IAES International Journal of Artificial Intelligence (IJ-AI)
Vol. 15, No. 4, August 2026, pp. 3352~3364
ISSN: 2252-8938, DOI: 10.11591/ijai.v15.i4.pp3352-3364

Intelligent object sorting system using Dobot Magician and computer vision

Tossapol Jangnoi[1], Thewin Sakunbunyong[2], Viroch Sukontanakarn[2], Tanawat Chalardsakul[2]
[1]Department of Mechanical Engineering, Faculty of Engineering, Rajamangala University of Technology Isan, Khon Kaen, Thailand
[2]Department of Mechatronics Engineering, Faculty of Engineering, Rajamangala University of Technology Isan, Khon Kaen, Thailand

Article Info

Article history:

Received Aug 5, 2025
Revised May 13, 2026
Accepted May 24, 2026

Keywords:

Computer vision
Dobot Magician
Object sorting
OpenCV library
Python program

ABSTRACT

This study aims to analyze the relationship between object color, shape, and count and the performance of a robotic object sorting system, measured by execution time and accuracy. The Dobot Magician robot was used for object manipulation, while computer vision based on object detection was implemented using Python and an open source computer vision library (OpenCV). Color and shape segmentation were performed using the hue, saturation, and value (HSV) color space and OpenCV library. Experimental results indicate that three shapes (circle, triangle, and square) and four colors (green, red, blue, and yellow) affect picking time and accuracy differently. Squares generally required the longest time to pick, while triangles often had the shortest times across various colors. Accuracy remained consistently high across all colors and shapes, with green and yellow objects showing slightly higher average accuracy. These findings provide valuable insights into the factors influencing the performance of automated robotic grasping systems, which can be applied to optimize robot design and improve the efficiency and precision of object sorting tasks.

Corresponding Author:

Thewin Sakunbunyong
Department of Mechatronics Engineering, Faculty of Engineering
Rajamangala University of Technology Isan
Khon Kaen, Thailand
Email: thewin.sa@rmuti.ac.th

1. INTRODUCTION

In modern industrial automation, object sorting systems [1], [2] play a crucial role in enhancing productivity, accuracy, and efficiency. Traditional sorting methods rely heavily on pre-defined rules and handcrafted features, which often fail to generalize in dynamic or complex environments. To address these limitations, artificial intelligence (AI)—particularly deep learning—has emerged as a powerful solution, enabling machines to perceive and classify objects more effectively. This research presents a deep learning-based approach [3], [4] for object sorting using a robotic arm, with a specific focus on identifying and classifying objects based on their shape and color [5]–[7]. The system extracts meaningful visual features from captured images, allowing for robust object recognition under varying conditions. The classified information is then used to guide the robotic arm for accurate and efficient sorting. The integration of computer vision and robotic manipulation offers significant potential for applications in packaging, waste management, manufacturing, and logistics [8]–[10]. This study aims to design, develop, and evaluate a prototype system that demonstrates the feasibility and performance of such integration. The proposed system not only automates the sorting process but also improves adaptability and scalability compared to rule-based or sensor-only approaches. The key contributions of this work include the design of a vision-based

classification model, the implementation of an object-sorting mechanism using a robotic arm [11]–[13], and the evaluation of the system's performance in real-time sorting tasks.

Object detection is a continuously studied research topic in the field of computer vision, aiming to identify and classify objects across sequences of frames. Detecting objects in video sequences is particularly challenging and often requires significant processing time. Object detection is generally considered the first step in building more complex computer vision systems [14], [15]. Each object possesses several global features, such as color and shape, which can be used to describe its overall characteristics. Detecting object shapes and colors [16]–[18] is a fundamental task in computer vision. This task is typically approached using traditional methods with open source computer vision library (OpenCV), deep learning models such as convolutional neural networks (CNNs), object detectors like you only look once (YOLO), single shot detector (SSD), and faster region-based convolutional neural network (Faster R-CNN), as well as hybrid techniques that combine traditional and deep learning approaches. Additionally, specialized tools such as MediaPipe, scikit-image, TensorFlow, and PyTorch are often employed. A summary of the common methods used for detecting object shape and color is presented in Table 1.

Recent advances in deep learning have significantly improved the performance of machine vision systems used in object detection and classification tasks. CNNs [19], [20] have demonstrated exceptional accuracy in image classification, making them well-suited for industrial applications involving visual recognition. Several studies have explored object sorting using deep learning techniques. For example, one study developed a vision-based robotic system that sorts waste materials using a CNN to classify different types of plastic. The system achieved high classification accuracy under controlled lighting conditions. Similarly, another study implemented a sorting mechanism for fruits based on color and size, using a YOLO-based model for real-time detection [21]. In terms of robotic integration, researchers have demonstrated systems where a robotic arm is guided by image recognition results to manipulate and sort objects on an assembly line. These studies highlight the importance of accurate object localization and the role of robotic kinematics in achieving efficient sorting. However, many existing works focus on either color-based or shape-based sorting independently. Few systems combine both attributes to improve classification robustness, particularly in environments with visually similar objects. Moreover, while some systems use traditional image processing techniques (e.g., color thresholding and contour detection), these approaches are often sensitive to noise and less generalizable across varying conditions.

This research employs a method for detecting the shape and color of objects, focusing on three distinct shapes: circle, square, and triangle. Each shape is available in four colors: red, yellow, blue, and green. The detection process utilizes the OpenCV library—specifically hue, saturation, and value (HSV) color space and contour detection techniques [22]—a powerful tool widely used in computer vision applications. The model is capable of simultaneously identifying both the shape and color of objects, enabling the development of a real-time control system for the Dobot Magician robotic arm [23]–[25]. This system supports fast and efficient object sorting using easily accessible and low-cost resources. Furthermore, the research emphasizes a self-developed, code-based approach using Python and OpenCV, without relying on proprietary machine vision software provided by the Dobot Magician manufacturer. Table 2 presents a comparison of programming methods used to control the Dobot Magician.

Table 1. Summary of methods for detecting object shape and color

Method	Ease of use	Accuracy	Best for
OpenCV (HSV) + contours)	Easy	Medium	Basic shape and color detection
Deep learning (CNNs)	Complex	High	Advanced, precise detection
Hybrid approach	Moderate	High	Balanced speed and accuracy

Table 2. Methods for controlling Dobot Magician

Programming	Use teaching	Use vision	Controlled by Python	Suitable for
Python only	No	Yes	Yes	Automatic inspection system with a camera
DobotStudio only	Yes	No	No	Teaching-based tasks, manual operation
Hybrid (mix)	Yes	Yes	Yes	Camera-based inspection followed by teaching

2. CONCEPT DESIGN AND CONTROL

The proposed object-sorting system integrates a classification model with a robotic arm to identify and sort objects based on their shape and color. The hardware and software components work together to form a real-time sorting system. The architectural workflow of this robotic object-handling system is illustrated in Figure 1. The user initiates or controls the sorting process via a personal computer (PC). The PC receives visual input from a camera that detects the position of the workpiece [26]. The image is then

processed, and control commands are sent to the Dobot Magician robotic arm. Based on these commands, the Dobot Magician picks up the detected workpiece and places it into the designated workpiece box.

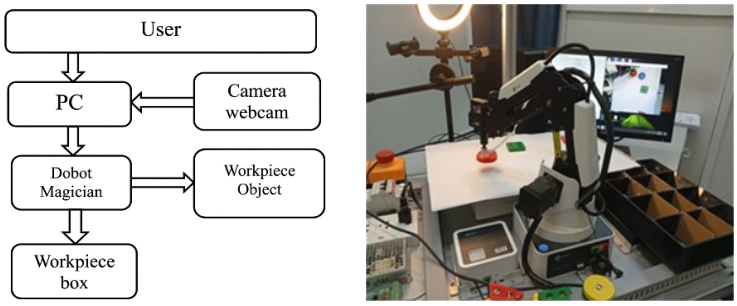


Figure 1. Concept of the architecture model

2.1. Mechanical and electrical setup for the Dobot Magician

The Dobot Magician, shown in Figure 2, is a versatile desktop robotic arm equipped with stepper motors, designed to perform tasks such as pick-and-place operations [27]. Its working principle combines mechanical motion control, sensors, and software programming. The robotic arm features four joints (axes): base rotation, shoulder, elbow, and wrist rotation. Each joint is powered by a stepper motor that precisely controls its angle. By coordinating these motors, the arm can move its end effector (tool) to any position within its working envelope. Table 3 presents the working range and maximum speed of each joint of the Dobot Magician robotic arm. The robot features four main joints: base rotation (J_1), shoulder rotation (J_2), elbow rotation (J_3), and wrist rotation (J_4).

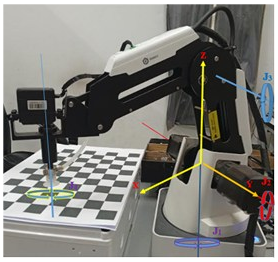


Figure 2. Rotation axes of each joint of the Dobot Magician robotic arm

Table 3. Area and speed of the Dobot Magician working

Axis	Range	Maximum speed
Bbase rotation (J_1)	-135° to 135°	320 °/s
Shoulder rotation (J_2)	0° to 85°	320 °/s
Elbow rotation (J_3)	-10° to 95°	320 °/s
Wrist rotation (J_4)	90° to -90°	480 °/s

2.2. Process of object detection

The system shown in Figure 3 begins with step 1: input device (camera/image), where a camera or image input device captures images of the objects to be detected and sorted. In step 2: image preprocessing, the captured images are resized and filtered to enhance quality and reduce noise, preparing them for further analysis. Step 3: feature extraction involves extracting key features from the objects, such as color—using color spaces like red, green, blue (RGB) or HSV—and shape, detected through contour or edge detection methods. In step 4: object detection, the system identifies and locates the objects present in the images. During step 5: object classification, the detected objects are classified based on labels defined by their shape and color, along with their image coordinates. In step 6: coordinate mapping, these image coordinates are converted into physical coordinates within the robot's working area (Dobot workspace coordinates). Step 7: Dobot controller sends commands to the Dobot robotic arm to pick up the objects and move them to designated locations. Finally, in step 8: physical sorting, the robotic arm physically places the objects into specific bins or locations according to their classified shape or color.

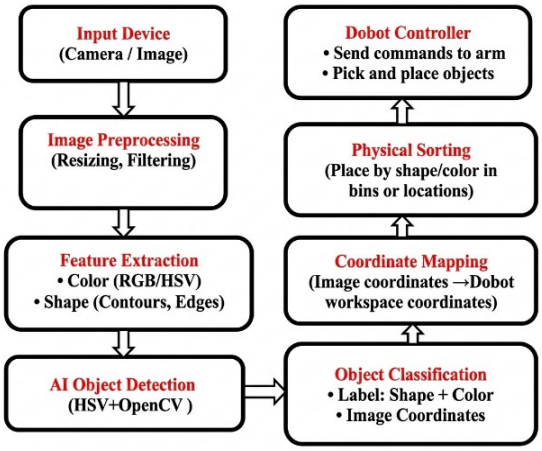


Figure 3. Block diagram of shape and color object detection using the Dobot Magician robotic arm

2.3. Methods for detecting object shape and color

Detecting object shapes and colors is a fundamental task in computer vision, with various methods available depending on the required complexity and accuracy. This research employs traditional computer vision techniques using OpenCV to identify the shape of an object (e.g., circle, triangle, square) and its color (e.g., red, green, blue, yellow). The process of detecting an object's shape and color is as:

i) Read the image
   - Use cv2.imread() to load the image.
   - The image is read as a matrix of pixels in blue, green, red (BGR) color space.

ii) Convert to grayscale
   - Convert the image from BGR to grayscale using cv2.cvtColor(image, cv2.COLOR_BGR2GRAY).
   - This simplifies the image by removing color information, making shape detection easier.

iii) Blur and threshold (or edge detection)
   - Use cv2.GaussianBlur() to remove noise and smooth the image.
   - Use cv2.threshold() or cv2.Canny() to create a binary image (black and white).
   - Objects become white.
   - Background becomes black.

iv) Find contours
   - Use cv2.findContours() to detect the boundaries of objects (shapes).
   - Contours are curves that connect continuous points around a shape.

v) Detect shapes
   - Approximate the contour using cv2.approxPolyDP() to get the number of corners:
   - 3 corners → triangle
   - 4 corners → square (check width/height ratio)
   - More than 4 → circle or oval
   - Use cv2.boundingRect() to get the position and size for labeling.

vi) Detect color
   - Take the pixel values at the center of the shape (or average inside it).
   - Convert the image to HSV color space using cv2.cvtColor(image, cv2.COLOR_BGR2HSV) for more accurate color detection.
   - Define HSV ranges for:
   - Red: H around 0 or 170–180
   - Green: H around 60
   - Blue: H around 120
   - Yellow: H around 30
   - Use cv2.inRange() to classify colors based on HSV values.

vii) Label, draw and display results
   - Use cv2.putText() to write the name of the shape and color on the image.
   - Use cv2.drawContours() to outline the shape.

2.4. Coordinate system in method proposed

This section discusses the concepts and mathematical foundations of coordinate transformations [28] for object pick-and-place tasks using a fixed overhead camera with a Dobot robotic arm, particularly for sorting objects based on shape and color. The camera captures the workspace from a top-down fixed position, identifying objects in the image using pixel coordinates. To enable the robot to interact accurately with these objects, it is necessary to transform the pixel coordinates into real-world coordinates and subsequently into the robot's coordinate frame. A USB camera is mounted above the workspace to capture real-time images of the objects. This process requires precise camera calibration and coordinate transformation methods. The setup of the equipment is shown in Figure 4.

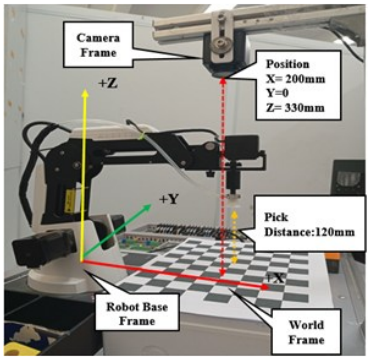


Figure 4. Coordinate system of the experiment setup

When using the Dobot Magician to pick up an object, it is essential to transform the object's coordinates from the camera frame to the robot's coordinate system, enabling precise movement to the target position. Homogeneous transformation equations are applied to convert object positions from the camera coordinate system to the robot's coordinate system, allowing the Dobot Magician to accurately perform pick-and-place tasks. Generally, three coordinate frames are involved: the camera frame, the world frame, and the robot frame. The general transformation is shown in (1).

$$P_{robot} = T_{robot \leftarrow wolrd} \cdot T_{world \leftarrow cam} \cdot P_{cam} \quad (1)$$

Where P_{cam} is position of the object in camera coordinates, $T_{world \leftarrow cam}$ is transformation matrix from camera frame to world frame, $T_{robot \leftarrow wolrd}$ is transformation from world frame to robot frame, and P_{robot} is object position in robot coordinates (input for the robot). Each transformation matrix is a 4×4 homogeneous matrix, as shown in (2).

$$T = \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix} \quad (2)$$

Where R =3×3 rotation matrix (orientation between frames), t =3×1 translation vector (position offset between frames).

2.5. Python code for implementation of process

Using PyDobot involves writing Python programs [29] to control the Dobot Magician robotic arm via a serial (COM) port. It allows users to move the arm to specific positions, set movement speeds, turn the suction cup on or off, and read the current pose of the arm. This makes it easy to automate tasks with the Dobot without relying on the DobotStudio software. To connect Dobot Magician using a manually specified port in PyDobot, simply pass the port name directly when creating the Dobot object:

```
from pydobot import Dobot
device = Dobot(port="COM5")  # Replace COM5 with your actual port
```

This establishes a serial connection between your Python script and the Dobot Magician using specified port.

3. METHODOLOGY AND MATERIAL

This section outlines the system workflow and the setup of equipment used in the experiment. The Dobot Magician is controlled using a program written in Python and OpenCV, utilizing the PyDobot library. The experimental procedure is carried out in the following steps:

3.1. Setting of the relation of coordinates

The first step involves calibrating the camera coordinates, the surface coordinates where the objects are placed, and the suction head position of the Dobot to ensure they align correctly. This is done by writing a Python program and saving it in a folder for continuous use in color and shape detection. As shown in Figure 5, the coordinate mapping process involves identifying the camera's projected points on the surface and moving the Dobot's suction head to those points. A total of four points is used, and the data is saved using a method called the homography matrix.

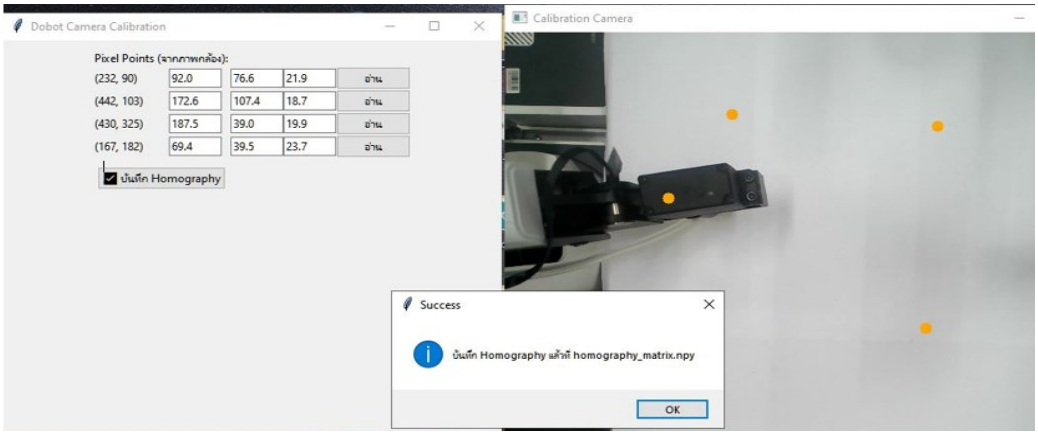
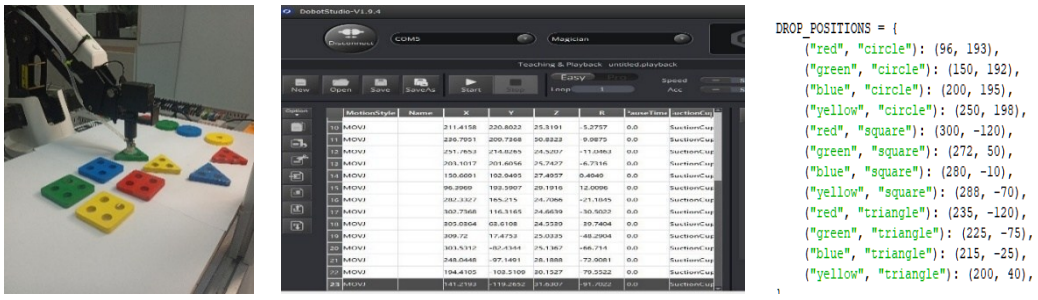


Figure 5. The calibrating of camera and dobot head coordinates four point

3.2. Workpiece placement

First, open the DobotStudio program. Then, arrange the workpieces—featuring all four colors and three shapes—in the desired positions within the reachable area of the robotic arm to ensure it can operate without errors. Figure 6 illustrates the arrangement of the workpieces. Figure 6(a) shows the placement of the workpieces on the table surface. Figure 6(b) demonstrates how to use the DobotStudio program to position the robotic arm over each workpiece and save the position values at those specific points. Figure 6(c) shows how the (x, y) coordinates of each workpiece are entered into the custom-developed program.



(a) (b) (c)

Figure 6. Arrangement and localization of workpieces: (a) workpiece positions on the table surface, (b) positioning the Dobot's suction head over each workpiece to record coordinates, and (c) entering position data into the Python program for implementation

3.3. Program execution testing

Before starting each operation, the DobotStudio program must be opened to set the robot to its home position. Once this is done, close the program and run the module in the Python program. The interface will

display two sections: one for color and shape detection, and the other for start/stop button controls, along with a table that logs the number of items sorted, as shown in Figure 7.

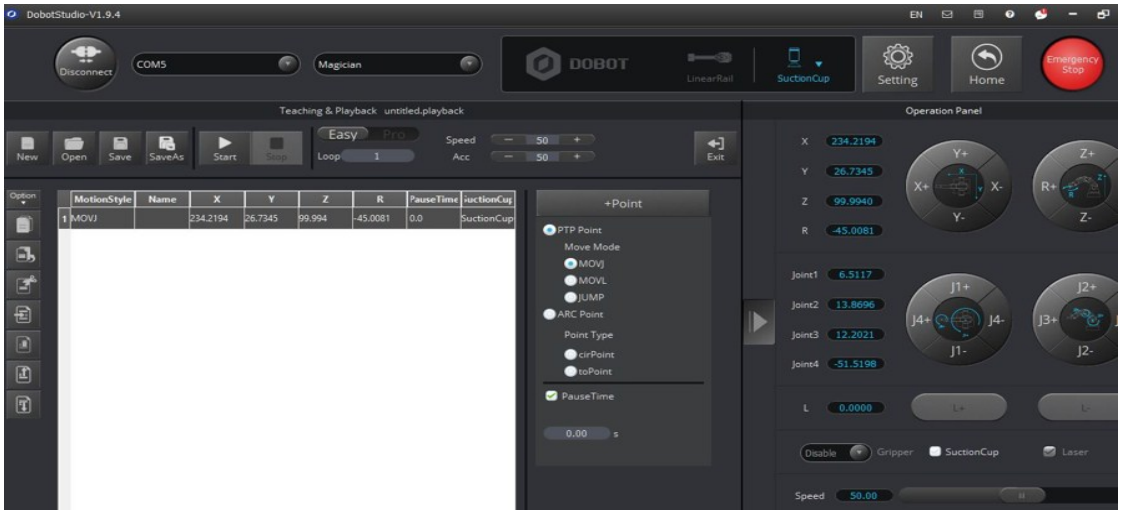


Figure 7. Setting up the DobotStudio program to move the robotic arm to the home position

3.4. Starting the program for pick-and-place operations

Before operating the system, the positions for placing the objects must first be defined, as illustrated in Figure 6(a). The DobotStudio program was used to manually move the robotic arm to each predefined object position. These coordinates were then entered into a custom-developed application. Once the setup was complete, the start button was pressed to initiate the process. The program was designed to detect objects based on their color and shape and includes the following key features:

- Ability to click two points to define the pickup zone
- Visual feedback with a green border during zone selection
- Display of a purple border once the pickup zone is set
- Buttons for export comma-separated values (CSV), reset count, and reset pickup area
- Capability to save the pickup zone to a file named pickup_area.json
- Use of a homography matrix (homography_matrix.npy) to convert between pixel coordinates and robot coordinates
- Integration with PyDobot for controlling the Dobot in automated pick-and-place operations

4. RESULTS AND DISCUSSION

4.1. Experimental results

The dataset comprises various objects with different shapes (e.g., circles, squares, triangles) and colors (e.g., red, green, blue). The images were collected under varying lighting conditions to enhance the model's generalization capability. Each image includes both shape and color attributes, as shown in Figure 8. The preliminary testing of the object shape and color detection program was carried out, as shown in Figure 9.

Figure 8. Workpieces featuring 4 colors and 3 shapes

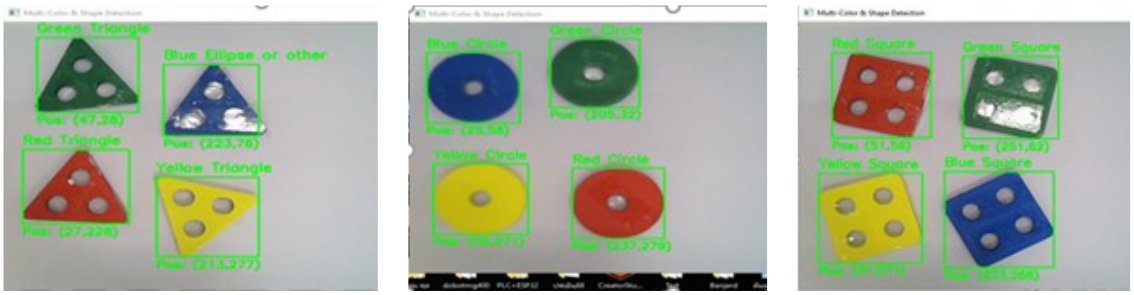


Figure 9. Object detection by shape and color using HSV and OpenCV

Illustrated in Figure 10 is a screenshot taken from a video demonstrating the operation of the automatic sorting system for workpieces with three shapes and four colors. The process begins by launching the custom-developed Python program. Next, the DobotStudio software is used to move the robotic arm to the home position. After that, the Python program is executed. When the camera detects a workpiece with a specific shape and color within the defined area, it draws a bounding box around the object and marks its center point. The coordinates are then sent to the robotic arm, which moves to pick up the object and place it in a predefined location. The system continues operating until there are no more workpieces in the designated area or the program is manually stopped.

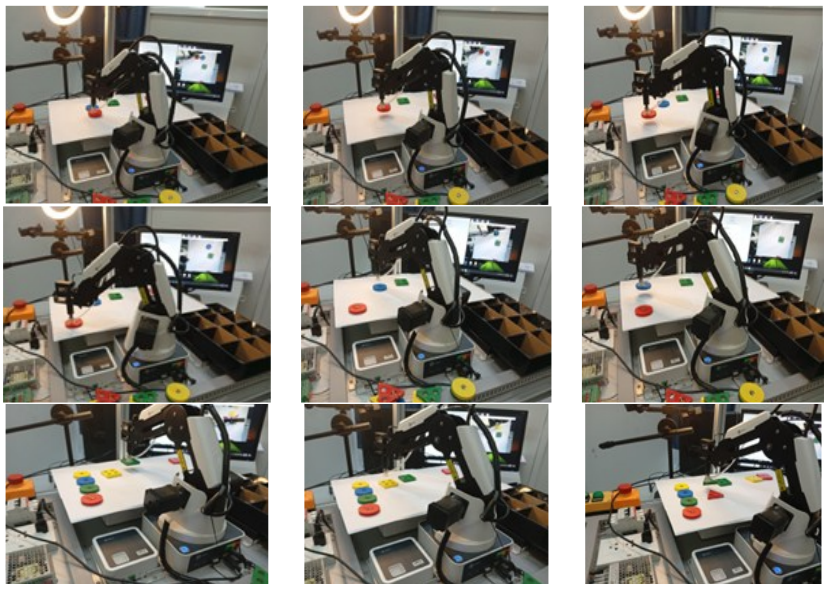


Figure 10. Actual movement of the Dobot Magician during object sorting

Table 4 shows the results of the Dobot Magician's workpiece grasping test, which indicate that the robot successfully grasped all objects in every trial. The accuracy ranged from 95.0% to 99.2%, with the red triangle achieving the highest accuracy of 99.2%. The grasping times varied, from 11.76 seconds for the red triangle to 15.02 seconds for the green square. Additionally, the robot's performance remained consistent across different shapes and colors, with pick positions varying slightly for each object.

This bar chart in Figure 11 shows the average accuracy percentages categorized by color and shape. For each color—blue, green, red, and yellow—the average accuracy is presented for three shapes: circle, square, and triangle. Overall, the accuracy values are very close across all groups, with most shapes achieving above 95% accuracy. Triangles generally have the highest average accuracy within each color group, followed closely by squares and circles. The chart suggests that accuracy is consistently high regardless of color or shape, with only slight variations.

Table 4. Results of Dobot Magician's workpiece grasping test according to shape and color

Color	Shape	Count	Success	Time (s)	Accuracy (%)	Pick position
Red	Square	1	Yes	14.31	97.8	(219, -155)
Blue	Circle	1	Yes	14.35	96.1	(270, -127)
Green	Circle	1	Yes	12.98	98.6	(273, -117)
Blue	Square	1	Yes	14.35	98.2	(206, -148)
Red	Circle	1	Yes	14.56	95.2	(212, -144)
Yellow	Square	1	Yes	13.71	97.4	(268, -120)
Red	Triangle	1	Yes	13.39	96.5	(275, -130)
Yellow	Triangle	1	Yes	13.07	95.9	(274, -132)
Green	Triangle	1	Yes	12.71	97.4	(241, -72)
Yellow	Circle	1	Yes	12.04	95.7	(263, -71)
Green	Triangle	2	Yes	13.43	96.3	(290, -144)
Blue	Triangle	1	Yes	12.04	97.3	(235, -98)
Blue	Circle	2	Yes	12.10	97.6	(262, -73)
Red	Triangle	2	Yes	13.02	99.2	(264, -144)
Red	Square	2	Yes	13.95	95.6	(271, -146)
Green	Square	1	Yes	15.02	98.4	(211, -157)
Green	Circle	2	Yes	13.35	98.1	(209, -92)
Blue	Square	2	Yes	13.29	97.1	(267, -86)
Yellow	Square	2	Yes	12.40	96.2	(266, -149)
Red	Circle	2	Yes	14.32	96.8	(285, -143)
Green	Triangle	3	Yes	13.05	98.9	(225, -93)
Yellow	Circle	2	Yes	13.33	96.2	(280, -139)
Yellow	Triangle	2	Yes	13.66	96.6	(237, -148)
Red	Triangle	3	Yes	13.62	97.2	(158, -136)
Blue	Triangle	2	Yes	12.66	97.6	(215, -160)
Red	Square	3	Yes	13.04	98.3	(264, -70)
Green	Circle	3	Yes	13.32	95.9	(262, -74)
Red	Triangle	4	Yes	11.76	98.4	(267, -77)
Green	Triangle	4	Yes	13.03	96.1	(209, -147)
Blue	Square	3	Yes	14.26	97.0	(272, -132)
Red	Circle	3	Yes	13.96	95.0	(268, -125)
Blue	Circle	3	Yes	14.09	96.3	(276, -136)
Yellow	Square	3	Yes	13.12	98.4	(207, -74)
Green	Square	2	Yes	14.58	97.9	(270, -135)
Yellow	Circle	3	Yes	13.72	95.9	(214, -133)
Blue	Triangle	3	Yes	13.08	99.1	(209, -159)
Yellow	Triangle	3	Yes	12.41	96.4	(263, -72)

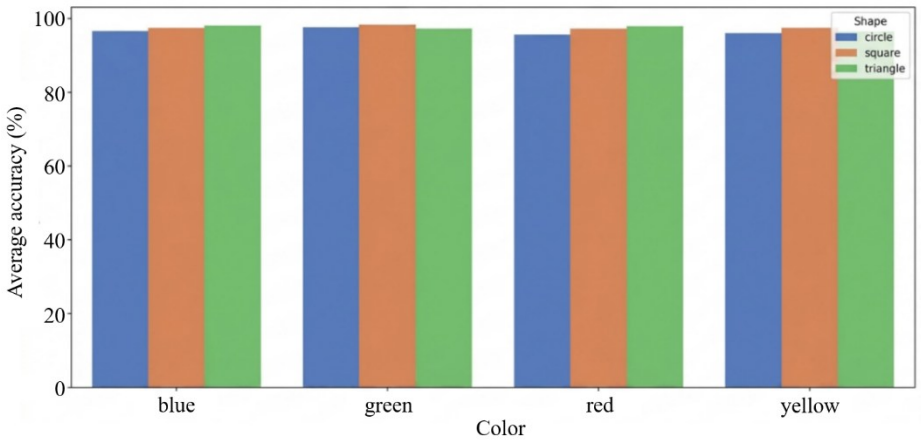


Figure 11. Average accuracy by color and shape

Figure 12 shows the average time taken for each task, categorized by color and shape. The four main colors are blue, green, red, and yellow. For each color, the average times for three shapes—circle, square, and triangle—are displayed. Overall, squares tend to take the longest time, especially for the green and blue colors. Triangles generally require the shortest time across most colors, except for red, where circles take the longest time. In yellow, the average times for all three shapes are quite similar. This chart highlights how the time taken varies depending on both shape and color.

Figure 13 shows the relationship between accuracy and time taken, with each point representing a data set categorized by color and shape. The size of each point indicates the count, or how many times that

data set occurs. Generally, data points with a time of less than 2 seconds tend to have higher accuracy. Triangles mostly show high accuracy with shorter times, while squares tend to take longer and often have lower accuracy. Larger points, indicating higher counts, are spread across longer time ranges with moderate to lower accuracy. This suggests that taking more time does not necessarily result in better accuracy.

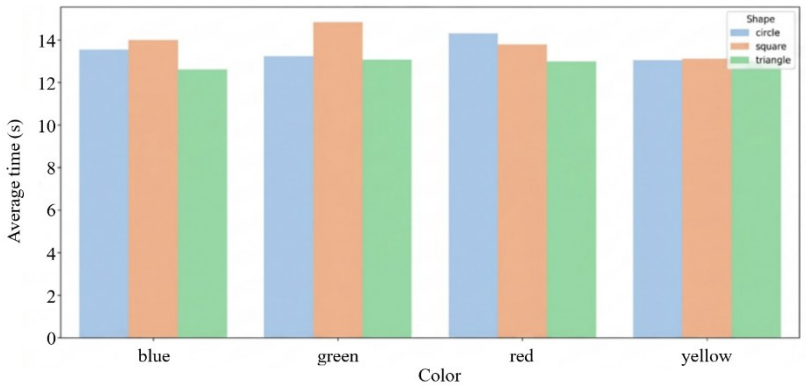


Figure 12. Average time by color and shape

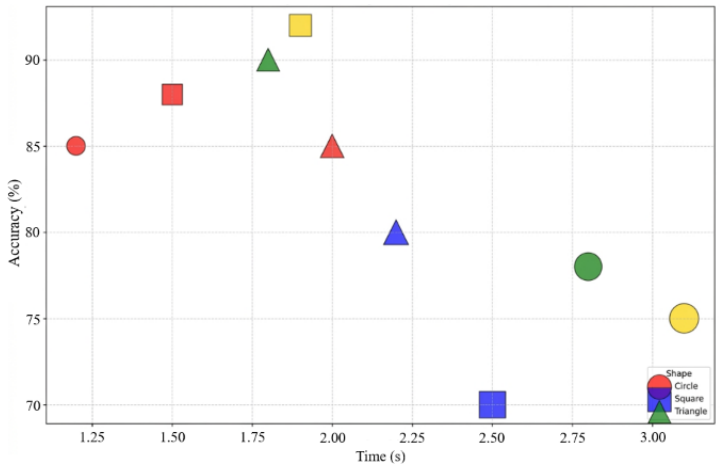


Figure 13. Accuracy vs time of color, shape, and count

4.2. Discussion

One of the main advantages is more accurate color separation compared to RGB, as HSV separates color information from brightness. This allows for more precise color detection, even under uneven lighting conditions. The hue component specifically represents the main color (e.g., red, green, blue), making it easier to classify objects by color. Another advantage is fast processing with OpenCV, which provides built-in functions such as cv2.inRange (), cv2.findContours (), and cv2.approxPolyDP (). These tools enable rapid and efficient detection of both color and shape. HSV and OpenCV also support accurate shape detection. Basic geometric shapes like circles, squares, and triangles can be identified by analyzing object contours and counting the number of vertices. For example, a shape with 3 vertices is classified as a triangle, 4 vertices as a square, and a smooth, curved contour as a circle. The system is also easy to implement without the need for AI models, making it ideal for embedded applications and real-time robotic systems, such as those using the Dobot robotic arm. Lastly, HSV-based detection offers flexibility, as users can adjust the HSV range to adapt to different lighting conditions. This tuning helps maintain high detection accuracy in various environments. The accuracy of object detection using HSV and OpenCV depends on several key factors, including lighting conditions, camera quality, HSV range settings, and the clarity of the object's shape.

Based on typical experiments and real-world applications:

i) Color detection (HSV)
   - Under normal lighting: accuracy ranges from 90–98%.
   - Under variable lighting or with shadows: accuracy drops to 80–90%.
   - With well-calibrated HSV ranges and controlled lighting, accuracy can reach up to 99%.

ii) Shape detection (contours + approxPolyDP)
   - For clear, non-overlapping shapes: accuracy is around 95–98%.
   - For distorted or unclear shapes: accuracy is approximately 85–90%.

iii) Combined color and shape detection
   - Using both color and shape filtering together significantly reduces false positives.
   - Overall detection accuracy is typically between 92–97% when lighting is stable, and the system uses a mid-range or better camera.

5. CONCLUSION

The object-sorting system presented in this study demonstrates an effective integration of a robotic arm with a classification model for detecting and sorting objects based on shape and color. The system utilizes traditional computer vision techniques, including HSV color space and OpenCV, for real-time detection and classification. The results show that the Dobot Magician robotic arm was able to successfully grasp objects with high accuracy (ranging from 95.0% to 99.2%) and within reasonable time limits (ranging from 11.76 to 15.02 seconds). This highlights the system's robustness in various conditions and its ability to handle a range of object shapes and colors efficiently. Moreover, the system's performance remains consistent across different object categories, demonstrating the reliability of the integration between the vision system and the robotic arm. While some variations in time and accuracy were observed depending on the shape and color, the system's overall efficiency makes it suitable for embedded applications and real-time robotic tasks. Future work could focus on further improving detection accuracy under varying lighting conditions, enhancing the shape detection algorithm to handle more complex or irregular shapes, and optimizing the system's performance for larger-scale sorting operations. Additionally, exploring the use of AI-based techniques, such as deep learning models for object classification, could enhance the flexibility and accuracy of the system, especially in more dynamic environments. Furthermore, the integration of multiple robotic arms or a conveyor belt system could allow for more advanced sorting operations, increasing the system's efficiency and scalability in industrial applications. Overall, this system provides a solid foundation for automated object sorting in industrial and commercial applications, with potential for future improvements and expansion.

ACKNOWLEDGMENTS

This research has been carried out at the Laboratories of Mechatronics and Mechanical Engineering, Faculty of Engineering, Rajamangala University of Technology Isan, Khon Kaen Campus. We would like to express our appreciation for the support of staff of the Department of Mechatronics Engineering and Mechanical Engineering.

FUNDING INFORMATION

This research received no external funding.

AUTHOR CONTRIBUTIONS STATEMENT

This journal uses the Contributor Roles Taxonomy (CRediT) to recognize individual author contributions, reduce authorship disputes, and facilitate collaboration.

Name of Author	C	M	So	Va	Fo	I	R	D	O	E	Vi	Su	P	Fu
Tossapol Jangnoi	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Thewin Sakunbunyong	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Viroch Sukontanakarn	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	
Tanawat Chalardsakul	✓		✓		✓		✓	✓		✓				

C : **C**onceptualization
M : **M**ethodology
So : **So**ftware
Va : **Va**lidation
Fo : **Fo**rmal analysis
I : **I**nvestigation
R : **R**esources
D : **D**ata Curation
O : Writing - **O**riginal Draft
E : Writing - Review & **E**diting
Vi : **Vi**sualization
Su : **Su**pervision
P : **P**roject administration
Fu : **Fu**nding acquisition

CONFLICT OF INTEREST STATEMENT

The authors declare that they have no conflict of interest.

INFORMED CONSENT

We have obtained informed consent from all individuals included in this study.

ETHICAL APPROVAL

Ethical approval was not required for this study, as it involved only publicly available secondary data.

DATA AVAILABILITY

The data supporting the findings of this study are available from the corresponding author, [TS], upon reasonable request.

REFERENCES

[1] B. Schmidt and L. Wang, "Automatic work objects calibration via a global–local camera system," *Robotics and Computer-Integrated Manufacturing*, vol. 30, no. 6, pp. 678–683, Dec. 2014, doi: 10.1016/j.rcim.2013.11.004.

[2] 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 Robotics and Control*, vol. 4, no. 6, pp. 769–779, Nov. 2023, doi: 10.18196/jrc.v4i6.20087.

[3] A. Nasiri, M. Omid, and A. T. Garavand, "An automatic sorting system for unwashed eggs using deep learning," *Journal of Food Engineering*, vol. 283, Oct. 2020, doi: 10.1016/j.jfoodeng.2020.110036.

[4] X. Zhang *et al.*, "Design and operation of a deep-learning-based fresh tea-leaf sorting robot," *Computers and Electronics in Agriculture*, vol. 206, Mar. 2023, doi: 10.1016/j.compag.2023.107664.

[5] S. W. Sidehabi, M. F. Azis, and M. Asbar, "Development of a color-based image recognition system for robotic sorting and picking," *INTEK: Jurnal Penelitian*, vol. 11, no. 2, Oct. 2024, doi: 10.31963/intek.v11i2.5009.

[6] M. K. Hossen, S. M. Bari, P. P. Barman, R. Roy, and P. K. Das, "Application of Python-OpenCV to detect contour of shapes and colour of a real image," *International Journal of Novel Research in Computer Science and Software Engineering*, vol. 9, pp. 20–25, 2022.

[7] M. A. Al-Noman, A. N. Eva, T. B. Yeahyea, and R. Khan, "Computer vision-based robotic arm for object color, shape, and size detection," *Journal of Robotics and Control*, vol. 3, no. 2, pp. 180–186, Feb. 2022, doi: 10.18196/jrc.v3i2.13906.

[8] M. Di Capua, A. Ciaramella, and A. De Prisco, "Machine learning and computer vision for the automation of processes in advanced logistics: the integrated logistic platform (ILP) 4.0," *Procedia Computer Science*, vol. 217, pp. 326–338, 2023, doi: 10.1016/j.procs.2022.12.228.

[9] S. Vasudevan *et al.*, "Machine vision and robotics for primary food manipulation and packaging: a survey," *IEEE Access*, vol. 12, pp. 152579-152613, 2024, doi: 10.1109/ACCESS.2024.3479781.

[10] O. M. Banur, B. K. Patle, and S. Pawar, "Integration of robotics and automation in supply chain: a comprehensive review," *Robotic Systems and Applications*, vol. 4, no. 1, pp. 1–19, Jun. 2024, doi: 10.21595/rsa.2023.23349.

[11] V. Cong, L. Hanh, L. Phuong, and D. Duy, "Design and development of robot arm system for classification and sorting using machine vision," *FME Transactions*, vol. 50, no. 2, pp. 181–181, 2022, doi: 10.5937/fme2201181C.

[12] M. T. Shahria, M. S. H. Sunny, M. I. I. Zarif, J. Ghommam, S. I. Ahamed, and M. H. Rahman, "A comprehensive review of vision-based robotic applications: current state, components, approaches, barriers, and potential solutions," *Robotics*, vol. 11, no. 6, Dec. 2022, doi: 10.3390/robotics11060139.

[13] A. Al Fahim, M. M. Rahman, H. Kabir, and S. Bari, "Design and fabrication of automatic weight, color and height based sorting system," *Journal of Integrated and Advanced Engineering*, vol. 3, no. 2, pp. 111–126, Sep. 2023, doi: 10.51662/jiae.v3i2.101.

[14] S. Xu, J. Wang, W. Shou, T. Ngo, A.-M. Sadick, and X. Wang, "Computer vision techniques in construction: a critical review," *Archives of Computational Methods in Engineering*, vol. 28, no. 5, pp. 3383–3397, Aug. 2021, doi: 10.1007/s11831-020-09504-3.

[15] T. Diwan, G. Anirudh, and J. V. Tembhurne, "Object detection using YOLO: challenges, architectural successors, datasets and applications," *Multimedia Tools and Applications*, vol. 82, no. 6, pp. 9243–9275, 2023, doi: 10.1007/s11042-022-13644-y.

[16] L. Yang *et al.*, "Computer vision models in intelligent aquaculture with emphasis on fish detection and behavior analysis: a review," *Archives of Computational Methods in Engineering*, vol. 28, no. 4, pp. 2785–2816, Jun. 2021, doi: 10.1007/s11831-020-09486-2.

[17] V. Sampath, I. Maurtua, J. J. A. Martín, and A. Gutierrez, "A survey on generative adversarial networks for imbalance problems in computer vision tasks," *Journal of Big Data*, vol. 8, no. 1, Dec. 2021, doi: 10.1186/s40537-021-00414-0.

[18] Z. Ren, F. Fang, N. Yan, and Y. Wu, "State of the art in defect detection based on machine vision," *International Journal of Precision Engineering and Manufacturing-Green Technology*, vol. 9, no. 2, pp. 661–691, Mar. 2022, doi: 10.1007/s40684-021-00343-6.

[19] I. R.- Conde, C. Campos, and F. F.- Riverola, "Optimized convolutional neural network architectures for efficient on-device vision-based object detection," *Neural Computing and Applications*, vol. 34, no. 13, pp. 10469–10501, Jul. 2022, doi: 10.1007/s00521-021-06830-w.

[20] S. A. Singh and K. A. Desai, "Automated surface defect detection framework using machine vision and convolutional neural networks," *Journal of Intelligent Manufacturing*, vol. 34, no. 4, pp. 1995–2011, Apr. 2023, doi: 10.1007/s10845-021-01878-w.

[21] K. M. Abubeker, Abhijit, S. Akhil, V. K. A. Kumar, and B. K. Jose, "Computer vision-assisted real-time bird eye chili classification using YOLO V5 framework," *Journal of Artificial Intelligence and Technology*, Nov. 2023, doi: 10.37965/jait.2023.0251.

[22] P. F. Vidal, D. Gómez, J. Castro, and J. Montero, "New aggregation approaches with HSV to color edge detection," *International Journal of Computational Intelligence Systems*, vol. 15, no. 1, Sep. 2022, doi: 10.1007/s44196-022-00137-x.

[23] H. Li, C. Dong, X. Jia, S. Xiang, and Z. Hu, "Design of automatic sorting and transportation system for fruits and vegetables based on Dobot Magician robotic arm," in *2023 2nd International Symposium on Control Engineering and Robotics (ISCER)*, Feb. 2023, pp. 229–234, doi: 10.1109/ISCER58777.2023.00045.

[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, Nov. 2021, doi: 10.3390/machines9120302.
[25] Y. Liu, H. Xu, D. Liu, and L. Wang, "A digital twin-based sim-to-real transfer for deep reinforcement learning-enabled industrial robot grasping," *Robotics and Computer-Integrated Manufacturing*, vol. 78, Dec. 2022, doi: 10.1016/j.rcim.2022.102365.
[26] N. K. Al-Karkhi, W. T. Abbood, E. A. Khalid, A. N. J. Al-Tamimi, A. A. Kudhair, and O. I. Abdullah, "Intelligent robotic welding based on a computer vision technology approach," *Computers*, vol. 11, no. 11, 2022, doi: 10.3390/computers11110155.
[27] P. Anggraeni, W. Candra, A. Sunarya, and D. Safitri, "Practical validation on simulation of multi Dobot Magician trajectory contour tracking," in *5th International Conference on Applied Science and Technology on Engineering Science*, 2022, pp. 965–970, doi: 10.5220/0011973200003575.
[28] A. Singh, V. Kalaichelvi, and R. Karthikeyan, "A survey on vision guided robotic systems with intelligent control strategies for autonomous tasks," *Cogent Engineering*, vol. 9, no. 1, Dec. 2022, doi: 10.1080/23311916.2022.2050020.
[29] A. Hasan and A. Maynes, "AI-enhanced DOBOT Magician for classroom education: hand gesture control for hazardous material handling simulation," in *2025 ASEE -GSW Annual Conference*, 2025, pp. 1-10, doi: 10.18260/1-2--55028.

BIOGRAPHIES OF AUTHORS

Tossapol Jangnoi received the B.Eng. degree in Mechanical Engineering from Rajamangala University of Technology Isan, Khon Kaen, Thailand, in 2006. The M.Eng. degree in Mechanical Engineering from King Mongkut's University of Technology Thonburi, Bangkok, Thailand, in 2009, and D.Eng. in Mechanical Engineering from King Mongkut's Institute of Technology, Ladkrabang, in 2019. He is currently a lecturer in the Mechanical Engineering, Faculty of Engineering, Rajamangala University of Technology, Isan Khon Kaen Campus, Thailand. His current research interests include automatic control, pneumatics and hydraulic system, automation system, robotics, IIOT, and computer-aided design. He can be contacted at email: tossapol.ja@rmuti.ac.th.

Thewin Sakunbunyong received the B.Eng. degree in Mechanical Engineering from Rajamangala University of Technology Thanyaburi, Phathumtany, Thailand, in 1996, and the M.Eng. degree in Mechanical Engineering from Srinakharinwirot University, Bangkok, Thailand, in 2009. He is currently a lecturer in the Field of Mechatronics Engineering, Faculty of Engineering, Rajamangala University of Technology, Isan Khon Kaen Campus, Thailand. His current research interests include programmable logic controllers, automation and robotics, pneumatic and hydraulic systems, and electric motor drives. He can be contacted at email: thewin.sa@rmuti.ac.th.

Viroch Sukontanakarn is a lecture in Mechatronics Engineering at the Rajamangala University of Technology Isan Khon Kaen Campus, Khon Kaen, Thailand. He received M.Eng. in Electric Power System Management and D.Eng. in Mechatronics from the Asian Institute of Technology, in 1998 and 2011, respectively. He has been an associate professor at the Field of Mechatronics Engineering, Faculty of Engineering, Rajamangala University of Technology, Isan Khon Kaen Campus, Thailand since 2002. His current research interests are power electronics, electrical power systems, microcontrollers, robotics, programmable logic controllers, and electric motor drives. He can be contacted at email: viroch.su@rmuti.ac.th.

Tanawat Chalardsakul is an associate professor at the Mechatronics Engineering, Faculty of Engineering, Rajamangala University of Technology, Isan Khon Kaen Campus, Thailand. He received an M.Eng. in Electric Power System and a Ph.D. in Buddhist studies from Mahachulalongkornrajavidyalaya University. His research interests are mechanical systems, renewable energy, and robot design. He can be contacted at email: tanawat.ca@rmuti.ac.th.