for placing

new blocks led to less stability. In these cases, even small deviations in center alignment during placement could destabilize the entire column. However, increasing the number of columns tended to reduce this failure mode, likely due to the broader base providing more lateral support for each block.

Overall, the evaluation highlights the system's ability to perform autonomous block manipulation with moderate reliability, particularly in controlled conditions. At the same time, it reveals key limitations in pose accuracy, grasp consistency, and structural stability that must be addressed for more robust and scalable performance. These insights directly inform the future work directions outlined in the next section.

V. CHALLENGES AND LIMITATIONS

While our system was able to complete the task of picking, classifying, and stacking colored blocks, several challenges impacted its performance and consistency. These challenges occurred primarily in the perception and manipulation stages of the pipeline.

A major issue arose in the color detection process. Our system relied on HSV thresholding to identify and classify blocks based on color. However, this approach was highly sensitive to lighting conditions. Inconsistent lighting across the workspace caused certain colors to fail detection entirely. Blue and green blocks were especially problematic. At times, the system would fail to identify them or confuse them with neighboring hues. Shadows, glare, and slight changes in camera exposure contributed to these inconsistencies. Furthermore, the similarity between certain color values, such as green and yellow, sometimes led to misclassifications, which disrupted the color sequence in the wall-building process.

The detection process was also affected by reflections from the surface of the table. The glossy table finish caused bright reflections that introduced noise into both the RGB and depth data. These reflections often confused the color segmentation pipeline and resulted in false positives or masked actual blocks. In some cases, the system incorrectly identified a reflection as a block or failed to detect a real block because it was partially washed out by glare.

Rotation detection of the blocks also presented challenges. Since blocks could appear in arbitrary orientations within the picking area, accurate pose estimation was essential for reliable grasping. However, due to the lack of distinct surface features and the symmetry of the blocks, estimating the correct orientation was difficult. Small rotational inaccuracies were often not caught until the robot attempted to grasp the block. In these cases, the gripper would approach from a suboptimal angle, causing the block to slip, rotate unexpectedly, or be missed entirely.

The picking action itself was another source of error. Ideally, the robot gripper would grasp each block at or near its center. This would ensure stability during movement and accuracy during placement. In practice, however, the system often picked blocks slightly off-center. These errors were caused by small inaccuracies in the estimated block pose, as well as sensor noise in the depth data. As a result, the blocks were not held firmly or evenly, which made the stacking process more unstable.

Another limitation was the constrained physical space of the picking area. The blocks we used were relatively large compared to the area available for grasping, and the robot frequently encountered the edges of the workspace or hit predefined virtual walls meant to prevent collisions. This limited our flexibility in selecting blocks and occasionally caused planned grasps to be aborted due to safety boundaries.

All of these challenges contributed to stacking behavior that lacked precision. Misalignment from early rows led to structural tilt in later stages. Without feedback-based corrections, these small errors accumulated quickly.

VI. FUTURE WORK

While the current implementation of BlockBuilder successfully demonstrates autonomous picking and placement of color-classified blocks into structured patterns, several promising directions remain for future development. These enhancements aim to increase robustness, expand functionality, and improve adaptability to real-world scenarios.

A. Improved Perception and Lighting Robustness

One of the core limitations observed was the sensitivity of the HSV-based color detection pipeline to variations in lighting, shadows, and reflections. Future systems could benefit from the use of adaptive illumination or controlled lighting environments to ensure consistent input. Additionally, upgrading the vision pipeline to incorporate machine learning–based color classification, such as convolutional neural networks trained under varied lighting conditions, could significantly improve detection accuracy across different colors and materials.

B. Feedback-Driven Grasping and Placement

The current system performs grasping and placement using open-loop control, which limits its ability to recover from pose estimation errors. Future implementations could include real-time visual servoing and force sensing during manipulation. This would allow the robot to adaptively adjust its grip and placement based on sensor feedback, improving alignment and reducing failures such as toppling or clipping.

C. Pose Averaging and Filtering

Although a basic form of sampling for block position estimation was introduced, more advanced statistical techniques, such as Kalman filtering or Bayesian sensor fusion, could yield more accurate and reliable pose estimates. Temporal filtering would also help mitigate noise introduced by the depth sensor.

D. Adaptive Motion Planning and Workspace Scaling

The limited physical size of the picking area occasionally led to aborted grasps due to workspace constraints. Expanding the usable area, perhaps with multiple camera views or mobile robot bases, could increase operational flexibility. Integrating real-time obstacle avoidance and optimal motion planning algorithms, such as RRT* or CHOMP, would further enhance reliability in dynamic or cluttered environments.

E. Reset Automation and Environment Awareness

The current reset process still requires human intervention. A more autonomous routine could be developed by incorporating better environmental awareness, such as 3D scene reconstruction and free-space detection. With techniques borrowed from SLAM, the robot could identify viable reset positions and autonomously reorganize the workspace.

F. Heterogeneous Block Handling and Structural Reasoning

Future systems could support blocks of varying sizes and shapes, requiring more complex perception and manipulation. This would also introduce the need for reasoning about stability, weight distribution, and center of mass. Integrating physics simulations or learning-based predictors could help assess the structural integrity of a build before execution.

G. High-Level Instruction Parsing and Human-Robot Interaction

We envision a more intuitive interface for human operators. Users could describe desired patterns using natural language or submit reference images. By incorporating a language or vision-language model, the system could interpret such inputs into executable plans. This would allow for user-friendly customization and dynamic on-the-fly adjustments.

H. Multi-Robot Collaboration

In more advanced implementations, BlockBuilder could be extended to coordinate with multiple robots. For instance, one robot could focus on block retrieval while another handles placement. This would require reliable task scheduling, shared perception, and communication between agents to ensure synchronized actions.

I. Generalization to Industrial Use Cases

Finally, adapting this system for industrial applications would involve generalizing to diverse object types, ensuring robustness in unpredictable settings, and scaling throughput. Applications may include warehouse automation, part assembly, and custom fabrication, where structured robotic manipulation is essential.

VII. CONCLUSION

In this project, we developed BlockBuilder, an autonomous robotic system capable of selecting, grasping, and arranging colored blocks into a structured pattern derived from pixel art. The system was designed to integrate perception, planning, and manipulation in a cohesive workflow. Using a 7-DOF Franka Emika robotic arm and a RealSense depth camera, the robot was able to identify blocks in a scattered arrangement, classify their colors, determine their poses, and place them according to a predefined color sequence.

Our approach involved converting pixel art into a simplified color grid and executing a forward process that involved selecting a matching block, planning the motion to pick it up, and placing it at the appropriate location in the color wall. The system was built with a modular ROS framework, which allowed for separate development of perception, planning, and control components. In many trials, the system demonstrated the ability to complete full rows of the desired pattern and stack multiple layers to form a complete structure.

However, while the core functionality was achieved, several challenges limited the reliability and precision of the final output. Color detection was inconsistent, especially for certain shades such as blue and green, which were sometimes misclassified or not detected at all. These inconsistencies were often caused by uneven lighting or visual noise from reflections on the table surface. Additionally, the robot had difficulty detecting the rotation of blocks accurately, particularly because the blocks had few distinctive visual features and could appear in arbitrary orientations.

Grasping accuracy was also a common issue. The robot often picked up blocks off-center, which made placement unstable. As a result, the structure would begin to tilt as more blocks were stacked. This tilt became more pronounced with each new row. Another physical constraint was the limited picking area. Because our blocks were relatively large, the robot frequently approached the limits of its workspace or triggered virtual collision boundaries, which restricted its ability to retrieve certain blocks.

These challenges exposed the sensitivity of the system to small perception and control errors. The lack of feedback during grasping and placing meant that minor issues compounded over time, leading to misalignment in the final structure.

Despite these limitations, BlockBuilder serves as a meaningful demonstration of an integrated robotic manipulation pipeline. It shows that even a simple task like stacking colored blocks requires careful coordination between sensing, motion planning, and physical interaction. In future work, we plan to address these issues by improving lighting conditions, refining the perception pipeline, expanding the picking area, and introducing feedback-based corrections. These improvements would lead to a more robust and precise system capable of handling more complex patterns and taller structures.

REFERENCES

[1] A. Bhapkar, M. Pawaskar and A. Raut, "Color Sorting Pick Up and Drop Robotic Arm," 2024 IEEE 9th International Conference for Convergence in Technology (I2CT), Pune, India, 2024, pp. 1-6, doi: 10.1109/I2CT61223.2024.10544011.

[2] Lim Jie, Teoh Sen, Nor Maniha Ghani and Mohammad Fadhil Abas, "Automatic control of color sorting and pick/place of a 6- dof robot arm", Journal Européen des Systèmes Automatisés, vol. 54, pp. 435-443, 06 2021.

[3] N. Macias and J. Wen, "Vision guided robotic block stacking," 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems, Chicago, IL, USA, 2014, pp. 779-784, doi: 10.1109/IROS.2014.6942647.

[4] Z. Chen, T. Li, L. Qin, and Y. Jiang, "Vision-Guided Autonomous Block Loading in a Dual-Robot Collaborative Handling Framework," Journal of Construction Engineering and Management, vol. 151, no. 5, 03 2025.