IJRCS

International Journal of Robotics and Control Systems
Vol. 6, No. 2, 2026, pp. 1041-1059
ISSN 2775-2658
http://pubs2.ascee.org/index.php/ijrcs

ASCEE

Design and Implementation of an Eye-in-Hand Vision-Guided QR Sorting System on Dobot MG400

Kriangkrai Waiyakan [a,1], Tossapol Jangnoi [b,2,*], Viroch Sukontanakarn [b,3], Thewin Sakunbunyong [b,4]

[a] Faculty of Engineering, Thaksin University, Phatthalung, Thailand
[b] Faculty of Engineering, Rajamangala University of Technology Isan, Khon Kaen Campus, Khon Kaen, Thailand
[1] kriangkrai.w@tsu.ac.th; [2] tossapol.ja@rmuti.ac.th; [3] viroch.su@rmuti.ac.th; [4] thewin.sa@rmuti.ac.th
*Corresponding Author

ARTICLE INFO

Article history
Received January 12, 2026
Revised February 20, 2026
Accepted April 06, 2026

Keywords
Vision-Guided Robotic;
Dobot MG400;
Eye-in-Hand Vision;
QR Code Detection;
Python Programming;
Pick-and-Place

ABSTRACT

This research presents a self-developed programming framework using Python and OpenCV, without manufacturer-provided software, to control a Dobot MG400 for automated pick-and-place tasks. The system handles mock workpieces labeled with QR code stickers. Custom algorithms were developed for QR code detection and edge-preserving image enhancement, while manual planar homography calibration was applied to accurately convert image-plane coordinates into real-world robot coordinates. An eye-in-hand camera mounted on the robot captures images and sends them to a computer for processing. The calculated coordinates are transmitted back to the robot to perform pick-and-place operations at predefined locations. Experimental validation included 22 cycles under varying object positions and lighting conditions, achieving a 100% success rate. The average QR code detection confidence reached 99.34% ± 0.32%, demonstrating highly consistent identification. The average positioning error was 0.20 ± 0.12 mm, with deviations between 0.057 mm and 0.397 mm, indicating sub-millimeter accuracy and strong repeatability. The average cycle time was approximately 6.8 seconds per operation. Real-time trajectory visualization enabled quantitative motion analysis. Results confirm the system's reliability, precision, and scalability for modern manufacturing applications.

1. Introduction

Computer vision–enabled robotic manipulation is widely recognized as a foundational technology for intelligent manufacturing systems. Recent studies highlight its impact in areas such as automated e-waste disassembly [1], AI-oriented robotic architectures [2], construction robotics with embedded vision [3], and adaptive human–robot collaboration [4]. Within the Industry 4.0 framework, smart factories emphasize interoperability, flexibility, and strategic technology integration [5]. Advanced AI modeling methods further strengthen automation capabilities [6], while interconnected industrial IoT infrastructures are accelerating the transition toward Industry 5.0 ecosystems [7]. Moreover, deep transfer learning and data-centric methodologies continue to drive rapid progress in industrial robotics [8]. To mitigate vendor lock-in and improve system transparency, researchers increasingly explore open-source machine vision solutions [9] and advanced perception techniques for manufacturing [10]. Vision-based action recognition enhances robotic autonomy [11], and open-hardware platforms combined with affordable computing resources facilitate reproducible laboratory-scale experimentation [12].

QR-code-based identification has emerged as a practical solution for real-time encoding and object recognition [13], particularly in heterogeneous sorting environments [14]. Eye-in-hand configurations, where cameras are mounted directly on the end-effector, enhance robustness and spatial consistency [15]. Accurate coordinate transformation is achieved through integrated kinematic and vision-based calibration strategies [16]. Reinforcement learning methods further contribute to adaptive control and intelligent decision-making in smart factories [17], while modular open-source architectures promote scalable automation frameworks [18]. Efficient trajectory generation and motion optimization remain critical for industrial manipulators [19], especially in compact and energy-aware automation platforms [20], as emphasized in recent smart manufacturing surveys [21]. Learning-based pick-and-place optimization has also demonstrated measurable performance gains [22]. Educational robotic platforms, including augmented-reality-assisted systems [23] and Dobot-based vision applications [24], provide accessible experimental infrastructures, complemented by advanced Cartesian trajectory planning [25] and deep reinforcement learning simulation frameworks [26].

Motivated by these technological advancements, this study introduces a laboratory-scale computer vision–guided robotic sorting system developed entirely within a university setting. The platform integrates a Dobot MG400 manipulator configured with an eye-in-hand camera. All perception modules-image acquisition, preprocessing, QR detection, decoding, and planar pose estimation-are implemented in Python using OpenCV and Pyzbar. Camera calibration combined with homography-based mapping enables transformation from image coordinates to Cartesian robot coordinates, allowing accurate pick-and-place of randomly distributed QR-labeled objects.

In contrast to conventional closed-loop visual servoing approaches such as IBVS and PBVS, which depend on continuous visual feedback and velocity-level control, the proposed framework adopts a calibrated open-loop visual guidance methodology. The operational workflow includes image capture at a fixed observation pose, QR localization, coordinate mapping, and execution of a predefined Cartesian trajectory. This strategy aligns with the position-level SDK interface of the Dobot MG400 and reduces computational overhead while maintaining adequate precision in structured laboratory conditions. Beyond object localization, the system incorporates real-time trajectory streaming and quantitative motion analysis, evaluating velocity characteristics, acceleration smoothness, jerk behavior, positional deviation, and overall cycle time. Experimental validation within a smart manufacturing testbed confirms that a vendor-independent, fully open-source architecture can deliver reliable and precise QR-based robotic sorting while remaining extensible for future AI-driven optimization.

2. Material Setup

2.1. Hardware Configuration

The system consists of a computer running the Windows 10 operating system with Python and OpenCV installed, which acquires visual data from a USB camera via a USB interface [27]. The camera is mounted on the end effector of the Dobot MG400 robotic arm in conjunction with a vacuum gripper [28]. The vision system processes the captured images to detect workpieces and determine their positions for pick-and-place operations. Based on the processing results, the robot transfers the workpieces from the designated placement area to the assigned storage locations (Box 1, Box 2, Box 3, or Box 4). Control commands are transmitted from the computer to an Arduino board to actuate a solenoid valve, which regulates compressed air from an air reservoir to enable or disable the vacuum gripper [29], [30] for secure object handling, as shown in Fig. 1.

Connecting a computer to the Dobot MG400 using Python is done through a TCP/IP network connection [31]. The Dobot MG400 uses a fixed IP address of 192.168.1.6, while the computer must be configured with an IP address in the same subnet, such as 192.168.1.10 and a subnet mask of 255.255.255.0. After connecting the Ethernet cable, the network connection can be verified by pinging 192.168.1.6 from the computer. In Python, the socket library is used to establish a TCP

connection with the robot controller [32]. The dashboard Port 29999 is used for sending system control and status commands. The move Port 30003 is used to send motion commands to control robot movement. The feedback Port 30004 provides real-time robot status and feedback data. Through these ports, Python programs can control and monitor the Dobot MG400 reliably over the network.

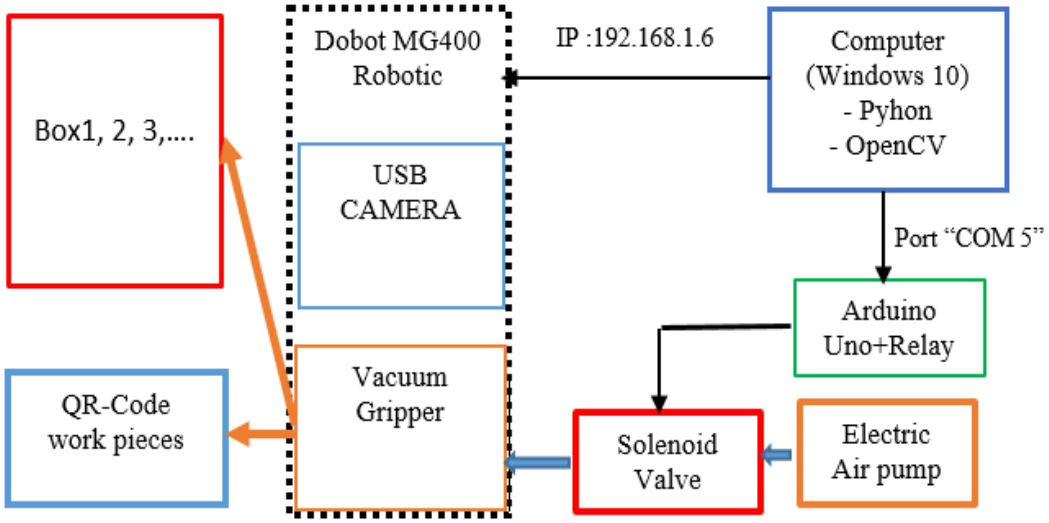

Fig. 1. Eye-in-hand calibration setup

2.2. Camera Calibration and Lens Distortion Compensation

In an eye-to-hand vision system, images captured by a real camera deviate from the ideal pinhole camera model due to lens distortion. Therefore, prior to homography estimation, intrinsic camera calibration and image distortion compensation must be performed to ensure geometric accuracy.

The ideal pinhole camera model can be expressed as in (1).

$$S\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = K[R \quad t]\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{1}$$

Where S is a scale factor, $R \in R^{3x3}$ and $t \in R^{3x1}$ denote the rotation matrix and translation vector, respectively, and K represents the intrinsic camera matrix given in (2).

$$K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{2}$$

Here, f_x and f_y denote the focal lengths in pixel units, and (c_x,c_y) represents the principal point. However, real lenses introduce radial and tangential distortions. Let in (3).

$$r^2 = u^2 + v^2 \tag{3}$$

Radial distortion is define in (4), (5) as follow.

$$u_r = u(1 + k_1 r^2 + k_2 r^4) \tag{4}$$

$$v_r = v(1 + k_1 r^2 + k_2 r^4) \tag{5}$$

Tangential distortion is define in (6), (7) as follow.

$$u_d = u_r + 2p_1 uv + p_2(r^2 + 2u^2) \tag{6}$$

$$v_d = v_r + p_1(r^2 + 2u^2) + 2p_2uv \tag{7}$$

Where k_1, k_2 are the radial distortion coefficients, and p_1, p_2 are the tangential distortion coefficients. Before estimating the homography matrix, all captured images are undistorted to obtain corrected image coordinates (u, v). These corrected coordinates are then used to establish correspondences with the robot workspace coordinates (X, Y) for homography computation. This preprocessing step ensures that the planar projective transformation assumption is better satisfied and significantly improves positioning accuracy, particularly near the image boundaries where distortion effects are more pronounced.

Eye-in-hand calibration aims to determine the rigid transformation between a camera mounted on a robot end-effector and the robot tool coordinate frame [33]-[35]. This calibration is essential for vision-guided manipulation, where visual information must be accurately mapped to robot motions. The relationship between the robot motion and the camera motion [36] is commonly formulated using the eye-in-hand calibration as in (8).

$$A_iX = XB_i \tag{8}$$

Where A_i represents the relative motion of the robot end-effector, B_i represents the corresponding motion of the camera, and X is the unknown transformation between the camera and the end-effector. Each transformation is expressed as a homogeneous transformation matrix [37] as in (9).

$$T = \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix} \tag{9}$$

Where R denoting the rotation matrix and t is the translation vector. By collecting multiple motion pairs, optimization or closed-form methods are applied to estimate X. Accurate eye-in-hand calibration significantly improves precision, robustness, and repeatability in robotic vision applications. Fig. 2 shows an eye-in-hand calibration setup, where a camera attached to the robot end-effector observes a calibration target to estimate the transformation between the camera and tool frames [38].

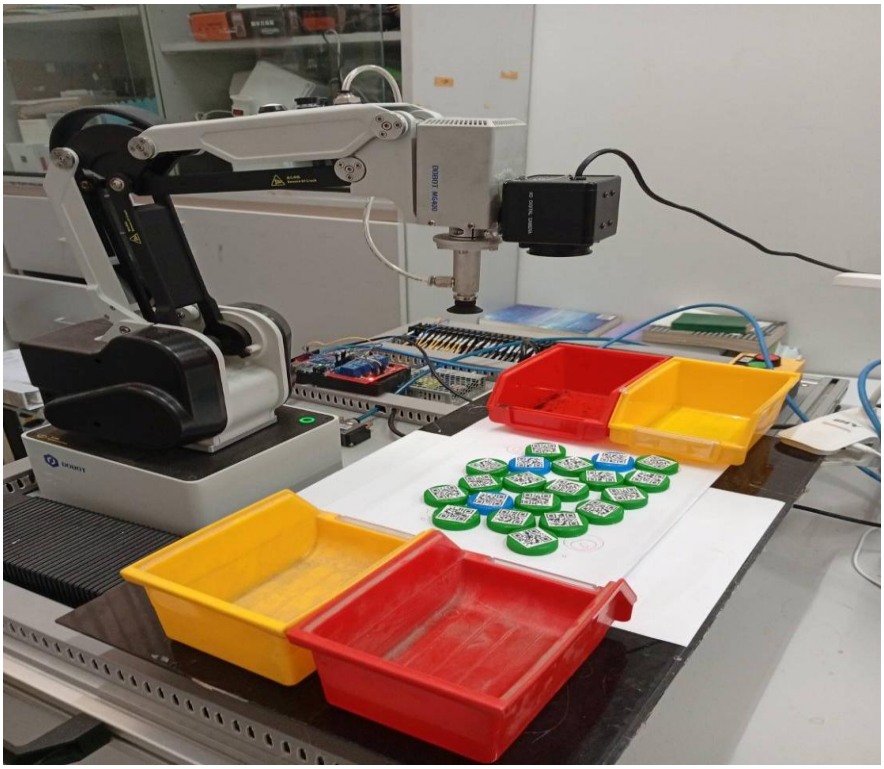

Fig. 2. Eye-in-hand calibration setup

2.3. QR Code Configuration for Robotic Picking Systems

The use of QR codes on products enhances the efficiency and accuracy of robotic picking systems in manufacturing environments [39], [40]. QR codes store essential information, such as product identifiers, location data, and handling instructions, enabling automated identification. When integrated with machine vision systems, robots can accurately scan and decode QR codes to perform precise pick-and-place operations. QR code generation and detection are implemented using tools such as the Python QR Code library, Zint, ZXing, and OpenCV, and are applied to the Dobot MG400 robotic arm for system control and vision processing. This approach reduces human error, increases picking speed, and supports automation in smart manufacturing systems [41]-[43].

3. Methodology

This experimental procedure kicks off with a critical calibration phase, aligning the eye-in-hand camera mounted on the robot's arm with both the workspace and the specific QR-coded workpieces. Once the spatial parameters are established through this calibration, the system transitions to fully autonomous sorting operations [44]-[46]. This involves the robot dynamically identifying and picking items scattered across the surface, then precisely transporting them to their designated drop-off zones based on the decoded information.

3.1. Homography-Based Transformation Between Image and Robot Coordinates

In this study, a homography-based transformation [47] is employed to establish the relationship between image coordinates and the corresponding coordinates in the robot workspace. The image coordinates are defined in the pixel domain as(u, v), while the MG400 robot coordinates are defined in the world coordinate system as (X, Y) [48]-[50]. The Z-coordinate of the robot is assumed to be constant and is therefore not involved in the homography transformation. Consequently, the mapping between the image plane and the robot workspace can be expressed using a projective transformation (homography) [51], as shown in (10).

$$\begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} = H \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \tag{10}$$

Where H is the 3×3 homography matrix defined as in (11).

$$H = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & 1 \end{bmatrix} \tag{11}$$

From this formulation, the relationships between the image coordinates (u, v) and the robot coordinates (X, Y) can be written explicitly as in (12), and (13).

$$X = \frac{h_{11}u + h_{12}v + h_{13}}{h_{31}u + h_{22}v + 1} \tag{12}$$

$$Y = \frac{h_{21}u + h_{22}v + h_{23}}{h_{31}u + h_{22}v + 1} \tag{13}$$

By rearranging these equations, two linear equations are obtained for each corresponding point as in (14), and (15).

$$uh_{11} + vh_{12} + h_{13} - Xuh_{31} - Xvh_{32} = X \tag{14}$$

$$uh_{21} + vh_{22} + h_{23} - Yuh_{31} - Yvh_{32} = Y \tag{15}$$

The unknown parameters of the homography matrix are in (16).

$$h_{11}, h_{12}, h_{13}, h_{21}, h_{22}, h_{23}, h_{31}, h_{32} \tag{16}$$

Fig. 3 illustrates a specialized procedure for hand-assisted calibration of a homography-based robotic system. The figure shows a robotic arm or positioning device being manually calibrated using a human hand. The operator guides the robot's end-effector to touch specific reference points marked on a sheet of paper. These known reference points are used to establish correspondences between the robot's coordinate system and the camera (image) coordinate system. By touching multiple reference points, the system can compute a homography transformation, which maps positions from the 2D image plane to the robot's physical workspace. This manual, hand-guided calibration process improves system accuracy prior to automated operation, ensuring that the robot can precisely move to target locations detected by the vision system. In summary, the figure demonstrates a hand-assisted homography calibration method, which is commonly employed in vision-guided robotics and experimental robotic setups.

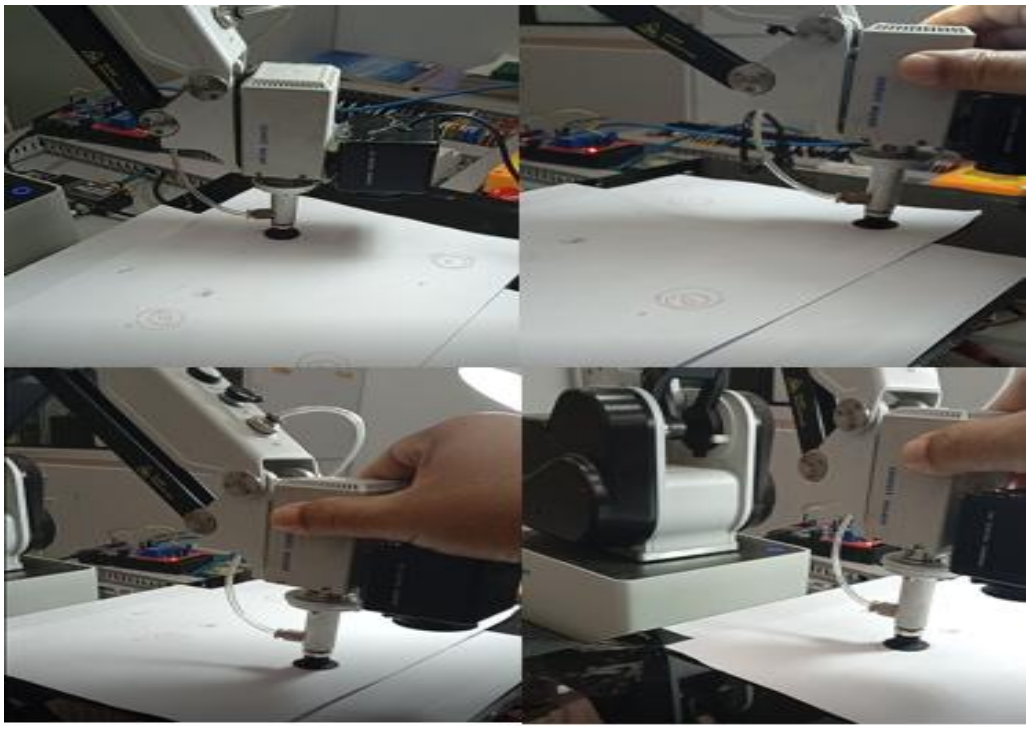

Fig. 3. Manual homography calibration at four points

Table 1 records four strategic calibration points linking the (u, v) pixel domain to the MG400's physical workspace coordinates. These empirical pairs establish the geometric foundation for the homography matrix, effectively bridging the gap between digital vision and robotic motion [52]. Utilizing these specific data points ensures the spatial precision necessary for reliable, autonomous pick-and-place operations [53]-[55].

Table 1. Coordinate mapping for homography calibration

Point	(u, v) Pixel	(X, Y) Robot (mm)
1	(241, 581)	(250.6219, 117.6952)
2	(206, 148)	(358.6468, 134.9675)
3	(778, 206)	(356.9252, -10.6901)
4	(778, 585)	(260.3982, -18.2163)

To valid these parameters, multiple pairs of corresponding points between the image and robot coordinate systems are used. For example, at Point 1, the image coordinates are (u, v) = (241, 581),

and the corresponding robot coordinates are (X, Y) = (250.6219, 117.6952). Substituting these values as in (17) and (18).

$$241h_{11} + 581h_{12} + h_{13} - 250.6219(241h_{31} - 581h_{32}) = 250.6219 \quad (17)$$

$$241h_{21} + 581h_{22} + h_{23} - 117.6952(241h_{31} - 581h_{32} = 117.6952 \quad (18)$$

The same procedure is applied to the remaining calibration points (Points 2 to 4). With four point correspondences, a total of eight linear equations are obtained, which can be expressed in matrix form as in (19).

$$Ah = b \quad (19)$$

Where h is the vector of the unknown homography parameters. The system is solved using the least squares method [56] as in (20).

$$h = (A^T A)^{-1} A^T b \quad (20)$$

After computation using a numerical program, the resulting homography matrix is obtained as in (21).

$$\begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} = \begin{bmatrix} 0.0135 & -0.2496 & 391.0283 \\ -0.2512 & -0.0195 & 188.9269 \\ 0.0000 & 0.0000 & 1.00000 \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \quad (21)$$

This homography matrix provides the fundamental transformation equation for converting image coordinates into robot workspace coordinates, enabling the MG400 robot to accurately move based on visual information obtained from the camera.

3.2. Mathematical Modeling Equations of Control System

The intelligence of the system arises from mathematical modeling and deterministic algorithms, not from data-driven learning models.

1. Planar Homography Transformation

The most critical mathematical part of your project is the mapping between the image plane (pixels) and the robot plane (millimeters). This is a Projective Transformation. The relationship is defined by the matrix H in (22).

$$\begin{bmatrix} x_{robot} \cdot \omega \\ y_{robot} \cdot \omega \\ \omega \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & 1 \end{bmatrix} \begin{bmatrix} u_{pixel} \\ v_{pixel} \\ 1 \end{bmatrix} \quad (22)$$

Where (u_{pixel}, v_{pixel}) are the coordinates of a point in the image (in pixels), (x_{robot}, y_{robot}) are the coordinates of the corresponding point in the robot or world coordinate system, matrix $H_{(3\times3)}$ is the homography matrix that defines the geometric transformation between two planes [57] (e.g., the image plane and the ground plane), and ω is a scale factor introduced by the use of homogeneous coordinates [58], [59]. The implementation code then uses cv2.findHomography() to solve for the eight degrees of freedom in matrix H using the four points clicked during calibration.

2. Centroid Geometry (Feature Localization)

To accurately determine the robot's pick location, the system computes the centroid of the detected QR code polygon. If the QR code vertices are denoted as are P_1, P_2, P_3, P_4, the centroid coordinates (C_x, C_y) are calculated [60] as shown in (23).

$$C_x = \frac{1}{n}\sum_{i=0}^{n} x_i \, , \, C_y = \frac{1}{n}\sum_{i=0}^{n} y_i \quad (23)$$

This provides a single coordinate point (u, v) to pass into the homography equation.

3. Image Enhancement (Domain Transform Filtering)

It is a fundamental concept used in computer vision and medical image analysis, particularly for tasks like interactive image segmentation. In camera enhance_frame, the cv2.detailEnhance function applies an edge-preserving filter. It relies on the Domain Transform (derived from the Geodesic distance) [61] in (24).

$$D(x,y) = \int_{\Omega} \sqrt{1 + \sigma_s^2 \cdot \|\nabla I\|^2}\, dt \tag{24}$$

Where $D(x,y)$ is the geodesic distance between two points x and y, Ω is the path or curve connecting the two points, ∇I is the Image Gradient. It measures the rate of change in intensity (brightness/color). Large values typically represent edges or boundaries in the image, $\|\nabla I\|^2$ is the squared magnitude of the gradient, σ^2 is a scaling parameter (weight) that controls how much the image content (the edges) influences the distance calculation, and dt is the infinitesimal step along the path. This increases the Signal-to-Noise Ratio (SNR) for the pyzbar decoder, allowing it to read QR codes even in poor lighting or with slight motion blur [62].

4. Euclidean Distance for Duplicate Prevention

The code uses a logic to avoid re-processing the same QR code if it moves slightly. This is implicitly handled by your qr_unique_id: f" {data}_{cx//20}{cy//20}". Mathematically, this is a Grid-based Quantization (a form of spatial hashing) in (25).

$$Cell(x,y) = \left(\left|\frac{x}{\delta}\right|, \left|\frac{y}{\delta}\right|\right) \tag{25}$$

Where $\delta = 20$ pixels. If a QR stays within the same 20 x20 pixel block, the AI treats it as the same object.

5. Movement Kinematics (Linear Interpolation)

When we call move.MovL, the robot controller executes a Linear Interpolation (LERP) [63] in Cartesian space to ensure the end-effector travels in a straight line in (26).

$$P(t) = (1-t)P_{start} + tP_{end} \text{ for } \; t \in [0,1] \tag{26}$$

The speed_factor adjusts the derivative $\frac{dP}{dt}$, controlling the velocity profile (usually a trapezoidal or S-curve profile) to prevent mechanical vibration during "Pick" and "Drop" phases.

3.3. Select and Define of QR Code Stricker and Workpieces

This process begins with the selection and definition of unique QR code stickers assigned to each individual workpiece. Four distinct identifiers, ranging from EMC_RMUTI1 to EMC_RMUTI4, are generated to ensure precise tracking of each item. These labels function as digital signatures, enabling the camera system to recognize and differentiate components in real time. By integrating these QR codes, the system can automatically associate physical workpieces with their corresponding records in the central database. This structured identification approach significantly enhances the accuracy and reliability of automated monitoring and quality control processes, as illustrated in Fig. 4.

3.4. Program System Architecture

The system starts with a Tkinter-based user interface [64] that allows the user to send commands to different subsystems. The vision engine captures camera input using OpenCV and processes pixel data from the environment. This visual data can be enhanced using vision based filters and decoded for QR codes. The processed pixel coordinates are then passed to the transformation module, where

a homography matrix is applied to convert pixel coordinates into real-world robot coordinates. The resulting robot coordinates are forwarded to the motion control system. Motion control uses the Dobot API [65] to calculate movement commands such as position and speed. These commands are sent to an Arduino, which handles low-level on/off control and provides real-time position feedback. Throughout operation, the system continuously feeds positional data back for monitoring and adjustment. Finally, the data analytics and reporting modules export results to Excel, plot robot paths, and visualize performance trends, as shown in Fig. 5.

Fig. 4. Definition and assignment of unique QR code identifiers for workpieces

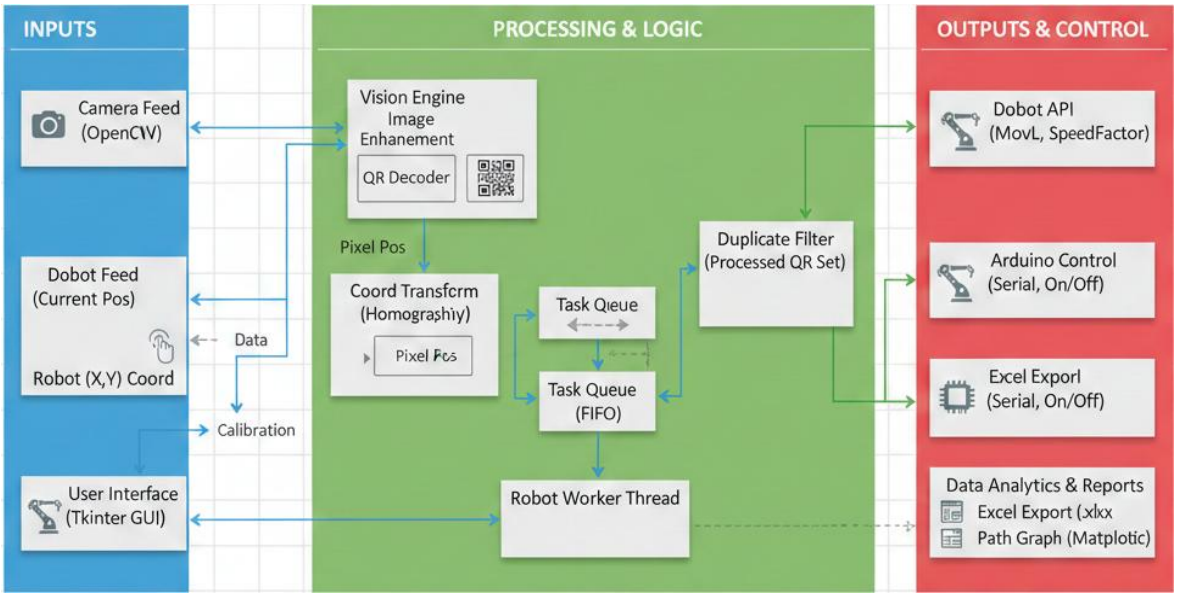

Fig. 5. Workflow diagram of the Dobot robotic system integrated with image processing (structural design by the researcher, and diagram arrangement using high-level image processing tools)

3.5. Computer Enhanced Robotic Pick-and-Place System

The system begins with an initialization phase, during which global variables are set, communication is established with the Dobot and Arduino, and the robot returns to its home position while launching specialized threads for feedback, task execution, and vision. The core logic relies on the camera thread, which captures live frames, utilizes the PyZbar library to detect QR codes. Once a code is identified, the system calculates its center, prevents duplicate detections, and maps pixel data to real-world coordinates using a homography matrix [66]. These coordinates are then sent to the worker thread, which manages the execution logic by popping tasks from a queue to perform precise pick-and-place movements-descending to pick an item with an Arduino-controlled gripper and moving it to a designated drop zone based on the QR label. Finally, the process is supported by calibration logic to ensure spatial accuracy and an analytics and visualization component that exports performance logs to Excel and generates movement path plots using Matplotlib for post-operation analysis. The algorithmic process is as follows:

1. Initialization Phase
    a. Step 1.1: Initialize global variables (coordinates, path buffers, and state flags).

b. Step 1.2: Establish communication with the Dobot Robot (Dashboard, Move, and Feed ports) and the Arduino (for the gripper/suction control).
c. Step 1.3: Move the robot to the predefined HOME_POSE.
d. Step 1.4: Launch background threads:
   - Feedback Thread: Constantly updates the robot's real-time position.
   - Worker Thread: Monitors the task queue and executes physical movements.
   - Camera Thread: Handles the live video stream.

2. Vision & Detection Logic (Camera Thread)
   a. Step 2.1: Capture live frames from the camera.
   b. Step 2.2: If a Region of Interest (ROI) is defined, crop the frame to that area.
   c. Step 2.3: Apply Filter enhancement (detailEnhance) to improve QR code readability.
   d. Step 2.4: Scan for QR codes using the PyZbar library.
   e. Step 2.5: For every detected QR:
      - Calculate the center pixel (x,y).
      - Generate a unique_id to prevent duplicate picking of the same object.
      - If Auto Mode is ON and the robot is idle, convert pixels to robot coordinates using the Homography Matrix.
      - Add the coordinates and QR data to the Task Queue.
3. Execution Logic (Worker Thread)
   a. Step 3.1: Check if the task_queue has pending items.
   b. Step 3.2: Pop the first task (Pick coordinates + QR label).
   c. Step 3.3: Picking Phase:
      - Move linearly to a "Lift" position above the object.
      - Descend to the fixed_Z depth.
      - Trigger ArduinoOn() to activate the vacuum gripper.
   d. Step 3.4: Dropping Phase:
      - Lift the object and move to the specific Drop Zone associated with the QR text.
      - Lower the arm and trigger ArduinoOff() to release the object.
   e. Step 3.5: Return to Home and log the operation data (timestamp, accuracy, and path).
4. Calibration Logic (Homography)
   a. Step 4.1: Manual Calibration: The user clicks 4 points on the camera view while the robot is physically at those 4 corresponding locations.
   b. Step 4.2: Use cv2.findHomography to create a transformation matrix H that maps image pixels to real-world millimeters.
5. Analytics & Visualization
   a. Data Export: Save log_data (containing confidence levels and coordinates) into an Excel file using Pandas.
   b. Path Analysis: Use Matplotlib to plot:

- Red Lines: Movement toward the pick point.
- Green Lines: Movement toward the drop zone.
- Markers: Specific points where items were picked or placed.

4. Results and Discussion

This section presents the experimental results and performance analysis of the automated robotic pick-and-place system that integrates machine vision and artificial intelligence. The system is implemented using a Dobot MG400 robotic arm combined with an eye-in-hand vision configuration and a homography-based coordinate transformation approach, under the hardware and software setup described in the materials setup and methodology sections. The experiments were designed to evaluate the accuracy of camera calibration, the precision of image-to-robot coordinate mapping, and the system's capability to reliably detect QR codes and autonomously perform pick-and-place operations to predefined target locations.

4.1. Experiment Homography Calibration

Fig. 6 shows the robot camera calibration and coordinate mapping process, including calibration parameters and reference points used to convert image coordinates into real-world robot coordinates. The interface displays the workspace, where the blue polygon marks the calibrated working area. Red points and real-time X, Y, and Z values indicate active vision–robot communication.

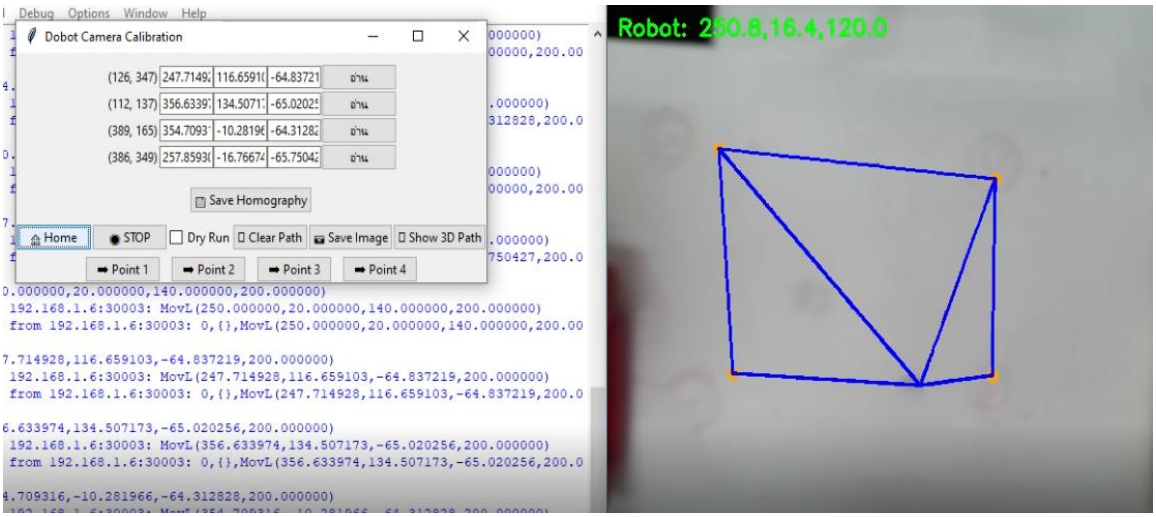

Fig. 6. Recording four-point coordinates into the program

The graph in Fig. 7 shows a 3D robot trajectory plotted in X, Y, and Z coordinates, measured in millimeters. The blue line represents the path the robot follows through space, forming a pyramidal-like structure with curved transitions. The green marker indicates the starting position of the robot, while the red marker shows the ending position at a higher Z value. Overall, the trajectory demonstrates smooth motion with multiple directional changes in three-dimensional space.

4.2. Experiment on QR-Labeled Object Sorting

The robotic arm system, as illustrated in Fig. 8, employs an automated pick-and-place mechanism to sort QR-labeled objects. Initially, the system detects and localizes individual objects on the workspace using predefined vision parameters. The robotic arm then grasps each object with a vacuum gripper while maintaining positional accuracy and operational stability. Based on the object classification obtained from QR code identification, the control system determines the appropriate destination container. The robotic arm subsequently transfers and releases the object into the corresponding colored bin. This process is continuously repeated, demonstrating efficient and systematic operation suitable for both industrial and educational automation applications.

International Journal of Robotics and Control Systems
Vol. 6, No. 2, 2026, pp. 1041-1059
ISSN 2775-2658

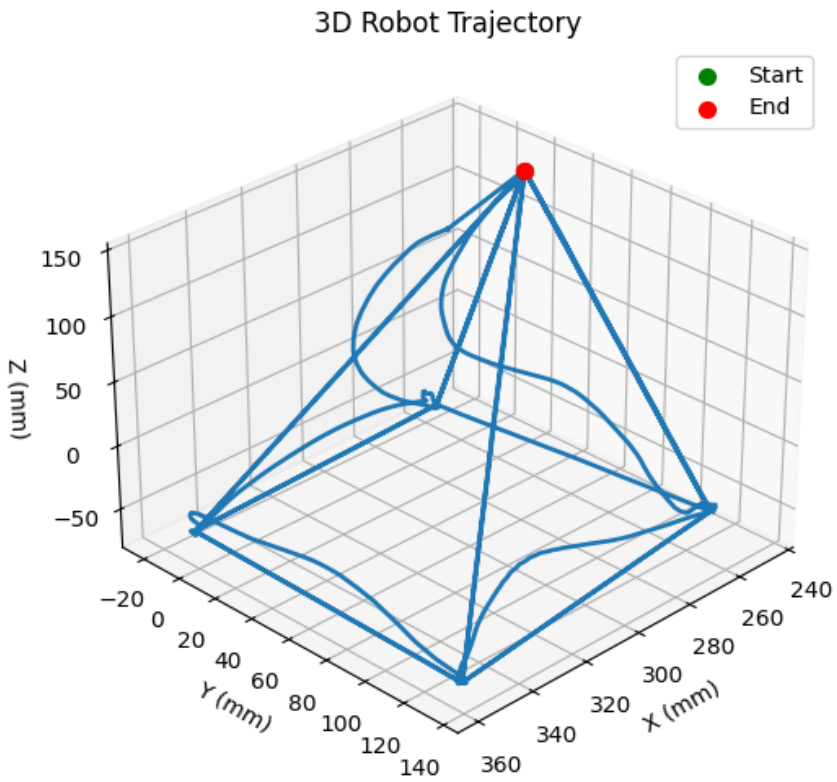

Fig. 7. 3D trajectory of the vacuum gripper based on homography coordinates from four points programmed into the system

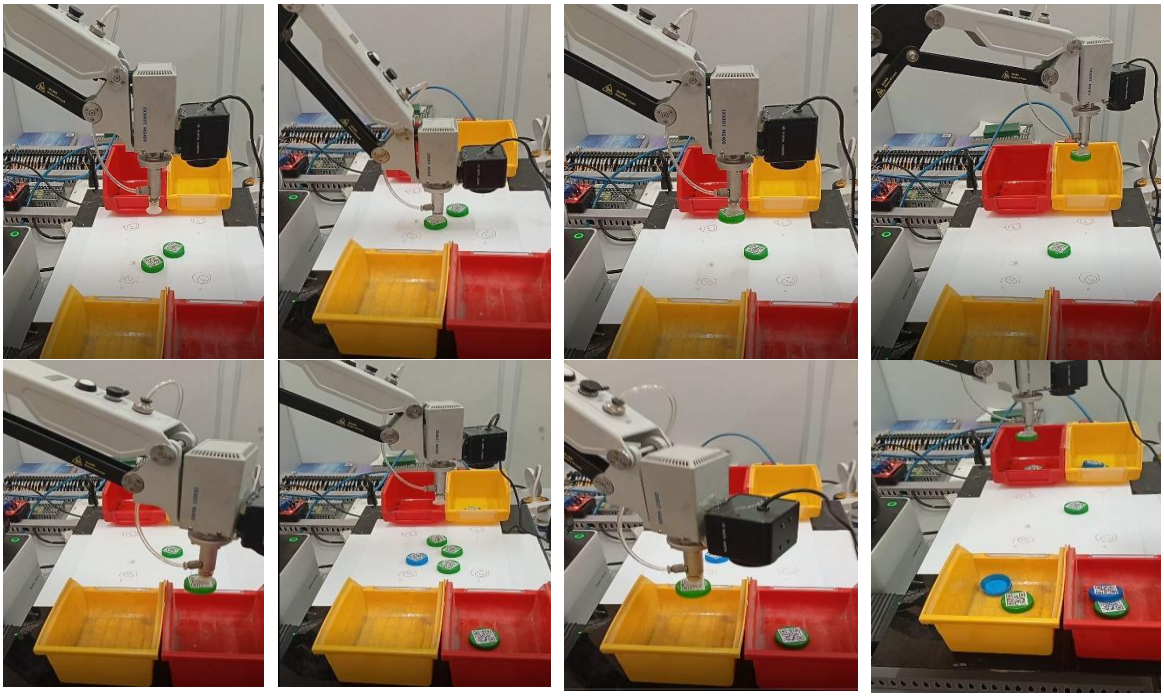

Fig. 8. Pick-and-place experiment using QR-code-labeled objects

The image in Fig. 9 illustrates a desktop-based computer vision system interface integrating a live camera feed, control panels, and a system log. The interface demonstrates successful QR code detection, where identified targets are enclosed within bounding boxes and labeled accordingly. The detected objects are processed in real time, with corresponding coordinate data and system status messages displayed in the console window.

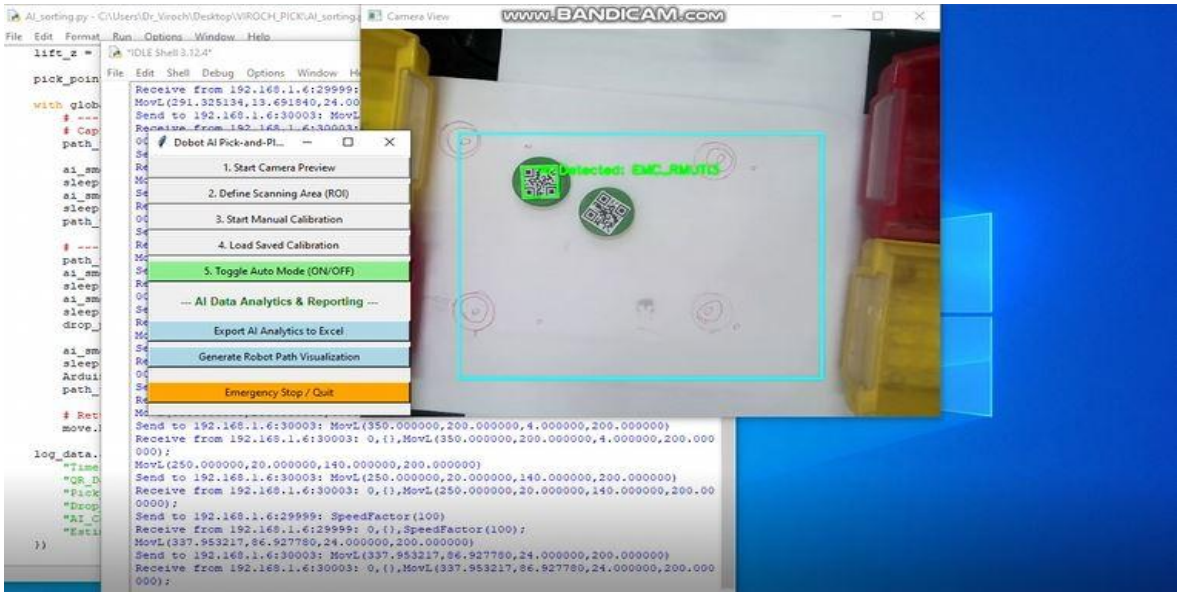

Fig. 9. Recording the coordinates of four points in the program

This graph in Fig. 10 illustrates the robot operation path analysis, where the X and Y axes represent movement coordinates in millimeters. The red points indicate pick locations that the robot must visit to collect items, while the green points represent drop locations for placing the items. The blue star marks the robot's current position at the start of the operation. The red dashed lines show the robot's travel path from its current position to the pick locations, and the green lines indicate the path from the pick locations to the drop locations. The black line represents the overall movement path of the robot during the operation cycle.

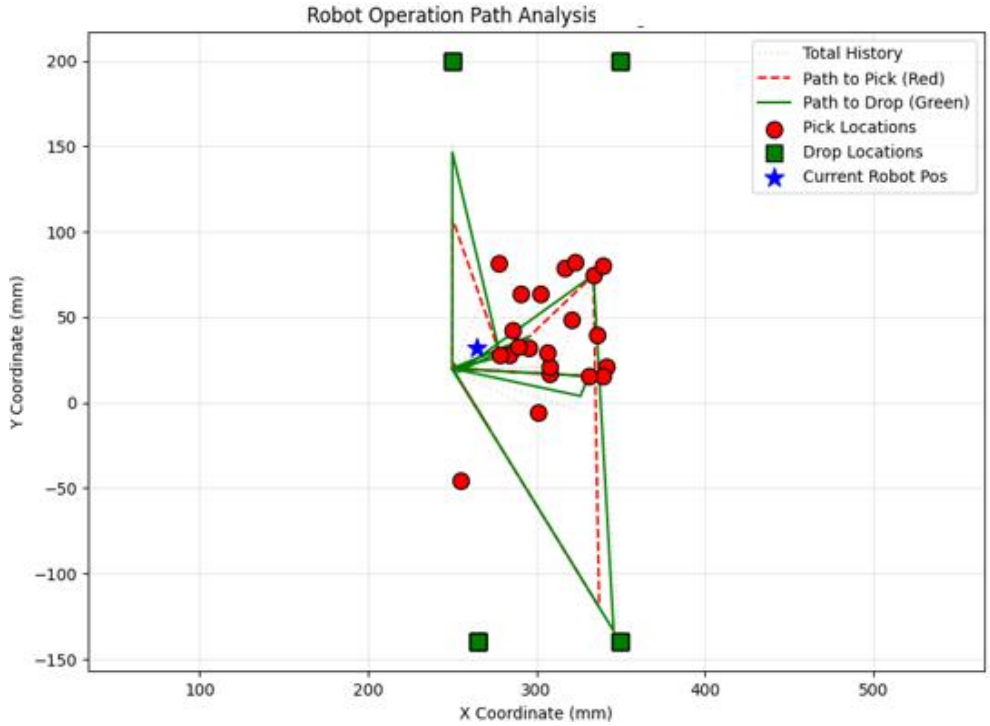

Fig. 10. Motion planning visualization for robotic arm

Table 2 presents the experimental results from 22 QR-based sorting trials, including the pick coordinates (Pick_X, Pick_Y) and drop coordinates (Drop_X, Drop_Y) executed by the robot in each cycle.

Table 2. Experimental results of controlled pick-and-place positioning using QR code data

QR_Data	Pick_X	Pick_Y	Drop_X	Drop_Y	Success rate	Position Error (mm)	Average cycle time (s)
EMC_RMUTI4	320.80905	48.797192	265	-140	98.82%	0.135	6.6
EMC_RMUTI2	301.10419	-5.5735254	350	200	99.62%	0.066	6.8
EMC_RMUTI4	333.91476	74.623108	265	-140	98.99%	0.387	6.8
EMC_RMUTI3	307.83258	16.891714	350	-140	98.97%	0.157	6.7
EMC_RMUTI3	339.39273	80.334747	350	-140	99.31%	0.121	6.6
EMC_RMUTI4	316.91284	79.115891	265	-140	99.73%	0.277	6.5
EMC_RMUTI2	308.06329	21.433868	350	200	99.06%	0.271	6.8
EMC_RMUTI4	283.88672	28.090902	265	-140	99.45%	0.057	6.9
EMC_RMUTI3	278.39078	28.155251	350	-140	99.87%	0.284	6.8
EMC_RMUTI1	285.98825	42.266323	250	200	99.15%	0.397	6.8
EMC_RMUTI1	335.94107	39.862103	250	200	98.9%	0.22	6.9
EMC_RMUTI1	295.71359	32.034012	250	200	99.46%	0.351	6.9
EMC_RMUTI3	289.62506	32.547077	350	-140	99.07%	0.264	6.7
EMC_RMUTI2	306.55307	29.374174	350	200	99.36%	0.205	6.8
EMC_RMUTI2	277.42767	81.714325	350	200	99.32%	0.101	6.6
EMC_RMUTI1	341.73788	21.40826	250	200	99.46%	0.093	6.7
EMC_RMUTI3	339.44785	15.635478	350	-140	99.02%	0.239	6.9
EMC_RMUTI4	331.09744	15.545965	265	-140	99.87%	0.068	6.8
EMC_RMUTI1	290.67285	63.827541	250	200	99.81%	0.062	6.7
EMC_RMUTI2	302.40631	63.414291	350	200	99.5%	0.106	6.7
EMC_RMUTI4	322.97308	82.297142	265	-140	99.65%	0.275	6.8
EMC_RMUTI4	254.8979	-45.840065	265	-140	99.25%	0.123	6.8

This study presents a vision-guided robotic pick-and-place system using an eye-in-hand camera and homography-based coordinate transformation. Unlike conventional industrial vision systems that depend on expensive fixed cameras, controlled lighting, or deep learning models such as CNN or YOLO, the proposed approach relies on a standard camera with classical image processing and QR decoding. Although deep learning techniques can handle complex recognition tasks, they typically require large datasets, higher computational power, and longer training time. In contrast, the QR-based method is lightweight, accurate, and easier to deploy in structured industrial environments.

Lighting variations often affect QR readability due to brightness changes, reflections, and shadows. However, the implemented edge-preserving enhancement and real-time preprocessing maintained a high detection rate (98.82%–99.87%) under varying illumination. Experimental results showed an average Euclidean positioning error of about 0.2 ± 0.12 mm, with errors ranging from 0.057 mm to 0.397 mm, demonstrating sub-millimeter precision. The average cycle time was approximately 6.8 seconds per operation. Overall, the system offers a cost-effective, accurate, and reliable solution for QR-based automated sorting without the complexity of advanced industrial vision setups.

5. Conclusion

This study describes the development and validation of a compact laboratory robotic pick-and-place platform guided by computer vision. The system employs an eye-in-hand arrangement combined with homography-based coordinate mapping to enable precise object manipulation. The architecture integrates a Dobot MG400, a standard USB camera, and fully open-source software tools-including Python, OpenCV, and Pyzbar-resulting in a vendor-neutral and reproducible framework appropriate for Industry 4.0 experimentation and smart manufacturing education.

In contrast to conventional closed-loop visual servoing methods such as IBVS and PBVS, which depend on continuous visual feedback and velocity-level control, the proposed approach utilizes a calibrated open-loop guidance scheme. Through intrinsic camera calibration, lens distortion correction, and planar homography transformation, image-plane coordinates are reliably converted into robot Cartesian coordinates while keeping computational demands relatively low. To address illumination variability, edge-preserving filtering and tailored preprocessing techniques were applied to stabilize QR detection performance. Unlike industrial vision setups that require tightly controlled

lighting conditions, or deep learning–based detection systems that rely on extensive datasets and high processing power, this framework provides a transparent, economical, and practically deployable alternative for structured environments.

The findings demonstrate that a homography-driven, open-source eye-in-hand robotic system can deliver consistent accuracy and operational stability in QR-guided sorting tasks, while preserving flexibility for future integration of AI-based optimization and intelligent manufacturing enhancements.

Supplementary Materials: The following supporting information can be downloaded at: https://www.youtube.com/watch?v=Kie5iR5UjaM.

Author Contribution: All authors contributed equally to the main contributor to this paper. All authors read and approved the final paper.

Funding: This research received no external funding.

Acknowledgment: This research was carried out at the laboratories of the Faculty of Engineering, Thaksin University, Phatthalung, and in the field of Mechatronics Engineering, Faculty of Engineering, Rajamangala University of Technology Isan Khon Kaen Campus. The authors would like to sincerely thank the staff of the Mechatronics Engineering Department for their valuable support and assistance throughout this study.

Conflicts of Interest: The authors declare no conflict of interest.

References

[1] I. Haryanto, M. A. Rhamdhani, and R. Dhelika, “Mechanical disassembly of waste PCBs: Review of methods for removing through-hole technology and surface-mount technology components,” *International Journal of Environmental Science and Technology*, vol. 23, no. 12, 2026, https://doi.org/10.1007/s13762-025-06903-6.

[2] Z. Wraith, “AI-driven vision and robotics framework for intelligent manufacturing,” *Journal of Computer Technology and Software*, vol. 4, no. 4, 2025, https://doi.org/10.5281/zenodo.15340687.

[3] A. Fingrut, S. Altun, and Y. Li, “Integration of robotics and computer vision for responsive and autonomous bricklaying in construction,” *Technology | Architecture + Design*, vol. 8, no. 2, pp. 271–281, 2024, https://doi.org/10.1080/24751448.2024.2405406.

[4] R. Ma, Y. Liu, E. W. Graf, and J. Oyekan, “Applying vision-guided graph neural networks for adaptive task planning in dynamic human–robot collaborative scenarios,” *Advanced Robotics*, vol. 38, no. 23, pp. 1690–1709, 2024, https://doi.org/10.1080/01691864.2024.2407115.

[5] P. S. Aithal, “Enhancing industrial automation through efficient technology management in society,” *International Journal of Applied Engineering and Management Letters*, vol. 7, no. 4, pp. 184–215, 2023, https://dx.doi.org/10.2139/ssrn.4674882.

[6] I. H. Sarker, “AI-based modeling: Techniques, applications and research issues towards automation, intelligent and smart systems,” *SN Computer Science*, vol. 3, p. 158, 2022, https://doi.org/10.1007/s42979-022-01043-x.

[7] H. R. Chi, C. K. Wu, N. -F. Huang, K. -F. Tsang and A. Radwan, "A Survey of Network Automation for Industrial Internet-of-Things Toward Industry 5.0," *IEEE Transactions on Industrial Informatics*, vol. 19, no. 2, pp. 2065-2077, 2023, https://doi.org/10.1109/TII.2022.3215231.

[8] B. Maschler and M. Weyrich, "Deep Transfer Learning for Industrial Automation: A Review and Discussion of New Techniques for Data-Driven Machine Learning," *IEEE Industrial Electronics Magazine*, vol. 15, no. 2, pp. 65-75, 2021, https://doi.org/10.1109/MIE.2020.3034884.

[9] S. Răileanu, T. Borangiu, F. Anton, and S. Anton, “Open source machine vision platform for manufacturing and robotics,” *IFAC-PapersOnLine*, vol. 54, no. 1, pp. 522–527, 2021, https://doi.org/10.1016/j.ifacol.2021.08.060.

[10] L. Zhou, L. Zhang and N. Konz, "Computer Vision Techniques in Manufacturing," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 53, no. 1, pp. 105-117, 2023, https://doi.org/10.1109/TSMC.2022.3166397.

[11] V. Voronin, M. Zhdanova, E. Semenishchev, A. Zelenskii, Y. Cen, S. Agaian, "Action recognition for the robotics and manufacturing automation using 3-D binary micro-block difference," *International Journal of Advanced Manufacturing Technology*, vol. 117, pp. 2319–2330, 2021, https://doi.org/10.1007/s00170-021-07613-2.

[12] R. M. Sarıyer, A. D. Edwards, and S. H. Needs, "Open hardware for microfluidics: Exploiting Raspberry Pi single-board computer and camera systems for customisable laboratory instrumentation," *Biosensors*, vol. 13, no. 10, p. 948, 2023, https://doi.org/10.3390/bios13100948.

[13] M. Niveditha, S. Geetha, and M. L. Chayadevi, "Real-time QR code recognition for instant data access," *SSRN*, 2025, https://dx.doi.org/10.2139/ssrn.5876362.

[14] T. Kiyokawa, J. Takamatsu, and S. Koyanaka, "Challenges for future robotic sorters of mixed industrial waste: A survey," *IEEE Transactions on Automation Science and Engineering*, vol. 21, no. 1, pp. 1023–1040, 2024, https://doi.org/10.1109/TASE.2022.3221969.

[15] A.-P. Botezatu, A.-I. Iancu, and A. Burlacu, "Hybrid deep learning framework for eye-in-hand visual control systems," *Robotics*, vol. 14, no. 5, p. 66, 2025, https://doi.org/10.3390/robotics14050066.

[16] S. Sulaiman, T. B. Jensen, S. H. Bengtson, S. Bøgh, "Kinematic analysis and integration of vision algorithms for a mobile manipulator employed inside a self-driving laboratory," *International Journal of Intelligent Robotics Applications*, 2026, https://doi.org/10.1007/s41315-025-00518-3.

[17] Y. M. Alginahi, O. Sabri, and W. Said, "Reinforcement learning for industrial automation: A comprehensive review of adaptive control and decision-making in smart factories," *Machines*, vol. 13, no. 12, p. 1140, 2025, doi: 10.3390/machines13121140.

[18] J. A. Fortoul-Diaz, L. A. Carrillo-Martinez, A. Centeno-Tellez, F. Cortes-Santacruz, I. Olmos-Pineda, and R. R. Flores-Quintero, "A smart factory architecture based on Industry 4.0 technologies: Open-source software implementation," *IEEE Access*, vol. 11, pp. 101727–101749, 2023, https://doi.org/10.1109/ACCESS.2023.3316116.

[19] Y. Liu, J. Yi, P. Chi, H. Liao, Q. Zhang, Z. Wang, "Review of trajectory planning for humanoid welding robot manipulators: From fundamentals to industrial applications in intelligent manufacturing," *Journal of Intelligent Manufacturing*, 2026, https://doi.org/10.1007/s10845-025-02765-4.

[20] L. Zemite, J. Kozadajevs, L. Jansons, I. Bode, E. Dzelzitis, and K. Palkova, "Integrating renewable energy solutions in small-scale industrial facilities," *Energies*, vol. 17, no. 11, p. 2792, 2024, https://doi.org/10.3390/en17112792.

[21] A. Rozhok, R. Abate, E. Manoli, and L. Nele, "A review of recent advanced applications in smart manufacturing systems," *Journal of Manufacturing and Materials Processing*, vol. 10, no. 1, p. 1, 2026, https://doi.org/10.3390/jmmp10010001.

[22] A. Iqdymat and G. Stamatescu, "Reinforcement learning of a six-DOF industrial manipulator for pick-and-place application using efficient control in warehouse management," *Sustainability*, vol. 17, no. 2, p. 432, 2025, https://doi.org/10.3390/su17020432.

[23] L. Yutthanakorn and S. Charoenseang, "Augmented reality based visual programming of robot training for educational demonstration site," *Advances in Science, Technology and Engineering Systems Journal*, vol. 8, no. 5, pp. 8–16, 2023, https://doi.org/10.25046/aj080502.

[24] P.-S. Tsai, T.-F. Wu, J.-Y. Chen, and F.-H. Lee, "Drawing system with Dobot Magician manipulator based on image processing," *Machines*, vol. 9, no. 12, p. 302, 2021, https://doi.org/10.3390/machines9120302.

[25] M. Amersdorfer and T. Meurer, "Equidistant tool path and Cartesian trajectory planning for robotic machining of curved freeform surfaces," *IEEE Transactions on Automation Science and Engineering*, vol. 19, no. 4, pp. 3311–3323, 2022, https://doi.org/10.1109/TASE.2021.3117691.

[26] C. Calderón-Cordova, R. Sarango, D. Castillo, and V. Lakshminarayanan, "A deep reinforcement learning framework for control of robotic manipulators in simulated environments," *IEEE Access*, vol. 12, pp. 103133–103161, 2024, https://doi.org/10.1109/ACCESS.2024.3432741.

[27] P. Kohut and K. Skop, "Vision systems for a UR5 cobot on a quality control robotic station," *Applied Sciences*, vol. 14, no. 20, p. 9469, 2024, https://doi.org/10.3390/app14209469.

[28] N. Rupp, K. Peschke, M. Köppl, D. Drissner, and T. Zuchner, "Establishment of low-cost laboratory automation processes using AutoIt and 4-axis robots," *SLAS Technology*, vol. 27, no. 5, pp. 312–318, 2022, https://doi.org/10.1016/j.slast.2022.07.001.

[29] G. Fang and J. Cheng, "Advances in climbing robots for vertical structures in the past decade: A review," *Biomimetics*, vol. 8, no. 1, p. 47, 2023, https://doi.org/10.3390/biomimetics8010047.

[30] M. S. Xavier *et al*., "Soft Pneumatic Actuators: A Review of Design, Fabrication, Modeling, Sensing, Control and Applications," *IEEE Access*, vol. 10, pp. 59442-59485, 2022, https://doi.org/10.1109/ACCESS.2022.3179589.

[31] F. Nagata, R. Abe, S. Sakata, K. Watanabe, and M. K. Habib, "Hyper CLS-data-based robotic interface and its application to intelligent peg-in-hole task robot incorporating a CNN model for defect detection," *Machines*, vol. 12, no. 11, p. 757, 2024, https://doi.org/10.3390/machines12110757.

[32] A. Burghardt *et al*., "TCP parameters monitoring of robotic stations," *Electronics*, vol. 11, no. 20, p. 3415, 2022, https://doi.org/10.3390/electronics11203415.

[33] I. Enebuse, M. Foo, B. S. K. K. Ibrahim, H. Ahmed, F. Supmak, and O. S. Eyobu, "A comparative review of hand-eye calibration techniques for vision guided robots," *IEEE Access*, vol. 9, pp. 113143–113155, 2021, https://doi.org/10.1109/ACCESS.2021.3104514.

[34] J. Jiang, X. Luo, Q. Luo, *et al.*, "An overview of hand-eye calibration," *International Journal of Advanced Manufacturing Technology*, vol. 119, no. 1–2, pp. 77–97, 2022, https://doi.org/10.1007/s00170-021-08233-6.

[35] L. Wu, F. Yu, T. N. Do, and J. Wang, "Camera frame misalignment in a teleoperated eye-in-hand robot: Effects and a simple correction method," *IEEE Transactions on Human-Machine Systems*, vol. 53, no. 1, pp. 2–12, 2023, https://doi.org/10.1109/THMS.2022.3217453.

[36] X. Xiao, B. Liu, G. Warnell, P. Stone, "Motion planning and control for mobile robot navigation using machine learning: A survey," *Autonomous Robots*, vol. 46, no. 4, pp. 569–597, 2022, https://doi.org/10.1007/s10514-022-10039-8.

[37] X. Mo, Z. Huang, Y. Xing, and C. Lv, "Multi-agent trajectory prediction with heterogeneous edge-enhanced graph attention network," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 7, pp. 9554–9567, 2022, https://doi.org/10.1109/TITS.2022.3146300.

[38] S. Qiu, M. Wang, and M. R. Kermani, "A modern solution for an old calibration problem," *IEEE Instrumentation & Measurement Magazine*, vol. 24, no. 3, pp. 28–35, 2021, https://doi.org/10.1109/MIM.2021.9436097.

[39] S. Chen *et al*., "Research on high-precision localization method for transport robots in industrial environments based on improved AMCL and QR code assistance," *Scientific Reports*, vol. 15, p. 6214, 2025, https://doi.org/10.1038/s41598-024-85067-8.

[40] A. Vukićević, M. Mladineo, N. Banduka, I. Mačužić, "A smart Warehouse 4.0 approach for the pallet management using machine vision and Internet of Things (IoT): A real industrial case study," *Advances in Production Engineering & Management*, vol. 16, no. 3, pp. 297–306, 2021, https://doi.org/10.14743/apem2021.3.401.

[41] J.-S. Jwo, C.-S. Lin, and C.-H. Lee, "Smart technology-driven aspects for human-in-the-loop smart manufacturing," *International Journal of Advanced Manufacturing Technology*, vol. 114, no. 5–6, pp. 1741–1752, 2021, https://doi.org/10.1007/s00170-021-06977-9.

[42] J. Trojanowska, J. Husár, S. Hrehova, and L. Knapčíková, "Poka-yoke in smart production systems with pick-to-light implementation to increase efficiency," *Applied Sciences*, vol. 13, no. 21, p. 11715, 2023, https://doi.org/10.3390/app132111715.

[43] F.-K. Wang, B. Rahardjo, and P. R. Rovira, "Lean Six Sigma with value stream mapping in Industry 4.0 for human-centered workstation design," *Sustainability*, vol. 14, no. 17, p. 11020, 2022, https://doi.org/10.3390/su141711020.

[44] G. Maier, R. Gruna, T. Längle, and J. Beyerer, "A survey of the state of the art in sensor-based sorting technology and research," *IEEE Access*, vol. 12, pp. 6473–6493, 2024, https://doi.org/10.1109/ACCESS.2024.3350987.

[45] F. K. Konstantinidis, S. Sifnaios, G. Tsimiklis, S. G. Mouroutsos, A. Amditis, and A. Gasteratos, "Multi-sensor cyber-physical sorting system (CPSS) based on Industry 4.0 principles: A multi-functional approach," *Procedia Computer Science*, vol. 217, pp. 227–237, 2023, https://doi.org/10.1016/j.procs.2022.12.218.

[46] N. Boysen, S. Schwerdfeger, and M. W. Ulmer, "Robotized sorting systems: Large-scale scheduling under real-time conditions with limited lookahead," *European Journal of Operational Research*, vol. 310, no. 2, pp. 582–596, 2023, https://doi.org/10.1016/j.ejor.2023.03.037.

[47] M. Huo, Z. Zhang, X. Ren, X. Yang, and C. Ye, "AbHE: All attention-based homography estimation," *IEEE Transactions on Instrumentation and Measurement*, vol. 73, pp. 1–11, p. 5013911, 2024, https://doi.org/10.1109/TIM.2024.3374320.

[48] C. Zhang, S. Li, and Z. Zhang, "Modeling and simulation of industrial robot arms using Simscape Multibody," *Facta Universitatis, Series: Mechanical Engineering*, vol. 23, no. 2, pp. 265–286, 2025, https://doi.org/10.22190/FUME230215017Z.

[49] D. Zhang *et al*., "Design and benchmarking of a multimodality sensor for robotic manipulation with GAN-based cross-modality interpretation," *IEEE Transactions on Robotics*, vol. 41, pp. 1278–1295, 2025, https://doi.org/10.1109/TRO.2025.3526296.

[50] X. Zhang, Y. Zhou, P. Qiao, X. Lv, J. Li, T. Du, and Y. Cai, "Image registration algorithm for remote sensing images based on pixel location information," *Remote Sensing*, vol. 15, no. 2, p. 436, 2023, https://doi.org/10.3390/rs15020436.

[51] C. Cao, "Research on a visual servoing control method based on perspective transformation under spatial constraint," *Machines*, vol. 10, no. 11, p. 1090, 2022, https://doi.org/10.3390/machines10111090.

[52] Y. Luo, X. Wang, Y. Liao, Q. Fu, C. Shu, Y. Wu, and Y. He, "A review of homography estimation: Advances and challenges," *Electronics*, vol. 12, no. 24, p. 4977, 2023, https://doi.org/10.3390/electronics12244977.

[53] A. Lobbezoo, Y. Qian, and H.-J. Kwon, "Reinforcement learning for pick and place operations in robotics: A survey," *Robotics*, vol. 10, no. 3, p. 105, 2021, https://doi.org/10.3390/robotics10030105.

[54] A. A. Santos *et al*., "Integration of artificial vision and image processing into a pick and place collaborative robotic system," *Journal of Intelligent & Robotic Systems*, vol. 110, p. 159, 2024, https://doi.org/10.1007/s10846-024-02195-z.

[55] B. Tipary and G. Erdős, "Tolerance analysis for robotic pick-and-place operations," *International Journal of Advanced Manufacturing Technology*, vol. 117, pp. 1405–1426, 2021, https://doi.org/10.1007/s00170-021-07672-5.

[56] R. B. Yunus, N. Zainuddin, H. Daud, R. Kannan, S. A. A. Karim, and M. M. Yahaya, "A modified structured spectral HS method for nonlinear least squares problems and applications in robot arm control," *Mathematics*, vol. 11, no. 14, p. 3215, 2023, https://doi.org/10.3390/math11143215.

[57] S. de Marco, M.-D. Hua, R. Mahony, and T. Hamel, "Homography estimation of a moving planar scene from direct point correspondence," *IEEE Transactions on Control Systems Technology*, vol. 29, no. 3, pp. 1284–1295, 2021, https://doi.org/10.1109/TCST.2019.2960483.

[58] L. Bu, H. Huo, X. Liu, and F. Bu, "Concentric circle grids for camera calibration considering lens distortion," *Optics and Lasers in Engineering*, vol. 140, p. 106527, 2021, https://doi.org/10.1016/j.optlaseng.2020.106527.

[59] N. M. Nazari, N. Samandari, and I. Waqar, "Practical applications of homogeneous coordinates in image transformations using MATLAB," *Turkish Journal of Computer and Mathematics Education*, vol. 16, no. 1, pp. 63–80, 2025, https://doi.org/10.61841/turcomat.v16i1.15087.

[60] M. K. Teoh, K. T. K. Teo, and H. P. Yoong, "Numerical computation-based position estimation for QR code object marker: Mathematical model and simulation," *Computation*, vol. 10, no. 9, p. 147, 2022, https://doi.org/10.3390/computation10090147.

[61] H. Nogami, Y. Kanetaka, Y. Naganawa, Y. Maeda, and N. Fukushima, "Decomposed multilateral filtering for accelerating filtering with multiple guidance images," *Sensors*, vol. 24, no. 2, p. 633, 2024, https://doi.org/10.3390/s24020633.

[62] I. Benito-Altamirano *et al*., "Back-compatible color QR codes for colorimetric applications," *Pattern Recognition*, vol. 133, p. 108981, 2023, https://doi.org/10.1016/j.patcog.2022.108981.

[63] G. Wang, F. Xu, K. Zhou, and Z. Pang, "S-velocity profile of industrial robot based on NURBS curve and slerp interpolation," *Processes*, vol. 10, no. 11, p. 2195, 2022, https://doi.org/10.3390/pr10112195.

[64] B. Mandal and P. S. Bhowmik, "dssCLATT: A tkinter-based software tool to learn and analyze for advancement of dye-sensitized solar cell technology," *Computers and Electrical Engineering*, vol. 119, p. 109530, 2024, https://doi.org/10.1016/j.compeleceng.2024.109530.

[65] E. A. Nugroho, J. D. Setiawan, and M. Munadi, "Handling four DOF robot to move objects based on color and weight using fuzzy logic control," *Journal of Robot Control (JRC)*, vol. 4, no. 6, pp. 769–779, 2023, https://doi.org/10.18196/jrc.v4i6.20087.

[66] Z. Li, B. Lai, H. Wang and Y. Pan, "Homography-Based Visual Servoing of Robot Pose Under an Uncalibrated Eye-to-Hand Camera," *IEEE/ASME Transactions on Mechatronics*, vol. 29, no. 3, pp. 1891-1902, 2024, https://doi.org/10.1109/TMECH.2023.3326462.