Journal of Electrical Engineering, Energy, and Information Technology (J3EIT)
Vol. 11, No. 2, August 2023, pp. 73 - 83
ISSN: 3026-1856, DOI: 10.26418/j3eit.v11i2.68483

Color and Size Sorting System with An ESP32-Based Robot Arm

Muhamad Gilang Satria Adiguna[1)], Muhammad Saleh[2)], Elang Derdian Marindani[2)]
[1]Electrical Engineering Study Program, University of Tanjungpura, Pontianak, Indonesia
[2]Department of Electrical, Faculty of Engineering, University of Tanjungpura, Pontianak, Indonesia

Article Info

Received Aug 08, 2023
Revised Aug 29, 2023
Accepted Sep 07, 2023

Keywords:

Robot Arm
Inverse Kinematics
Camera
Color
Size

ABSTRACT

Robotics has seen an explosive increase in usage in the past few decades owing to its abundance of benefits and automation capabilities. The use of robotics is especially prominent in industries where the demand for precise and automatic robots in order to increase production efficiency is high. One of the robots commonly used in these industries is the arm robot model, which is a machine designed specifically to mimic some of the human arm's capabilities in order to execute certain tasks over and over again, either autonomously or not. The development of a simple and replicable robot arm that is capable of autonomously performing certain tasks will certainly be beneficial to the application of robotics throughout the country. This paper discusses the design of a robotic arm with five degrees of freedom capable of picking up and placing objects with three different colors and sorting the objects based on their size, starting from the largest to the smallest. The inclusion of an additional degree of freedom allows the robot to adjust the rotation of its grip to properly grab rotated objects. With an ESP32-CAM acting as its eyes, the robot is aided by a Python program run remotely on a separate Laptop in order to analyze video stream uploaded by the camera to determine the coordinates, color, size, and rotation of objects placed within the Detection Zone. The implementation of inverse kinematics equations run on an ESP32 microcontroller allows the robot to reach arbitrary coordinates specified by the Python program. Testing revealed that the resulting robot is able to perform the tasks it was designed to, namely recognizing and sorting colored boxes of different colors and sizes, with an accuracy 100% in recognition tasks and an accuracy of 91.5% in sorting tasks.

Corresponding Author:
Muhammad Gilang Satria Adiguna
Email: muhgilangsa@student.untan.ac.id

1. INTRODUCTION

The advancement of modern technology has gotten to the point where humans are able to create machines that mimic human movements to varying degrees in order to help humans perform certain tasks. The Oxford English Dictionary defines a robot as "a machine resembling a human being and able to replicate certain human movements and functions automatically." [1] Some of these machines are designed specifically to mimic the versatility of the human arm, allowing the robot, dubbed "robotic arm" to perform tasks requiring some degree of precision. These robot arms are commonly found in modern factories, where they act as autonomous, tireless workers that with extremely high reliability and efficiency compared to their human counterparts. Robot arms commonly move by utilizing servo motors, actuators that are capable of remembering their current position and move to other, specified position. [2]

Given the advantages a robot arm could provide to the efficiency of industrial processes, it is worthwhile to invest time and effort into creating a robot arm with high flexibility and a variety of functions that is capable of autonomously sorting objects based on certain characteristics, namely size and color. By implementing inverse kinematics, it is possible to allow the robot to pick up objects placed in arbitrary locations within the robot's reach. The utilization of Computer Vision allows the robot to monitor a certain area and act autonomously according to changes that happens in this area. [3]

This research references a number of previous relevant and similar researches as a basis upon which to build the robot arm system. The following contains an enumeration of previous researches that have influenced

this project. Work referenced in number [4] was done to create an automatic robot arm with the capability to sort objects according to their color using a TCS 3200 sensor. A robot arm was designed in work [5] with a single degree of freedom on which it pivoted to grab and detect object's color before placing said object in its appropriate storage zone. Work [6] delved into applying inverse kinematics in order to control a 3-DOF robot arm. The work referenced in number [7] was done to design a 5 DOF robot arm capable of distinguishing objects of three different colors and placing them in their respective storage areas. Utilizing inverse kinematics, this robot arm was able to pick up objects placed in arbitrary positions without additional inputs from the user as long as the objects were located on the robot's detection area.Work [8] was a research project in which a robot arm that was capable of moving objects that were placed in arbitrary locations within the robot's reach was designed. The objects used in this research were cylindrical and triangular in shape, between which the robot was also able to properly distinguish. This three-degrees-of-freedom robot utilizes a three DOF inverse kinematics to define the joint parameters required to move the gripper to grab objects. Powering this robot are the microcontroller Raspberry Pi 3 Model B and the OpenCM 9.04.

To further develop the projects created in these previous researches, this research aims to create a robot arm with the capability to accurately sort objects not only based on color, but also size. This project uses an ESP32 as the brain of the robot, a microcontroller with dual core processor developed by Espressif Systems. [9]. The robot also uses an ESP32-CAM equipped with OV2640 camera module as its eyes. This robot is designed to pick up objects with three different colors placed arbitrarily on the detection zone. This feature was allowed by the utilization of a 5-DOF inverse kinematics system run on the ESP32 microcontroller. As an added feature, this robot arm is also able to properly pick up objects placed with arbitrary orientation by rotating the gripper to match the object's orientation. This research uses Python, a high-level programming language used widely around the world developed by Guido van Rossum known for its intuitiveness and ease of use [10], as the primary language to build the robot arm's software.

2. RESEARCH METHOD

This section contains the overview of the system, the robot arm's physical design, the inverse kinematics equations used for the robot arm, and the software running the robot arm.

2.1 General Overview of the System

Figure 1 below explains the general overview of the system.

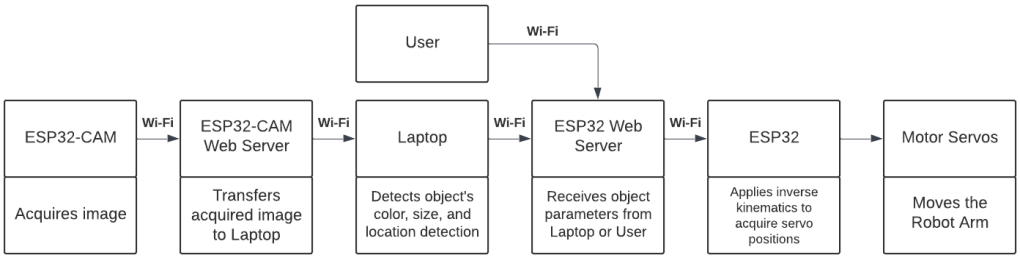

Figure 1. General Overview of the System

As seen on Figure 1, the system consists of an ESP32-CAM equipped with OV2640 camera module, a laptop or a computer on which the Python program is run, an ESP32 microcontroller, as well as three MG90s servos coupled with three MG996R servos to move the robot arm. Upon activation, the camera monitors a certain area containing the detection zone upon which objects are placed. This footage is streamed to ESP32-CAM's own web server via Wi-Fi. The Python program run on either a laptop or a computer then grabs and analyzes each frame of this stream to detect boxes placed on the detection zone. In this case, the camera's only function is to provide raw footage for the program to analyze. Once an object is detected by the program, it finds the characteristics of said object including the object's location (X, Y), its orientation (R), its color, and its size. The program subsequently creates a command to be sent to the ESP32 based on these characteristics. The ESP32 microcontroller receives commands from the Python program through its own ESP32 web server that takes specific URLs and translates them into parameters for the inverse kinematics equations, which output certain values for the servo motors to move the robot arm into position to grab the object. As an additional feature, it is possible to control the robot arm by manually inputting certain X, Y, and Z values to the ESP32 web server.

2.2 Software Design

The software handles object detection, which include detection of the object's size, color, and location. In order to allow the program to accurately determine the location of the object, a detection zone measuring 290mm by 200mm is used in this research. The program first detects this detection zone in the raw footage provided by the camera before projecting the detection zone onto an image with a dimension of 290 pixels by 200 pixels with perspective transform. This is done so that a distance of one pixel in the image is roughly equal to a distance of one millimeter in the real world. Contour approximation, a method to simplify the number of edges a contour has, [11] is applied here in order to extract a contour with four edges to use for perspective transform. Figure 2 shows this process.

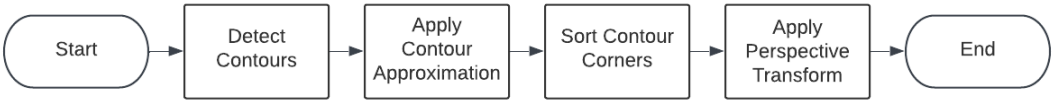

Figure 2. Detection Zone Extraction Process

After the detection zone has been extracted, it's important to inform the program of the location of the detection zone relative to the robot arm. As seen in Figure 3, the pixel coordinates system doesn't match the real-world coordinate system used in the robot arm's inverse kinematics equations. Therefore, it's necessary to provide the necessary X and Y translational values to reconcile these two coordinate systems, as stated in Equation 1.

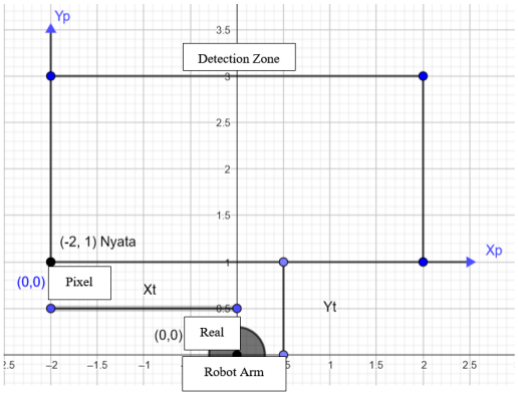

Figure 3. Illustration of Detection Zone's Location Relative to the Robot Arm's

$$\begin{bmatrix} Xp \\ Yp \end{bmatrix} + \begin{bmatrix} Xt \\ Yt \end{bmatrix} = \begin{bmatrix} Xn \\ Yn \end{bmatrix} \quad (1)$$

p = pixel; t = translational; n = real

The program then has to detect the characteristics of the objects placed on the detection zone. This process is explained in Figure 4.

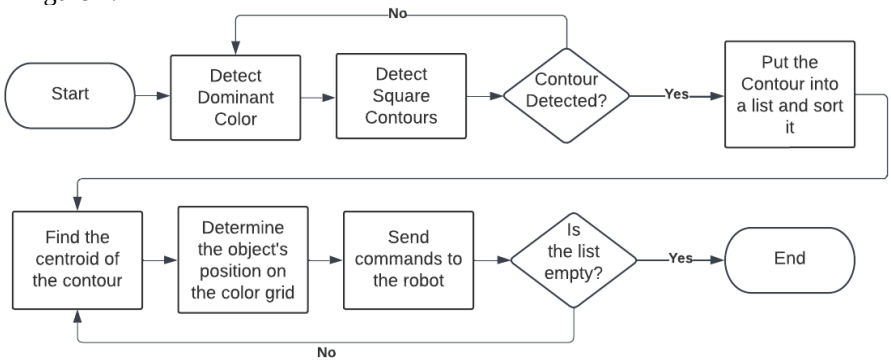

Figure 4. Object Detection Process

First, the program detects the dominant color present in the image. The program then applies a mask over the image to isolate one specific color and then draws contour over the objects to determine their size. The program also determines where to put each object based on its size and the size of objects already on the storage area. After sorting these objects from largest to smallest, the program determines the coordinates of the centroid of each object before sending these coordinates to the robot arm through the ESP32 web server for the inverse kinematics equations. This object detection method incorporates the usage of Canny's Edge Detection method, a method that measures the change in intensity between pixels (gradients) in order to detect edges in an image. [12]. This edge detection method also utilizes non-maxima suppression and hysteresis thresholding to increase accuracy. [13]

2.3 Robot Arm Design

The chassis of the robot arm is 3D printed using plastic as its primary material. The robot arm consists of five different parts: the base that supports the entire robot arm, a first link connecting the base and the second link, the second link connecting the first link and the third link, and the third link which connects the second link and the gripper, and the gripper itself. The robot arm has five degrees of freedom and uses six servo motors to move, three MG996R servos used for the base, first joint, and second joint, as well as three MG90s servos used for the third joint, gripper orientation joint, and the gripper joint. Figure 5 contains the digital design of the robot arm.

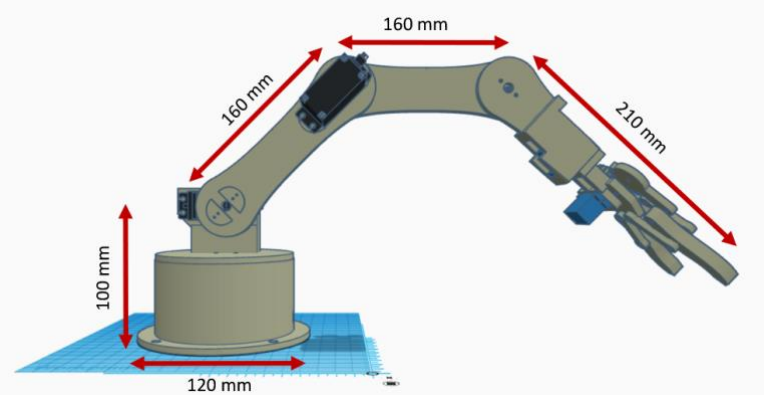

Figure 5. Robot Arm Design

2.4 Robot Arm Inverse Kinematics

Inverse kinematics is used to allow the robot arm to translate coordinates into movement. There are two parts of this process as illustrated in Figure 6 and Figure 7. The first part deals with the movement of the base joint seen from a bird's eye point of view, while the second deals with the movement of the first, second, and third joints seen from the side. The inverse kinematics equations are made based on Figure 6 and Figure 7, resulting in Equation 2 through Equation 12. The resulting values of these equations are then written to their corresponding servo motors in order to achieve the desired movement and robot arm position. [14] [15]

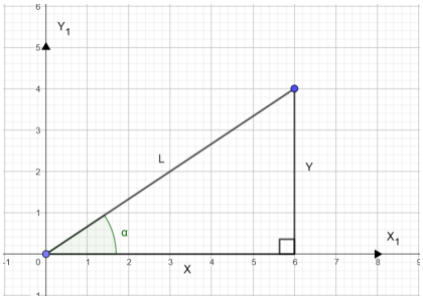

Figure 6. First Part of Robot Arm's Inverse Kinematics Calculation

$$\alpha = \tan^{-1}\frac{Y}{X} \quad (2)$$

$$L = \sqrt{X^2 + Y^2} \quad (3)$$

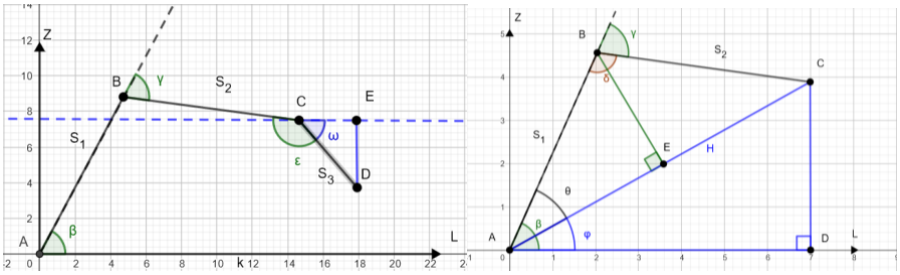

Figure 7. Second Part of the Robot Arm's Inverse Kinematics Calculation

$$wL = L - S_3 \times \cos\omega \quad (4)$$

$$wZ = Z - S_3 \times \sin\omega \quad (5)$$

$$H = \sqrt{L^2 + Z^2} \quad (6)$$

$$\theta = \cos^{-1}\frac{\frac{H}{2}}{S_1} \quad (7)$$

$$\varphi = \tan^{-1}\frac{Z}{L} \quad (8)$$

$$\beta = \theta + \varphi \quad (9)$$

$$2\theta + \delta = 180°$$

$$\delta = 180° - 2\theta \quad (10)$$

$$\gamma = 180° - \delta$$

$$\gamma = 180° - (180° - 2\theta)$$

$$\gamma = 2\theta \quad (11)$$

$$\varepsilon = 180° - (-\omega + \beta - \gamma) \quad (12)$$

2.5 Wiring Diagram

Below is Figure 8, which contains the wiring diagram of the system created in this project.

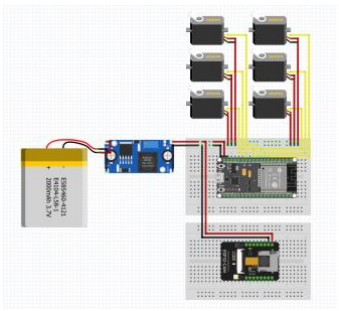

Figure 8. Wiring Diagram

3. TESTING AND ANALYSIS

In order to evaluate the performance of the robot arm, a number of tests was performed on it. The tests are as follows: Object Detection and Object Relocation Test, Object Relocation Test with Arbitrary Object Orientation, and Object Sorting Test, which will be explained in their respective sections. All of these tests were conducted with object storage coordinates presented in Table 1. In the table, each color's storage zone is divided into three sections. The X and Y values denotes the location of the section, while the R value denotes the desired object placement orientation.

Table 1. Object Storage Coordinates

Color	Section 3 Coordinates (X, Y, R)	Section 2 Coordinates (X, Y, R)	Section 1 Coordinates (X, Y, R)
Blue	(-200, 140, 90)	(-200, 70, 90)	(-200, 0, 90)
Red	(-100, 320, 90)	(0, 320, 90)	(100, 320, 90)
Green	(210, 140, 90)	(210, 70, 90)	(210, 0, 90)

The test is conducted by placing objects in front of the robot arm, in view of the camera. The robot arm will then move the objects to their appropriate location based on each object's size and color. Figure 9 shows a snippet during testing.

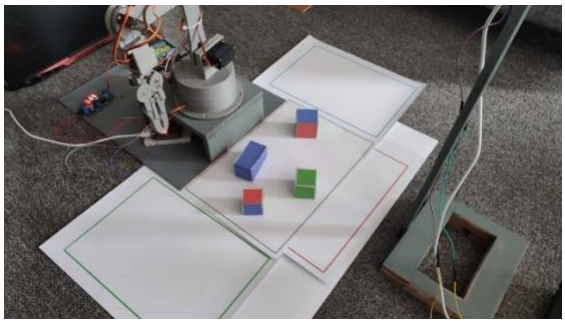

Figure 9. Robot Arm Testing

3.1 Object Detection and Object Relocation Test

This test was conducted to evaluate the robot arm's ability to correctly identify colored objects and place them into their respective storage areas. In this test, a single object was placed upon the detection zone, and the reaction of the robot arm was observed. This test was done ten times for each color, for the total of 30 tests done. Table 2, 3, and 4 hold the results of this test. Out of the 30 tests done, the robot was successful in all of them.

Table 2. Blue Object Detection and Relocation Test Result

Test	Object Coordinates (X, Y, R)	Detected Color	Result
1	(-36, 153, 147)	Blue	Success
2	(5, 160, 90)	Blue	Success
3	(-4, 181, 154)	Blue	Success
4	(-8, 173, 90)	Blue	Success
5	(24, 193, 53)	Blue	Success
6	(-21, 176, 82)	Blue	Success
7	(-5, 153, 29)	Blue	Success
8	(88, 130, 25)	Blue	Success
9	(100, 146, 26)	Blue	Success
10	(73, 139, 141)	Blue	Success

Table 3. Red Blue Object Detection and Relocation Test Result

Test	Object Coordinates (X, Y, R)	Detected Color	Result
1	(-53, 170, 21)	Red	Success
2	(39, 189, 65)	Red	Success
3	(1, 164, 79)	Red	Success
4	(-81, 158, 152)	Red	Success
5	(-36, 166, 75)	Red	Success
6	(32, 184, 42)	Red	Success
7	(-49, 151, 106)	Red	Success
8	(-16, 148, 147)	Red	Success
9	(-106, 222, 29)	Red	Success
10	(-52, 181, 17)	Red	Success

Table 4. Green Object Detection and Relocation Test Result

Test	Object Coordinates (X, Y, R)	Detected Color	Result
1	(-44, 178, 34)	Green	Success
2	(-95, 205, 7)	Green	Success
3	(-99, 165, 45)	Green	Success
4	(37, 171, 25)	Green	Success
5	(50, 169, 157)	Green	Success
6	(-67, 158, 5)	Green	Success
7	(13, 137, 36)	Green	Success
8	(-86, 126, 67)	Green	Success
9	(-84, 199, 71)	Green	Success
10	(81, 161, 67)	Green	Success

3.2 Object Relocation Test with Arbitrary Object Orientation

In this test, the same object was placed in different orientation in order to evaluate the arm's ability to pick up objects with different orientations. For this test, a rectangular prism was used in place of the usual cube, which until this point have been used in previous tests, so that the robot's ability to adjust its gripper's orientation to match the orientation of an object could be better observed. This test was done 15 times and the results can be found in Table 5. The robot was observed to have completed all 15 tests successfully.

Table 5. Object Orientation Detection Test Result

Test	Object Coordinates (X, Y)	Object Orientation (R)	Result
1	(-65, 176)	141	Success
2	(15, 167)	56	Success
3	(-105, 224)	116	Success
4	(-4, 236)	178	Success
5	(-93, 124)	139	Success
6	(6, 160)	86	Success
7	(89, 184)	90	Success
8	(-20, 120)	171	Success
9	(-10, 176)	85	Success
10	(69, 149)	180	Success
11	(-3, 188)	159	Success
12	(-12, 225)	160	Success
13	(3, 201)	142	Success
14	(-8, 150)	146	Success
15	(-10, 231)	175	Success

3.3 Object Sorting Test

In Object Sorting Test, the robot arm's ability to correctly identify objects' size and sort them accordingly was evaluated. There was two parts to this test: one where the test was done with empty storage area and another where the same test was done while the storage area was full, the former only requiring the arm to sort the three objects on the detection zone while the latter required the arm to first make space to move the new object introduced on the detection zone into.

3.3.1 Object Sorting Test while the Storage is Empty

In this test, the storage is left empty while three same-colored objects of differing sizes ranging from small to large were introduced to the detection zone. The robot arm would then sort these objects in descending order starting from largest to smallest. This test is done 10 times for each color and the results can be found in Table 6 through 8. Out of the 30 tests conducted, the robot was able to complete 27 of them successfully.

Table 6. Blue Object Sorting Test Result (Empty Storage)

Test	Section 3	Section 2	Section 1	Result
1	Large	Medium	Small	Success
2	Large	Medium	Small	Success
3	Large	Medium	Small	Success
4	Large	Medium	Small	Success
5	Large	Medium	Small	Success
6	Large	Medium	Small	Success
7	Large	Medium	Small	Success
8	Large	Medium	Small	Success
9	Large	Small	Medium	Failed
10	Large	Medium	Small	Success

Table 7. Red Object Sorting Test Result (Empty Storage)

Test	Section 3	Section 2	Section 1	Result
1	Large	Medium	Small	Success
2	Large	Medium	Small	Success
3	Large	Medium	-	Failed
4	Large	Medium	Small	Success
5	Large	Medium	-	Failed
6	Large	Medium	Small	Success
7	Large	Medium	Small	Success
8	Large	Medium	Small	Success
9	Large	Medium	Small	Success
10	Large	Medium	Small	Success

Table 8. Green Object Sorting Test Result (Empty Storage)

Test	Section 3	Section 2	Section 1	Result
1	Large	Medium	Small	Success
2	Large	Medium	Small	Success
3	Large	Medium	Small	Success
4	Large	Medium	Small	Success
5	Large	Medium	Small	Success
6	Large	Medium	Small	Success
7	Large	Medium	Small	Success
8	Large	Medium	Small	Success
9	Large	Medium	Small	Success
10	Large	Medium	Small	Success

3.3.2 Object Sorting Test while the Storage is Full

Full storage in this context is defined as a condition in which all three of the storage sections are occupied by items. As opposed to the empty storage used in the previous test, this test was conducted with section 3, 2, and 1 of the storage zone filled with a large object, a medium object, and a small object respectively. A new, extra-large object would then be introduced to the detection zone. After comparing the size of the new object to the size of the objects present on the storage sections, the robot would conclude that the extra-large object belonged in section 3. It would then proceed to arrange the objects in storage in such a way such that allowed the extra-large object to be placed on section 3. This test was done 5 times for each color, and the results can be found in Table 9, Table 10, and Table 11.

Table 9. Blue Object Sorting Test Result (Full Storage)

Test	Section 3	Section 2	Section 1	Result
1	Extra Large	Large	Medium	Success
2	Extra Large	Large	Medium	Success
3	Extra Large	Large	Medium	Success
4	Extra Large	Large	Medium	Success
5	Extra Large	Large	Medium	Success

Table 10. Red Object Sorting Test Result (Full Storage)

Test	Section 3	Section 2	Section 1	Result
1	Extra Large	Large	Medium	Success
2	Extra Large	Large	Medium	Success
3	Extra Large	Large	Medium	Success
4	-	Large	Medium	Failed
5	Extra Large	Large	Medium	Success

Table 11. Green Object Sorting Test Result (Full Storage)

Test	Section 3	Section 2	Section 1	Result
1	Extra Large	Large	Medium	Success
2	Extra Large	Large	Medium	Success
3	Extra Large	Large	Medium	Success
4	Extra Large	Large	Medium	Success
5	Extra Large	Large	Medium	Success

3.4 Analysis

In this section, a grand total of 90 tests were performed on the robot to gauge its capabilities. In detection and relocation tests, of which there were 45, the robot arm managed to complete all of them successfully, resulting in an accuracy rating of 100% for detection and relocation tasks. In Sorting tests, of which there were also 45, the robot arm was observed to have completed 41 of them, resulting in an accuracy rating of 91.5%. These tests were all performed in ample lighting condition, as lighting was observed to have a significant impact on the robot arm's ability to correctly detect and identify objects. Replicating the tests in dim light with the exact same hardware and setup may result in dramatically lower accuracy compared to the results presented in this paper due to the poor low-light performance of the camera used in this project.

Another factor that plays a role in accuracy is the placement of the detection zone. If the detection zone is set not set up properly, the robot will have trouble correctly picking up objects and placing them in the correct section of the storage area. Therefore, it is paramount that the placement of the detection zone and robot arm be as precise as possible. If, for some reason, adjustments to the placement of the robot arm or the detection zone is not possible, tweaking the X and Y translational values is an option to increase accuracy, though a proper alignment of robot arm and detection zone is still the preferred method of increasing accuracy.

4. CONCLUSION

From the 90 tests that were performed, it can be concluded that the 5 DOF robot arm created in this research was able to properly pick up and move objects according to the inverse kinematics equations as designed. The Python program was also observed to be able to correctly detect the color, size, and position of an object within the provided detection zone and communicate with the ESP32 microcontroller via ESP32 web server. The robot arm was observed to achieve an accuracy rating of 100% after successfully completing 45 out of 45 detection and relocation tasks while an accuracy rating of 91.5% was observed after the arm managed to complete 41 out of 45 sorting tasks.

To further develop this project, better quality servo motors can be utilized to achieve an increase in accuracy and stability of the robot arm. Aside from using a detection zone, another method can be utilized to convey the location of an object to the robot arm, such as using a reference point for the program to determine the position of an object relative to said reference point. An upgrade in camera quality could also improve the robot's performance in low-light scenarios.

ACKNOWLEDGEMENTS

The author extends his thanks to the academic community of Tanjungpura University's Faculty of Engineering for the help and guidance they've provided to the author. May this paper be useful, both for the author and the readers.

REFERENCE

[1] Oxford University Press, "Oxford Dictionary," Oxford University Press, 2013.
[2] H. Jaya, Desain dan Implementasi Sistem Robotika Berbasis Mikrokontroler. 2016.
[3] M. Sonka, V. Hlavac, and R. Boyle, Image Processing, Analysis and Machine Vision. 1993. doi: 10.1007/978-1-4899-3216-7.
[4] R. Arismarjito, "Robot Lengan Otomatis Sebagai Pemisah Barang Berdasarkan Color Dengan Menggunakan ATmega 8535," 2011.
[5] R. H. Prabanegara, M. F. Noor, and E. Kunia, "Rancang Bangun Robot Lengan Pemindah dan Penyeleksi Barang Berdasarkan Color Berbasis Arduino UNO," Energy-Jurnal Ilmiah Ilmu-Ilmu Teknik, vol. 5, no. 2, pp. 31–40, 2015.
[6] F. Rokhman Iskandar, I. Sucahyo, M. Yantidewi, P. Studi Fisika, and U. Negeri Surabaya, "Penerapan Metode Invers kinematik Pada Kontrol Gerak Robot Lengan Tiga Derajat Bebas," Jurnal Inovasi Fisika Indonesia (IFI, pp. 64–71, 2020.
[7] W. Tri Setiawan, E. M. Derdian, P. Studi Teknik Elektro, and J. Teknik Elektro, "Rancang Bangun Robot Lengan 5 DOF Pemindah Barang Menggunakan Sensor Kamera Berbasis Arduino DUE."
[8] R. Zefani, A. Zurendra, R. Maulana, and H. Fitriyah, "Implementasi Inverse Kinematics Pada Robot Lengan Untuk Pengambilan Benda Dengan Coordinates Awal Acak," 2020. [Online]. Available: http://j-ptiik.ub.ac.id
[9] N. Kolban, Kolban's Book on ESP32. 2017.
[10] D. Kuhlman, A Python Book: Beginning Python, Advanced Python, and Python Exercises. 2013.
[11] D. Chakraborty, "OpenCV Contour Approximation (cv2.approxPolyDP)," PyImageSearch, 2021.
[12] J. Canny, "A Computational Approach to Edge Detection," 1986.
[13] G. Bradski, "The OpenCV Library," Dr. Dobb's Journal of Software Tools, 2000.
[14] K. M. Lynch and F. C. Park, Modern Robotics. Cambridge University Press, 2017.
[15] H. Asada, Introduction to Robotics. 2005.