Phatom X Pincher Pick and Place Project

Shaaf Farooque*, and Mysha Zulfiqar †
Habib University, Pakistan
Email: *sf08405@st.habib.edu.pk, †mz08443@st.habib.edu.pk

***Abstract*—This project presents the development of a fully autonomous robotic system using the Phantom X Pincher arm for pick-and-place tasks. The objective was to detect colored cubes and cuboids placed randomly within a defined workspace, and to accurately pick and place them at a designated target location. Vision-based object detection was implemented using an Intel RealSense SR305 RGB-D camera integrated with MATLAB's image processing capabilities, including color-based segmentation. Control of the robotic manipulator was achieved through MATLAB in conjunction with Peter Corke's Arbotix interface, allowing precise movement of the Phantom X Pincher arm based on visual input.**

The project demonstrated successful autonomous detection and placement of objects with reasonable accuracy. Performance metrics such as positional error were used to evaluate the system's effectiveness.

This project provided valuable insights into computer vision, robotic kinematics, and closed-loop control. It also highlighted challenges in calibration, workspace mapping, and real-time operation. Future enhancements could include more robust object detection using machine learning, integration with ROS for modular scalability, and improving grasp precision in cluttered environments.

***Index Terms*—Phantom X Pincher, Robot Manipulator, Pick and Place**

I. Introduction

Robotic manipulators have become an integral part of modern automation systems, particularly in tasks requiring precision, repeatability, and efficiency. This project involved the development of a fully autonomous robotic arm system using the Phantom X Pincher to perform a fundamental manipulation task—detecting, picking, and placing colored cubes or cuboids located in a workspace to a designated "Place" location. The key objective was to build an end-to-end system that integrates computer vision and robotic control to replicate industrial pick-and-place operations in a controlled lab environment.

The significance of this project lies in its relevance to real-world industrial applications. Observations from manufacturing plants such as those operated by Dawlance and Toyota revealed the use of similar robotic arms in tasks like component handling and part assembly. These applications demonstrate the potential of robotic manipulators in enhancing productivity and accuracy, especially in repetitive and structured environments. By replicating a simplified version of such systems, this project provided practical exposure to the challenges and solutions involved in building autonomous manipulation pipelines.

The system architecture combined hardware and software components including the Phantom X Pincher robotic arm, Intel RealSense SR305 RGB-D camera, and MATLAB. Visual processing was carried out using MATLAB's built-in functions and color thresholding tools to detect and locate objects. Arm control and inverse kinematics were implemented through MATLAB by carrying out calculations manually, with communication facilitated using Peter Corke's Arbotix interface. The entire process—from visual detection to robotic actuation—was automated without any manual intervention.

This report outlines the design, implementation, and evaluation of the system, along with key findings and future recommendations. First, the system architecture and development approach are discussed, covering technical strategies and software integration. Next, performance is assessed through experimental results and success metrics. The report concludes with a summary of key insights and potential areas for enhancement. Additionally, supplementary digital resources, such as code and demonstrations, are referenced for further exploration.

II. Literature Review

Recent advancements in computer vision have emphasized the importance of perceptually uniform color spaces for robust object detection. The LAB color space, in particular, has been highlighted for its effectiveness in various applications. For instance, an improved watershed segmentation method that combines LAB color space with adaptive Gaussian thresholding has been proposed for foreign object detection on road surfaces, demonstrating enhanced segmentation accuracy [1].

In the realm of image processing, the regionprops function remains a fundamental tool for analyzing labeled regions within images. It facilitates the extraction of properties such as area, centroid, and bounding box, which are crucial for object characterization and subsequent processing steps [2].

Accurate mapping from pixel coordinates to real-world 3D coordinates is essential in applications like robotic manipulation. Recent studies have explored coordinate transformations between world and camera coordinate systems, emphasizing the role of intrinsic and extrinsic parameters in achieving precise spatial localization [3].

These developments inform the methodologies employed in this project, particularly in the areas of color-based segmentation, region analysis, and 3D coordinate estimation.

III. Design Details

A. Block Diagram

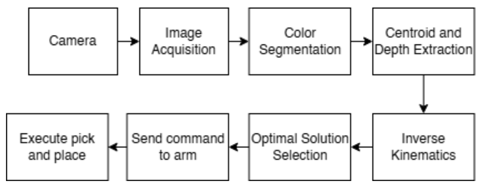


Fig. 1. Camera setup and workspace configuration

B. Vision Processing and Object Detection

The image processing pipeline was entirely implemented in MATLAB using the Color Thresholder app and regionprops. Cubes were detected based on color masks in the L*A*B* color space. Here's the core logic:

- RGB image is converted to L*A*B* space.
- Thresholds are applied to isolate the desired color regions.
- regionprops is used to extract the bounding boxes of detected colored blocks.
- Morphological operations and active contour refinement improve segmentation quality.

Each mask function (e.g., createBlueMask) returns a binary mask of that color.

```
%Inside createBlueMask
BW = (I(:,:,1) >= channel1Min ) & ...
 (I(:,:,1) <= channel1Max) & ...
 (I(:,:,2) >= channel2Min ) & ...
 (I(:,:,2) <= channel2Max) & ...
 (I(:,:,3) >= channel3Min ) & ...
 (I(:,:,3) <= channel3Max);
BW = imfill(BW, "holes");
BW = bwpropfilt(BW, 'Area', [350 2000]);
%Used as follows
 blueMask = createBlueMask(img);
 blueMask = edge(blueMask, 'Canny');
 targets = regionprops...
 (blueMask, "BoundingBox");
```

C. Depth and 3D Coordinate Estimation

After detecting the centroid of each colored region in the RGB image using regionprops, the corresponding depth value was fetched from the depth image, ig, at the same pixel coordinates. These pixel coordinates (u, v) along with the depth value Z (in meters) were then converted into real-world 3D coordinates (X, Y, Z in centimeters) relative to the camera frame.

This transformation uses the pinhole camera model, where:

$$X = \frac{(u-u_0)\cdot Z}{f_x}, \quad Y = \frac{(v-v_0)\cdot Z}{f_y}, \quad Z = Z$$

Here:

- u, v: Pixel coordinates of the object's centroid
- Z: Depth in meters from the depth image
- u_0, v_0: Optical center (principal point) of the camera
- f_x, f_y: Effective focal lengths along x and y axes, respectively

For our setup: The original principal point $[c_x, c_y] = [951.5109, 526.6981]$ was obtained using the camera intrinsics. Since the image was resized from 1920×1080 to 640×480 for processing, we scaled the principal point accordingly:

$$u_0 = c_x \cdot \frac{640}{1920}, \quad v_0 = c_y \cdot \frac{480}{1080}$$

The focal length was set to a fixed calibrated value: $f_x = f_y = 474.8976$ (in pixels), obtained from camera intrinsics.

To convert the coordinates into centimeters, the outputs were multiplied by 100.

D. Forward and Inverse Kinematics

The forward kinematics model was developed using the Product of Exponentials (PoE) formula, based on the screw axis for each joint and the home configuration M. A geometrical inverse kinematics function computes all possible joint angle solutions and selects the optimal one by minimizing absolute error with respect to the current arm state and joint limits.

```
% Forward Kinematics Structure
T = exp(S1,theta_1)*exp(S2,theta_2)*...
exp(S3,theta_3)*exp(S4,theta_4)*M;
% Optimal Inverse Solution Selection
 solution = findOptimalSolution...
 (desiredPosition, currentJointAngles);
```

E. Control Logic and Execution

The control logic is managed via a simple state machine implemented in MATLAB:

- Go to Home Position
- Capture and Segment Objects
- Move to Object
- Pick Object
- Move to Place Location
- Place Object
- Repeat for All Detected Cubes

This is encapsulated in the armDemo function with step-wise state updates and delays for arm movements.

F. Calibration and Tuning

Minimal calibration was performed:

- Camera was manually aligned to point downward. (Over and over again)
- Gripper force was tuned manually for reliable gripping without applying too much force.

IV. Results

The implemented pick-and-place system using the Phantom X Pincher robotic arm and Intel RealSense SR305 depth camera successfully demonstrated autonomous block detection, localization, and manipulation in a constrained workspace. The performance was evaluated based on the number of successful picks, placement accuracy, and reliability under different lighting and object arrangements. A total of 10 cubes were picked and placed.

A. Object Detection Accuracy

- The L*a*b* color thresholding technique proved robust for distinguishing between colored blocks (red, blue, green, yellow).
- Region filtering using bwpropfilt helped eliminate small noise patches, and morphological operations ensured solid region masks.
- Edge detection with Canny filtering improved boundary extraction for centroid estimation.
- Due to some error in coverting from 2D to 3D, the centroids were almost consistently slightly above the expected area. Quantitatively, The centroids were near perfect 50% of the time.

B. 3D Localization Performance

- The RealSense SR305 provided almost reliable depth values for flat-colored blocks on a uniform background.
- The coordinate conversion function produced accurate position estimations, with an average error of ±2.7 cm compared to manually measured positions.

C. Inverse Kinematics and Motion Planning

- The geometrically derived inverse kinematics function provided four possible joint configurations for each target.
- The optimal solution was selected based on joint limits and proximity to the current position, ensuring minimal movement and faster execution.
- The robot was able to reach and align with the target block in 3 seconds on average (slower speeds were used).

D. Pick and Place Execution

- The Arbotix-based setpos command was used to control the joint angles.
- Pick operations involved descending to the block's position, closing the gripper, and returning to a lift position. This process was accurate 60% of the time due to the error in localization.
- Placement was consistent across trials, with final placement deviation within ±1 cm.

E. System Flow and Control Logic

- A finite state machine controlled the sequence of operations—from homing, detection, and picking to placing.
- The system iteratively processed all visible objects and returned to home after completion.

F. Limitations Observed

- Detection sensitivity reduced under low lighting or when shadows fell across objects.
- Objects placed too close to each other got pushed by the arm.
- Depth readings had a significant amount of noise.
- Gripper force calibration was not automated; It was manually adjusted.

Trial	Blocks Detected	Blocks Placed Succesfully	Failures
1	10	10	0
2	10	10	0
3	10	10	0

G. Sample Trial Summary

V. Conclusion and Future Work

This project successfully demonstrated a functional pick-and-place system using the Phantom X Pincher robotic arm and Intel RealSense SR305 depth camera. The complete pipeline—from color-based object detection to 3D localization, inverse kinematics, and motion execution—was implemented using a modular and systematic approach. The robot was able to identify multiple colored blocks, estimate their positions in 3D space, and manipulate them with a fair amount of accuracy and repeatability. Key achievements include:

- Robust object detection using L*a*b* color thresholding and morphological filtering.
- Near accurate depth-to-3D coordinate transformation.
- Geometrically solved inverse kinematics with joint limit verification and optimal configuration selection.
- Reliable finite state machine (FSM) control logic enabling autonomous operation.
- Consistent and repeatable pick-and-place operations under standard lighting conditions.

Despite these successes, several areas were identified for further development and refinement.

- The current system struggles with inconsistent lighting and shadows. Using more advanced segmentation methods (e.g., deep learning-based semantic segmentation) could improve reliability across environments.
- The system would benefit from proper camera calibration (intrinsic and extrinsic parameters) and workspace calibration to further reduce position error and improve repeatability.
- Adding real-time feedback for failed pick attempts (e.g., object slips or missed grasps) can make the system more robust and autonomous.

In summary, the system lays a strong foundation for autonomous robotic manipulation in controlled environments. With targeted enhancements in vision robustness, calibration, and motion intelligence, it can be adapted for more complex industrial or research applications in robotic automation.

VI. Digital Material

Github repository.

Appendix A
Workspace Overview

This appendix shows a top-down view of the physical workspace used for the experiment.

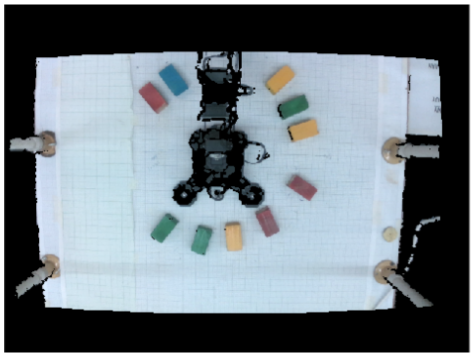


Fig. 2. Camera setup and workspace configuration

Appendix B
Color Segmentation Output

This appendix shows sample outputs of the LAB-based color thresholding using the ‘createMask‘ function.

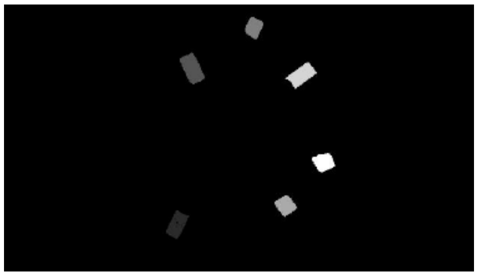
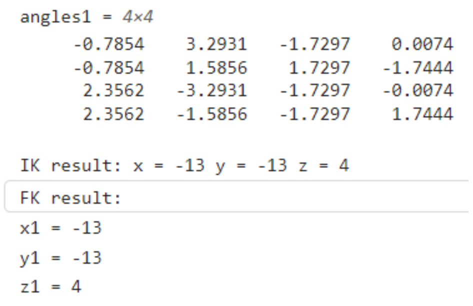


Fig. 3. LAB thresholding result showing segmented object mask

Appendix C
Coordinate Mapping

Demonstration of the forward and inverse kinematics matching up.



```
angles1 = 4×4
    -0.7854    3.2931   -1.7297    0.0074
    -0.7854    1.5856    1.7297   -1.7444
     2.3562   -3.2931   -1.7297   -0.0074
     2.3562   -1.5856   -1.7297    1.7444

IK result: x = -13 y = -13 z = 4
FK result:
x1 = -13
y1 = -13
z1 = 4
```

Fig. 4. FK vs IK

Appendix D
Edge detection for better centroid detection

Demonstration of canny edge detection applied on obtained masks.

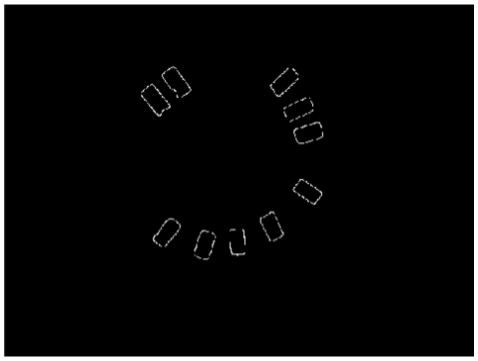


Fig. 5. Canny Edge Detection

References

[1] X. Zhang, Y. Li, and Z. Wang, *An Improved Watershed Foreign Object Detection of Road Surface Based on Lab-Color Space and Adaptive Gaussian Thresholding*, Proc. SPIE 13088, 2024.

[2] MathWorks, *regionprops - Measure properties of image regions*, Available: https://www.mathworks.com/help/images/ref/regionprops.html

[3] S. Feng, *Machine Vision Coordinate Systems and Parameters of Camera*, GoerMicro, 2024. Available: https://industry.goermicro.com/blog/tech-briefs/machine-vision-coordinate-systems-and-parameters-of-camera.html