Journal of Robotics and Control (JRC)
Volume 6, Issue 1, 2025
ISSN: 2715-5072, DOI: 10.18196/jrc.v6i1.24349

Enhanced Precision Control of a 4-DOF Robotic Arm Using Numerical Code Recognition for Automated Object Handling

Hanifudin Sukri [1*], Achmad Fiqhi Ibadillah [2], Rajermani Thinakaran [3], Faikul Umam [4], Ach. Dafid [5], Adi Kurniawan [6], Md. Monzur Morshed [7], Denni Kurniawan [8]
[1, 2, 4, 5, 6] Engineering Faculty, Universitas Trunojoyo Madura, Bangkalan, Indonesia
[3] Faculty of Data Science and Information Technology, INTI International University, Nilai 71800, Malaysia
[7] Department of Accounting and Information Systems, University of Dhaka, Dhaka-1000, Bangladesh
[8] Faculty of Engineering, Universiti Teknologi Burnei, Jalan Tungku Link Gadong BE1410, Brunei Darussalam
Email: [1] hanifudinsukri@trunojoyo.ac.id, [2] fiqhi.ibadillah@trunojoyo.ac.id, [3] rajermani.thina@newinti.edu.my, [4] faikul@trunojoyo.ac.id, [5] ach.dafid@trunojoyo.ac.id, [6] adi.kurniawan@trunojoyo.ac.id
*Corresponding Author

***Abstract*—This research develops a 4-DOF robotic arm system that utilizes numerical codes for accurate, automated object handling, supporting advancements in sustainable industrial automation aligned with the UN Sustainable Development Goals (SDGs), particularly Industry, Innovation, and Infrastructure (SDG 9). Key contributions include the integration of EasyOCR for reliable code recognition and a control mechanism that enables precise positioning. The robotic system combines a webcam for visual sensing, servo motors for movement, and a gripper for object manipulation. EasyOCR effectively recognizes numerical codes on randomly positioned objects against a uniform background while the microcontroller calculates servo angles to guide the arm accurately to target positions. Testing results show a success rate exceeding 94% for detecting codes 1 to 4, with minor servo angle errors requiring adjustments in arm extension by 30 mm to 50 mm. Positional error analysis reveals an average error of less than 1.5 degrees. Although environmental factors like lighting can influence code visibility, this approach outperforms traditional methods in adaptability and precision. Future research will focus on enhancing code recognition under variable lighting and expanding the system's adaptability for diverse object types, broadening its applications in industries demanding high efficiency.**

Keywords—Robotic Arm; 4-DOF; Manufacturing Innovation; EasyOCR; Numerical Code; Precision Control.

I. INTRODUCTION

As modern industries increasingly adopt automation, robotic systems are becoming essential tools for enhancing efficiency, precision, and safety in various operational tasks. Unlike human labor, which is susceptible to fatigue and error, robots are specifically designed to perform repetitive and precise tasks without compromising accuracy, thus significantly improving productivity in industrial settings. One of the most versatile robotic systems in such applications is the robotic arm, which replicates human arm movements and is particularly useful in assembly lines, material handling, and even hazardous environments [1], [2]. Robotic arms are capable of performing movements in multiple directions (up/down, right/left, forward/backward), allowing them to transfer objects with minimal human intervention [3]. While the robotic arm in this study remains a prototype, its potential for industrial applications is substantial.

To mimic the complexity of human arms, robotic arms are equipped with manipulators consisting of multiple segments and joints, typically categorized into the arm, wrist, and gripper [4]-[7]. These components enable robotic arms to handle delicate tasks with precision. Furthermore, the integration of computer vision enhances their functionality by allowing the perception and interpretation of visual data. Computer vision converts visual inputs (images or videos) into valuable information, allowing robots to recognize objects, track movements, and adapt to environmental changes [8]. In robotics, computer vision is widely applied in fields like automotive manufacturing, medical technology, and automated inspection [9]-[12]. By integrating computer vision algorithms, robotic systems can now process real-time data to control their movements effectively, which is critical for applications involving object recognition and manipulation [13]-[17].

Although significant progress has been made in robotic arm control, challenges remain, particularly in environments where objects are randomly positioned and where object recognition must be automated based on specific markers or codes. Previous research has explored various control strategies, such as Robotic Operating System (ROS) integration, depth sensors, fuzzy logic, and Proportional-Integral-Derivative (PID) controllers, each demonstrating advancements in control precision [18]-[24]. However, these approaches are often limited to structured settings and lack the flexibility needed for handling objects in unpredictable orientations or with specific identifying codes. Moreover, while some studies have focused on pattern or color-based recognition, there is a gap in research on implementing numerical code-based object manipulation in robotic arms, especially when dealing with randomly placed items.

This study addresses this gap by developing a 4-DOF robotic arm that uses numerical code recognition to identify and sort objects. The system integrates a webcam as a visual sensor and employs EasyOCR—a deep learning-based

Optical Character Recognition (OCR) tool—to detect and interpret numerical codes printed on objects. Unlike conventional color or shape-based recognition methods, numerical code-based recognition offers a straightforward and efficient means of classifying objects with high precision. However, this approach also presents unique challenges, such as ensuring reliable detection under various lighting conditions, background contrast, and object orientation. By overcoming these obstacles, the system demonstrates a high degree of flexibility and adaptability for industrial applications.

The robotic arm developed in this study operates by recognizing numerical codes on objects placed randomly on a white surface within its workspace. The OCR system processes images from the webcam to interpret the numerical code, which then directs the robot to transport the object to a designated location. This setup allows the robot to adapt to varying object positions and orientations, a capability not commonly seen in current robotic systems. The use of EasyOCR enables accurate code detection even when objects are partially rotated or placed in less ideal conditions, addressing a critical gap in automated object manipulation.

Introducing a precision control system for a 4-DOF robotic arm that leverages numerical code-based recognition, allowing for accurate object manipulation in unstructured environments. This feature enables the robot to manage tasks that require adaptable object handling.

Pioneering the application of EasyOCR in robotic systems to automate object sorting based on numerical codes. Unlike previous research that focuses on color-based detection or simpler control algorithms, this study utilizes OCR for enhanced adaptability, paving the way for broader industrial applications.

To address safety and ethical considerations, this robotic system is designed to perform its tasks in environments isolated from human operators, minimizing potential risks. This aspect is crucial, as robots increasingly coexist with human workers in various industries. Furthermore, this research provides a foundation for future developments in robotic automation, particularly in sectors where high-precision sorting and handling are required. By emphasizing OCR-based manipulation, the study contributes to an evolving field of industrial automation, with potential applications in logistics, automated warehousing, and beyond.

In conclusion, the system developed in this study demonstrates significant advancements in using computer vision and OCR for robotic manipulation. With enhanced adaptability to dynamic environments and an innovative approach to handling numerical codes, this 4-DOF robotic arm prototype has promising implications for improving both the efficiency and accuracy of material handling in industrial settings.

II. LITERATURE REVIEW

A. *Robot ARM*

A robot arm is a mechanical device designed to mimic tasks typically performed by the human hand. This robot is known as a manipulator or robot arm, consisting of links and joints. Each joint on the robot arm can move according to the commands given, but the movement of the robot arm is not the same as the movement of a human hand, which has the ability to perform complex motions[25]-[27]. However, the robot's joints possess a range of movement known as Degrees of Freedom (DOF). A robotic manipulator has more freedom of movement compared to a human arm. For example, the robot's articulated elbow can move up and down, whereas the human elbow can only bend in one direction when the arm is straight.

Fig. 1 shows a robot arm capable of performing movements that allow the robot to precisely reach specific positions. The manipulator of the robot usually consists of the arm and the wrist.

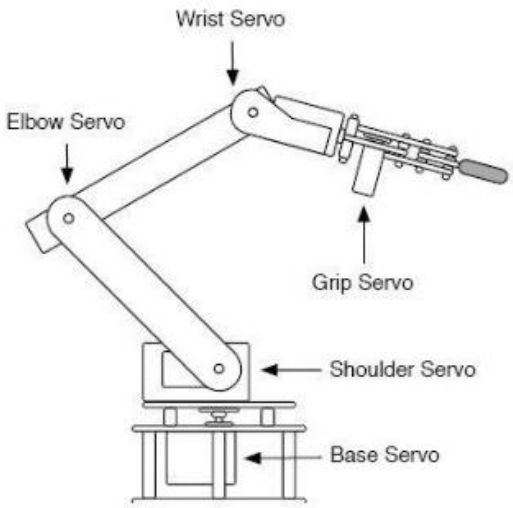

Fig. 1. Robot ARM

B. *Degree of Freedom*

Degrees of Freedom (DOF) refers to the number of independent movements that a system can perform. In robotics or mechanics, DOF refers to the number of joints or axes of motion a mechanism or robot has [28], [29]. DOF is described as the degree of freedom a system has to move in three-dimensional space, where each joint or axis of motion on the robot adds one degree of freedom [30]-[33]. The number of DOF required in the design of a robot arm or other mechanical system depends on the application needs and the complexity of the tasks to be performed. The higher the DOF, the more complex and flexible the movements that can be executed to complete specific tasks.

C. *Kinematics*

Kinematics is the study of the motion of bodies without considering the forces, torques, or moments that cause the motion. The kinematics discussed here focuses explicitly on studying and analyzing the movement of robot arms [6], [34], [35]. Robot kinematics consists of two types (Fig. 2). Forward kinematics calculates the orientation and position of the end-effector based on the joint angles. Inverse kinematics, on the other hand, is the reverse of forward kinematics; it provides the end-effector's position and the required angles that must be adjusted for each joint to reach that position.

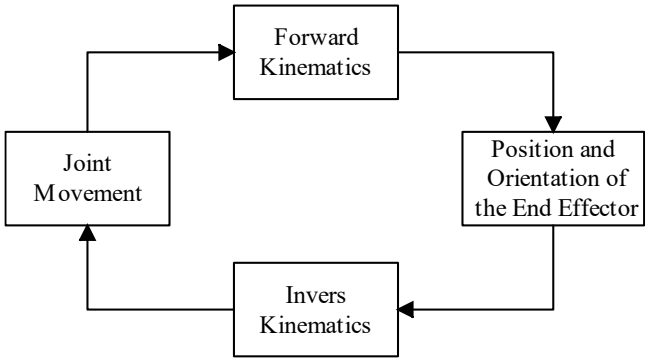

Fig. 2. Diagram block kinematics

D. *Forward Kinematics*

Forward kinematics is the kinematic analysis used to obtain the position coordinates (x, y, z) when the angles of each joint are known (Fig. 3). For instance, if a robot with multiple DOF is given, and the angles of each joint are known, forward kinematic analysis can be used to determine the robot's position coordinates. Forward kinematics is used to determine the position and orientation of the end-effector when the joint angle variables are known [36]-[42]. The given joint angle variables are converted into the position and orientation of the end-effector, which are referenced to a coordinate system. The purpose of the forward kinematics method is to obtain the x, y, and z coordinate values.

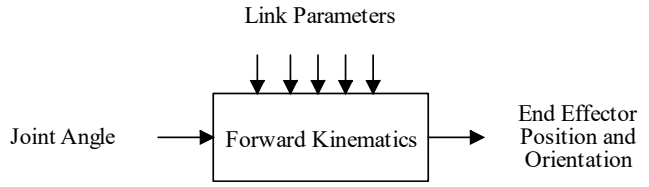

Fig. 3. Forward kinematics

E. *Invers Kinematic*

Inverse kinematics, also known as reverse kinematics, is used to find the joint angle variables of a robot in order to determine the position and orientation of the end-effector (Fig. 4) [36], [37], [41]. To achieve this, the robot's kinematic equations are used to determine the joint parameters that will provide the desired position of the end-effector.

Kinematic equations are used to simulate the movement of a robot. The robot's configuration is determined based on parameters for each actuator according to these equations. The parameters are calculated using forward kinematics, and this calculation is reversed to determine the joint parameters needed to achieve the desired configuration. The inverse kinematics method generally finds the parameter values required for each actuator to reach the end goal. To determine these parameter values, the robot must know the size, number, and degrees of freedom of the actuators. Furthermore, the formulas collected from various calculation models must be embedded in the robot. This must be done using direct graphical analysis and different research methods.

The analysis of kinematic equations can be solved in the most basic way, which is by using trigonometry with the help of graphs. Each component in the coordinates (x, y, z) is expressed as a transformation of each component in its own space (r, θ). The radius r in the equation is often referred to as the length of the arm or the first link.

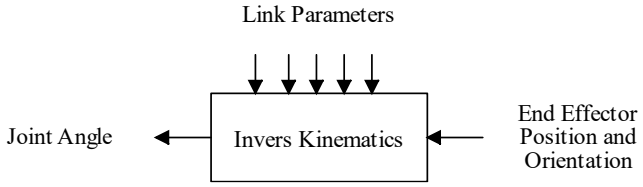

Fig. 4. Invers kinematics

Inverse kinematic equations can be determined by applying trigonometry, observing each joint that moves in a single direction. Below are the x, y, and z coordinates as shown in Fig. 5.

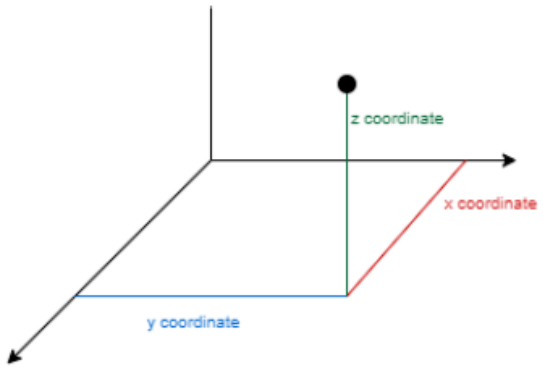

Fig. 5. Coordinate x, y, and z

To find the solution for the inverse kinematics method, the coordinates need to be simplified by converting the 3D form into 2D. The first step in solving inverse kinematics involves viewing the robot arm from above, or along the y-axis, thus displaying the (x, y) plane. The target coordinates of the end-effector along the x-axis are denoted as x, and the coordinates along the y-axis are denoted as y. The required angle for the motor to rotate, as well as the arm's extension length for the end-effector to reach the target position, is symbolized by $(\theta 1)$ for the rotation angle (Fig. 6).

In determining the coordinates of the end-effector, inverse kinematics must be adjusted to the workspace limits of the robot's reach. Inverse kinematics involves calculations opposite to forward kinematics. Forward kinematics is used to obtain the position coordinates (x, y, z) when the angles of each joint are known, whereas inverse kinematics is the process of finding the joint angle variables to determine the position and orientation of the end-effector.

The solution for inverse kinematics can be resolved using inverse kinematics, which involves the Pythagorean theorem and the law of cosines. This inverse kinematics solution must be approached from two perspectives: the top view and the side view of the robot arm structure. The first step in solving inverse kinematics is viewing the robot arm from above, or along the y-axis, displaying the (x, y) plane. The target coordinates of the end-effector on the x-axis are referred to as x, and on the y-axis as y. The angle that the motor must rotate and the arm extension length needed for the end-effector to reach the target position are symbolized by $(\theta 1)$ for the rotation angle.

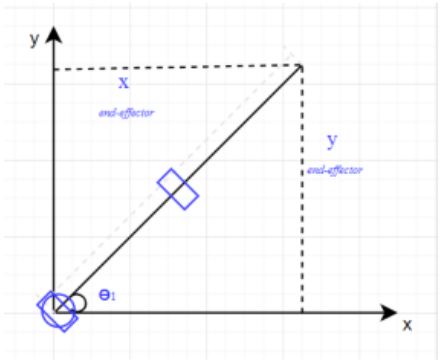

Fig. 6. Top view

To calculate the angle (θ) that the servo motor must rotate to reach the target position coordinates of the end-effector, the following formula is used:

$$\theta_1 = tan^{-1}\left(\frac{y}{x}\right) \quad (1)$$

Explanation, $\theta 1$ is the Angle, y is the coordinate on the y-axis, x is the coordinate on the x-axis.

The next step in solving inverse kinematics is calculating the rotation angles of the remaining three joints ($\theta 2$, $\theta 3$, and $\theta 4$) by viewing the robot's structure from the side, as shown in Fig. 7 below.

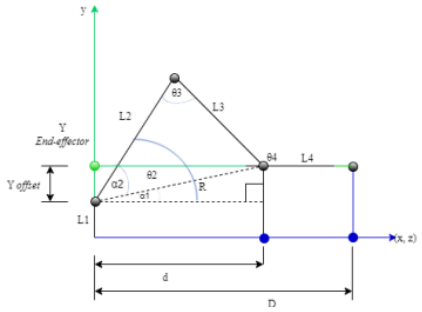

Fig. 7. Side view

The robot arm's structure from the side view, which displays the y-axis and the horizontal axis, represents the surface plane or the (x, z) plane. The vertical target coordinate of the end-effector, i.e., the coordinate on the y-axis, is called y, and the target coordinate on the horizontal axis (coordinate plane (x, z)) is the length of the extension (D). Therefore, the inverse kinematics calculations using geometry to determine the rotation angles of the joints $\theta 2$, $\theta 3$, and $\theta 4$ that the motor must rotate can be solved using the following formulas:

$$d = D - L_4 \quad (2)$$

Explanation, d is the extension length from θ2 to θ4, D is the extension length, $L4$ is the arm length 4.

Fig. 7 shown earlier forms a right-angled triangle with sides (d, $YOffset$, and R), where Y is the difference in distance between the y-axis coordinate of the end-effector ($YOffset$) and the position of the second motion axis ($L1$).

$$Y_{offset} = Y_4 - L_1 \quad (3)$$

Y_{offset} is the difference in distance (y-axis coordinate) with the end-effector point.

Y_4 is the distance between the first motion axis and the position of the fourth motion axis is 18.

L_1 is the arm length 1. Using the Pythagorean theorem and the cosine rule, R and α can be calculated using the following equations:

$$R = \sqrt[2]{(2)^2 + (Y_{offset})^2} \quad (4)$$

$$\frac{d}{R} = cos\ (a_1) \quad (5)$$

By using the inverse function to find $\alpha 1$, the equation becomes as follows:

$$a_1 = cos^{-1}\left(\frac{d}{R}\right) \quad (6)$$

Fig. 7 shown earlier forms a triangle with sides ($L2$, $L3$, and R). Using the cosine rule, $\alpha 2$ can be calculated with the following equation:

$$(L_3)^2 = (L_2)^2 + (R)^2 - 2xL_2xRxcos(a_2) \quad (7)$$

By using the inverse function to calculate $\alpha 2$, the equation becomes as follows:

$$a_2 = cos^{-1}(\frac{L_2^2 + R^2 + L_3^2}{2xL_2xR}) \quad (8)$$

Then, to solve for $\theta 2$, it is the sum of $\alpha 1$ and $\alpha 2$, as follows.

$$\theta_2 = a_1 - a_2 \quad (9)$$

To calculate $\theta 3$, the cosine rule can also be used based on the triangular plane formed in Fig. 7, with sides ($L2$, $L3$, and R). The equation is as follows:

$$R^2 = (L_2)^2 + (L_3)^2 - 2xL_2xL_3xcos(\theta_3) \quad (10)$$

By using the inverse function to calculate $\theta 3$, the equation becomes as follows:

$$\theta_3 = cos^{-1}(\frac{L_2^2 + R^2 + L_3^2}{2xL_2xL_3}) \quad (11)$$

The last joint, $\theta 4$, can be calculated based on the angle that must be formed by the servo motor rotation between $L3$ and $L4$. It is important to ensure that the gripper's orientation is parallel to the surface (coordinate plane (x, z)) for optimal object gripping. Using the rule that the sum of the angles in a triangle does not exceed 180°, the equation to calculate $\theta 4$ is as follows:

$$\theta_4 = 100^{\circ} - \{[100° - (a_2 + \theta_3)] - a_1\} \quad (12)$$

In some cases, if the height of the end-effector (Y4 position) is lower than the position of joint 2, the calculations for $\theta 2$ and $\theta 4$ require different equations or calculation methods. For the Y4 position lower than the position of joint 2, the robot's configuration is shown in Fig. 8 below.

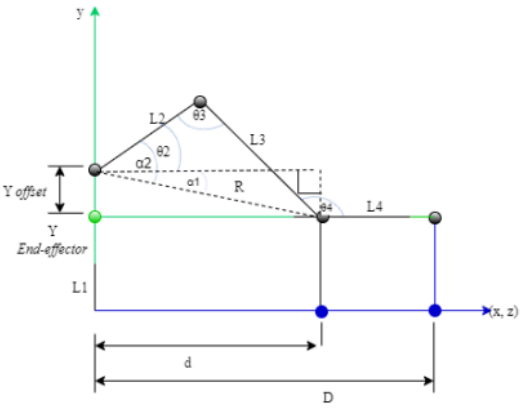

Fig. 8. Side view when the end-effector's y-axis position is below joint 2

Fig. 8 shows the movement scenario when the y-axis position of the end-effector is below the position of joint 2, creating a distance difference on the y-axis between the end-effector and joint 2. This distance difference is symbolized as $YOffset$. Therefore, the calculation for $\theta 2$ is as follows:

$$\theta_2 = a_2 - a_1 \quad (13)$$

To calculate α_1, the same principles of the Pythagorean theorem and the cosine rule for right-angled triangles are used, as in equation (6) previously. The calculation of α_2 employs the same cosine principle as in equation (8) previously. Then, the calculation for $\theta 4$ is as follows:

$$\theta_4 = 100^{\circ} - \{[100° - (a_2 + \theta_3)] - a_1\} \quad (14)$$

It is important to note that the zero reference point of the environmental coordinate system is set based on the position of joint 1 of the robot. Since the rotational axis of joint 1 is along the y-axis, joint 1 serves as the reference point for both the x-axis and z-axis in the environment. The zero point of the y-axis is the working surface, measured up to the center of the rotational axis of joint 2, where the rotational axis of joint 2 is perpendicular to joint 1.

F. *Motion Path Control System*

Forward and Inverse Kinematics are solutions to the kinematic control of a robotic manipulator, which only focus on the geometry of the robot [36], [37], [41], [43], [44]. These solutions do not take into account other constraints imposed by the workspace in which the robot operates. Specifically, forward and inverse kinematics do not consider the possibility of collisions between the robotic manipulator and objects within the workspace. Therefore, a robot motion path planning system is required so that the robot can move along path points, including the starting point, intermediate points, and the target point. There are various methods for performing path planning, all of which aim to provide a series of points, called via points, along the path. The simplest method of path planning is to provide a sequence of end-effector positions. In this method, inverse kinematics is required to convert the end-effector positions into joint configurations. For the robot to move precisely to each path point resulting from the path planning, motion planning is required. This motion planning is done by creating a position function based on time, commonly known as a trajectory. Since the trajectory is a function of time, the velocity and acceleration of the robot along the path can also be calculated. Trajectory planning can be carried out either in joint space or Cartesian space.

G. *Image Processing Procedure*

An image can be defined as a function $f(x, y)$ of size M rows and N columns, where x and y are spatial coordinates, and the amplitude of f at the coordinate point (x, y) is called the intensity or grayscale level of the image at that point. A digital image consists of a number of elements, each element having a specific location and value [45], [46], [47]. These elements are referred to as picture elements, image elements, pels, or pixels. Sources of noise in digital images can occur during image acquisition or transmission. The performance of image sensors or cameras is influenced by many factors, such as environmental conditions during image capture with a webcam, lighting levels, and sensor temperature, which are the main factors affecting the level of noise in the resulting image. If the values of x, y, and the amplitude f are finite and discrete, the image can be considered a digital image (Fig. 9).

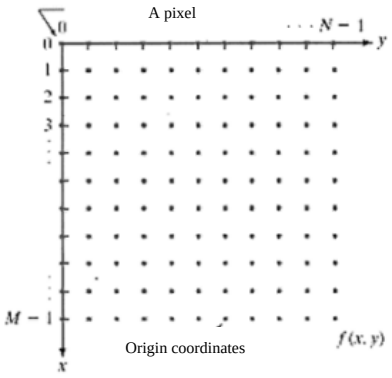

Fig. 9. Example of digital image coordinates

H. *Grayscale Image*

A grayscale image is a type of digital image where each pixel is represented by a single intensity value (Fig. 10). Grayscale images do not contain color information, only representing the brightness or intensity level of each pixel. One of the initial processes commonly performed in image processing is converting a color image to a grayscale image, as this simplifies the image model. For a digital image to be processed by a computer, it must have a specific format. The digital image format used is grayscale, called grayscale because it typically uses black as the minimum color (0) and white as the maximum color (255), with the colors in between being shades of gray.

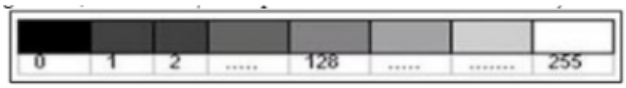

Fig. 10. Grayscale Image

I. *Otsu Thresholding*

Otsu thresholding is a method used in image processing for automatic segmentation based on histogram analysis. The approach applied by the Otsu method involves discriminant analysis by determining a variable that can distinguish between two or more naturally formed groups [48], [49], [50], [51], [52], [53], [54], [55]. The discriminant analysis maximizes the separation of the object (foreground) and the

background. To obtain the threshold value, a calculation must be performed. The first step is to create a histogram to determine the number of pixels at each grayscale level. The grayscale levels of the image are denoted by i to L. The probability of each pixel at the i-th level is expressed in Equation (15).

$$P_i = \frac{n_i}{N} \quad (15)$$

Explanation, P_i is the probability of the i-th pixel, n_i is the number of pixels with grayscale level i is 24, N is the total number of pixels in the image.

The next step is to find the values of the cumulative, cumulative mean, and global intensity. These values can be determined using the formulas for calculating the cumulative sum of $\omega(k)$ for L = 0, 1, 2, …, L-1 in equation (16), the formula for calculating the cumulative mean of $\mu(k)$ for L = 0, 1, 2, …, L-1 in equation (17), and the formula for calculating the global intensity mean of $\mu_T(k)$ for L = 0, 1, 2, …, L-1 in equation (18).

$$\omega(k) = \sum_{i=0}^{k} p_i \quad (16)$$

$$\mu(k) = \sum_{i=0}^{k} i.p_i \quad (17)$$

$$\mu_T(k) = \sum_{i=0}^{L-1} i.p_i \quad (18)$$

In the equation above, the value of k represents the grayscale level at which each pixel range will be calculated. To determine the between-class variance, the next step can be seen in equation (19).

$$\sigma_B^2(k) = \frac{[\mu_T\omega(k)-\mu(k)]^2}{\omega(k)[1-\omega(k)]} \quad (19)$$

From the results of the between-class variance calculation, the maximum value is determined. The largest value is used as the threshold (k), as shown in equation (20).

$$\sigma_B^2(k) = max_{1\le x\le L}\sigma_B^2(k) \quad (20)$$

Explanation, $\omega(k)$ is the cumulative total, $\mu(k)$ is the cumulative mean, $\mu_T(k)$ global intensity mean, $\sigma_B^2(k)$ is the threshold value.

The purpose of the between-class variance is to find the threshold value from a grayscale image, where the threshold value will be used as a reference to convert the grayscale image to a binary image. Each image does not have the same threshold value.

J. *Binary Image*

Binarization converts the grayscale image colors into black and white, or binary (Fig. 11). If pixels exist, the colors will change from values of 0 and 255 in the image to pixel values of 0 and 1 for each pixel [56], [57], [58], [59], [60]. As a result, the image becomes black and white. The formula used to convert a grayscale image to a black-and-white or binary image is represented by equation (21).

$$g(x,y) = \begin{cases} 1, jika\ f(x,y) \ge T \\ 0, jika\ f(x,y) < T \end{cases} \quad (21)$$

Explanation, $f(x,y)$ is the grayscale image, $g(x,y)$ is the binary image, T is the threshold value.

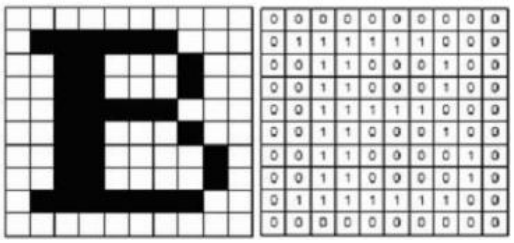

Fig. 11. Binary image and binary image representation

K. *Morphological Operations*

Morphological operations are commonly applied to binary (black-and-white) images to alter the structural shape of objects contained within the image [61], [62], [63], [64], [65], [66]. Morphology is an image processing technique based on the shape of image segments. The goal of morphological operations is to improve the results of segmentation. Examples of morphological operations/applications include:

a. Closing holes in the image.
b. Separating objects.
c. Forming a spatial filter.
d. Obtaining the skeleton of the object
e. Determining the position of the object in the image
f. Obtaining the structural shape of the object

L. *Erosion*

Erosion is a morphological operation that reduces the pixels at the boundaries between objects in a digital image [67]. When erosion is performed, pixels at the boundaries of the object being eroded are removed. The number of pixels added or removed depends on the size and shape of the structuring element used to process the image.

Fig. 12 Below is the result of the image after the erosion process, applied to the original image with the structuring element specified above. The green pixels represent those removed after the erosion process, leaving only the black pixels as the result.

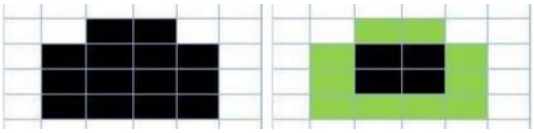

Fig. 12. Morphological erosion operation

M. *Labeling Object*

Labeling serves to group specific areas (pixels) and is commonly used in character recognition applications and object number detection. Binary images are frequently used in image processing for further analysis. In a binary image, multiple objects are displayed through the values that appear. The objects that appear can be counted using labeling. There are neighborhood rules that determine whether a pixel belongs to a specific neighboring region (Fig. 13).

Fig. 13. Types of neighborhoods

N. *Image Segmentation*

In image processing, sometimes we want to process only specific objects. Therefore, image segmentation is necessary to separate the foreground object from the background. Typically, the output of image segmentation is a binary image, where the desired object (foreground) is white (1), while the background to be removed is black (0) [68], [69], [70], [71], [72], [73], [74].

O. *Centroid*

The centroid is the center point or the midpoint of an object's area with coordinates (x, y). The centroid value for each object will vary. The centroid is used to determine the position of an object in an image. In a binary image or black-and-white image, the centroid is also known as the center of gravity, meaning the center of mass based on pixel intensity in the image (Fig. 14) [75], [76], [77], [78].

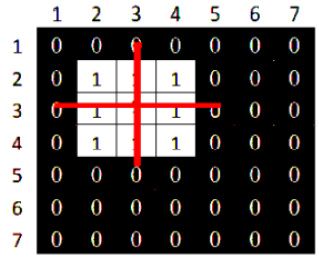

Fig. 14. Process of finding the centroid value

Based on the example in Fig. 14, the centroid value can be calculated using equations 22 and 23. Below is an example of how to calculate the centroid value.

a. Coordinate x

$$x_t = \frac{1}{9}(2 \times 1) + (2 \times 1) + (2 \times 1) + (3 \times 1) + (3 \times 1) + (3 \times 1) + (4 \times 1) + (4 \times 1) + (4 \times 1) \quad (22)$$

$$x_t = \frac{1}{9}(27)$$

$$x_t = 3$$

b. Coordinate y

$$y_t = \frac{1}{9}(2 \times 1) + (2 \times 1) + (2 \times 1) + (3 \times 1) + (3 \times 1) + (3 \times 1) + (4 \times 1) + (4 \times 1) + (4 \times 1) \quad (23)$$

$$y_t = \frac{1}{9}(27)$$

$$y_t = 3$$

Based on the calculation results, the centroid value (x, y) is (x = 3, y = 3).

P. *Optical Character Recognition (OCR)*

OCR stands for Optical Character Recognition. OCR is a technology used to recognize and extract text or characters from a scanned or photographed image or document, allowing the text to be processed and manipulated in computer software [79], [80], [81], [82], [83]. This OCR technology enables more effective and efficient document digitization and processing, as it allows physical documents to be converted into digital formats that can be processed more easily and quickly. OCR is commonly used in applications such as signature recognition, business card scanners, and OCR applications that can detect text in an image or video for automatic processing.

Q. *EasyOCR*

OCR, previously known as Optical Character Recognition, is revolutionary for today's digital world. OCR is essentially a complete process where images/documents in the digital world are processed, and the text is converted into editable, normal text. The purpose of OCR is to enable readers to convert various types of documents, such as scanned paper documents, PDF files, or images taken with a digital camera, into editable and searchable data.

EasyOCR is essentially a Python package that uses PyTorch as its backend handler [84], [85]. It detects text from images, and in the researcher's reference, when using it, the researcher found it to be the easiest way to detect text from images. Additionally, with the support of a top-tier deep learning library (PyTorch) on the backend, it makes the accuracy more credible.

EasyOCR supports over 42 languages for detection purposes (Fig. 15) [86], [87]. EasyOCR was created by a company called Jaided AI.

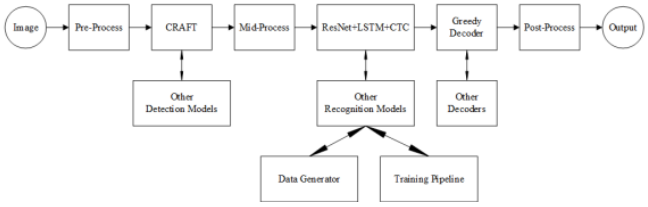

Fig. 15. EasyOCR framework

III. METHOD

A. *Block Diagram*

The system consists of four main sections: input, display, control, and output, as depicted in the block diagram (see Fig. 16).

1) Input Section

A Logitech C270 webcam captures images of objects within the workspace, which are sent to a computer for processing. The webcam captures both video and pixel depth data, essential for accurately mapping object positions.

2) Display Section

The computer interface, managed by the Processing IDE, visually displays real-time information, including object coordinates, recognized numerical codes, and calculated servo angles.

3) Control Section

The Arduino Uno microcontroller receives processed data and sends precise commands to each servo motor based on the calculated angles.

4) Output Section

The 4-DOF robotic arm utilizes servo motors to actuate its joints, with the gripper serving as the end-effector to grasp and move objects. Once object coordinates are identified, inverse kinematics determines joint angles for the base, shoulder, and elbow, which are then executed by the Arduino controller to move the robotic arm to the specified position.

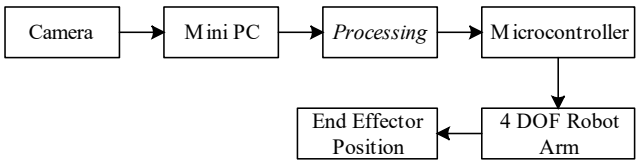

Fig. 16. Block diagram

B. *Flowchart of Object Retrieval Process*

The operational flow of the robotic arm's actions, as shown in the object retrieval flowchart Fig. 17, demonstrates how the arm identifies, captures, and moves objects based on their numerical codes and positions. Initially, the robotic arm is in a standby position, awaiting input. Once an object enters the designated workspace, the webcam captures an image of the workspace, focusing specifically on objects within the specified area. This captured image undergoes several preprocessing steps to ensure clarity and facilitate further processing. First, the image is converted to a binary format, simplifying the data by reducing it to black and white pixels, which aids in distinguishing the object from the background. Following this, noise filtering is applied to enhance image quality by removing unnecessary visual artifacts that could interfere with accurate detection.

After preprocessing, the system assigns labels to each detected object, allowing them to be uniquely identified in the subsequent segmentation phase. During segmentation, the object's shape and position are determined, which provides precise coordinates for accurate handling. With the object's numerical code and position identified, VsCode transmits this information to the Arduino microcontroller, initiating the robotic arm's movement sequence. Based on the received data, the microcontroller calculates the necessary angles and sends instructions to each servo motor, guiding the robotic arm's movement. Finally, the robotic arm's gripper is directed to the object's location, where it securely picks up the object and places it in a predetermined area according to the identified numerical code, completing the sequence.

C. *Digital Image Preprocessing Flowchart*

The image preprocessing steps, as outlined in the flowchart Fig. 18, are designed to enhance image quality and ensure optimal accuracy in object recognition. This process begins with grayscale conversion, where RGB images are transformed into grayscale, reducing data complexity and improving processing speed. By converting the image to grayscale, only the luminance values are retained, simplifying the information without losing essential visual details necessary for object detection.

Following grayscale conversion, thresholding using Otsu's method is applied. This is a binarization technique that automatically determines the optimal threshold to separate objects from the background, converting the image into clear black-and-white regions. This contrast-enhancing step is crucial, as it isolates the object by defining its boundaries against the background, facilitating more accurate recognition in the subsequent stages.

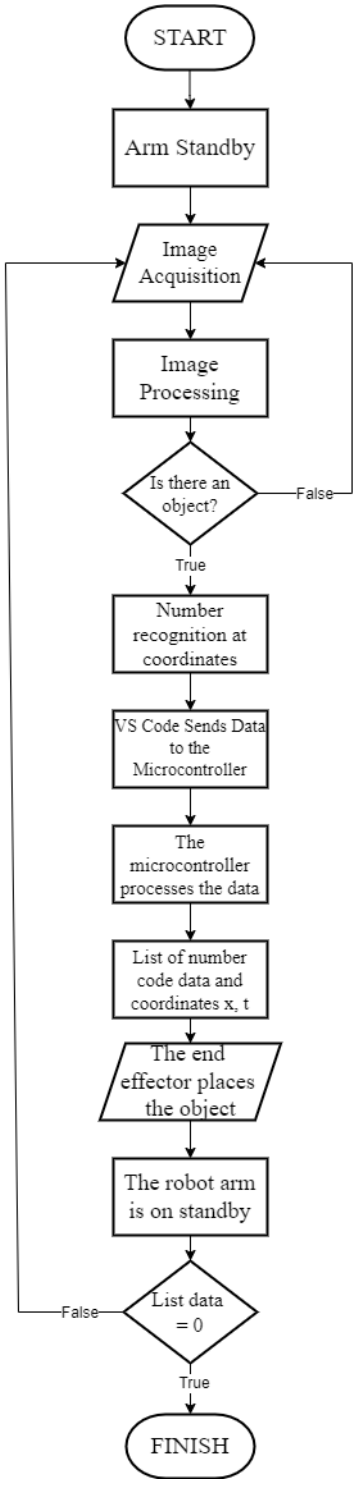

Fig. 17. Cube retrieval flowchart

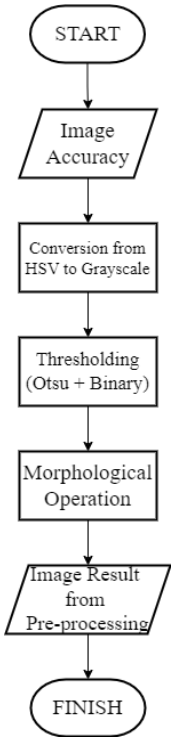

Fig. 18. Pre-processing process

The final step in the preprocessing phase is morphological erosion, a technique used to refine object boundaries by removing small noise particles. This operation reduces unnecessary visual details around the object, sharpening its edges and enhancing the accuracy of the OCR process. By eliminating minor artifacts and smoothing edges, morphological erosion ensures that the object's structure is well-defined, improving the reliability of numerical code detection.

Once these preprocessing steps are completed, the prepared image is ready for the number code detection phase, as depicted in Fig. 19. Here, the preprocessed image undergoes character recognition using EasyOCR, allowing for the precise identification of numerical codes on each object. This sequence of steps in image preprocessing and number code detection establishes a solid foundation for accurate and efficient object handling by the robotic arm.

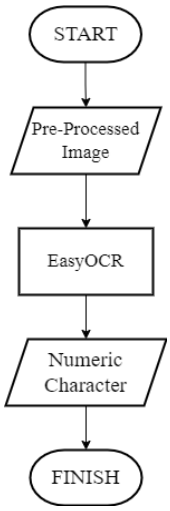

Fig. 19. Flowchart for number code detection

Fig. 20 depicts the number code detection process, where the preprocessed image undergoes further analysis to recognize numerical characters using the EasyOCR method. This stage leverages EasyOCR's character recognition capabilities to accurately identify and interpret the numerical codes on each object, essential for precise object classification and handling.

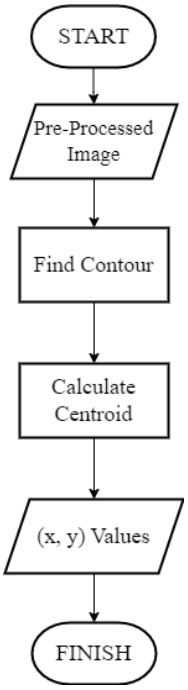

Fig. 20. Cube position determination flowchart

In addition to code detection, the system determines each object's position by locating its centroid, calculated from the preprocessed image data. The centroid, representing the center or midpoint of the object's area, is identified with coordinates (x, y). These coordinates are crucial for positioning, as they provide the precise location of the object within the workspace, allowing the robotic arm to accurately align with and manipulate the target object.

D. *Hardware Circuitry*

The webcam is the device used as the input. The input from the webcam is the image of the object and the captured characters. The object is within the specified workspace (20 cm in length, 27.5 cm in width). The resulting image is sent to the personal computer via a direct connection using a USB cable.

The GUI (Graphical User Interface) displays the angle values for each servo, the x and y coordinates of the object, and the numerical character. The obtained x and y coordinates are processed using inverse kinematics calculations to determine the angle values for each joint's servo. Once the angle values for each joint's servo are calculated, the Python program sends these values to the Arduino Uno through serial communication.

The Arduino Uno, acting as a microcontroller, moves each servo according to the obtained values. Afterward, the four-DOF robotic arm reaches the object's x and y coordinates, grips the object with the gripper, and moves it to the default location or a desired location.

In detail, the design of the tool used in this research is as shown in Fig. 21. With the design of the robot arm as shown in Fig. 22 and Fig. 23.

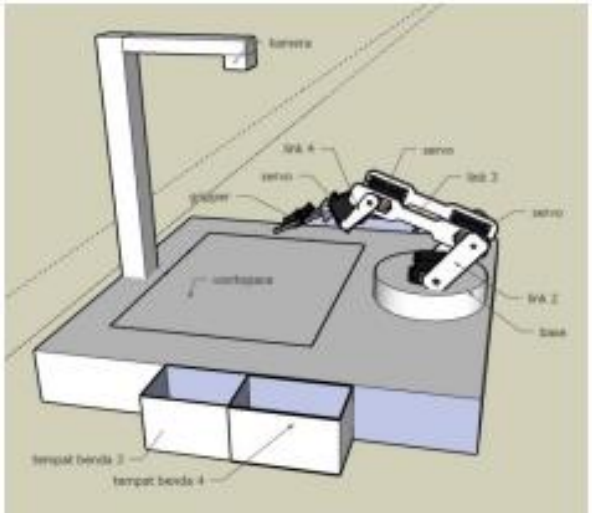

Fig. 21. Tool Design

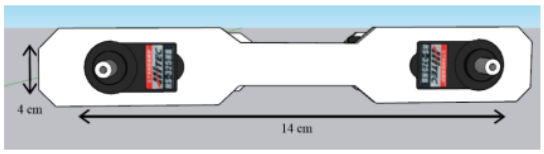

Fig. 22. Link 2 dan 3

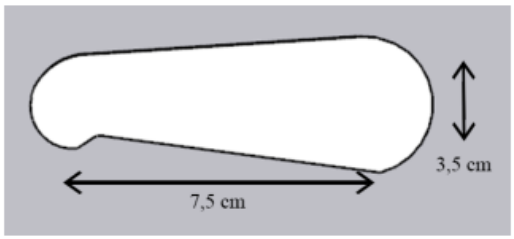

Fig. 23. Link 4

IV. RESULT AND DISCUSSION

A. *Object Placement Area Testing*

The object placement area detection testing evaluates the accuracy of the robotic ARM in locating and manipulating objects based on their coordinates within the workspace (Fig. 24). This test involves placing an object within a defined workspace area while a camera captures real-time video to identify the object's position. Once coordinates are determined, the robotic ARM responds by moving to the designated location, following specific programmed conditions that adjust its movement.

The object placement area measures 450×300 pixels, with each centimeter corresponding to 23 pixels. Notably, the webcam is positioned upside down, causing the centroid values of each position to be inverted. This setup requires software adjustments to ensure accurate location tracking and movement coordination.

To simplify the process of determining (X, Y) axis coordinates, the object placement area is divided into six columns, each with its own unique adjustments programmed in Arduino. These columns facilitate dynamic adjustments to the ARM's positioning based on the sX and sY values, which represent coordinates relative to a reference point within the workspace.

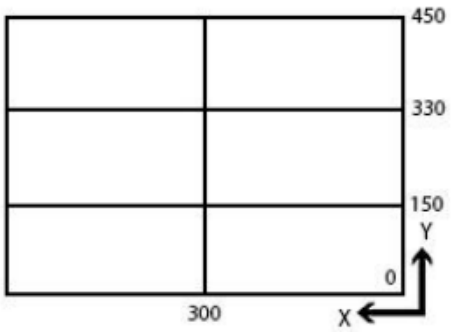

Fig. 24. Object Placement Area

In this system, the sX and sY values influence servo motor positioning by adjusting angle changes (dX) and height levels (dY1, dY2, and dY3) based on the detected object location. This configuration allows the robotic ARM to calibrate its mechanical response precisely in alignment with the object's visual position. The area division and corresponding adjustments ensure the ARM maintains accuracy and flexibility in real-time object handling. The area division is as follows:

a. Range sY 0-149:
   Condition: if (sY >= 0 && sY < 150)
   Action:
   dX Added with 45.
   dY1 Calculated using the formula: 105 - (((sumY * 10) / 1.52) / 30)
   dY2 Calculated using the formula: 180 - (((sumY * 10) / 1.52) / 40)
   dY3 Calculated using the formula: 50 + (((sumY * 10) / 1.52) / 50)

b. Range sY 150-330 dan sX <= 300:
   Condition: else if (sY > 150 && sY <= 330 && sX <= 300)
   Action:
   dX Added with 50.
   dY1 Calculated using the formula: 85 - (((sumY * 10) / 1.52) / 30)
   dY2 Calculated using the formula: 165 - (((sumY * 10) / 1.52) / 40)
   dY3 Calculated using the formula: 65 + (((sumY * 10) / 1.52) / 50)

c. Range sY 150-330 dan sX > 300:
   Condition: else if (sY > 150 && sY <= 330 && sX > 300)
   Action:
   dX Added with 40.
   dY1 Calculated using the formula: 85 - (((sumY * 10) / 1.52) / 30)
   dY2 Calculated using the formula: 165 - (((sumY * 10) / 1.52) / 40)
   dY3 Calculated using the formula: 65 + (((sumY * 10) / 1.52) / 50)

d. Range sY 331-450 dan sX <= 300:
Condition: else if (sY > 330 && sY <= 450 && sX <= 300) Action:
dX Added with 53
dY1 Calculated using the formula: 55 - (((sumY * 10) / 1.52) / 30)
dY2 Calculated using the formula: 130 - (((sumY * 10) / 1.52) / 40)
dY3 Calculated using the formula: 60 + (((sumY * 10) / 1.52) / 50)

e. Range sY 331-450 dan sX > 300:
Condition: else if (sY > 330 && sY <= 450 && sX > 300)
Action:
dX Added with 43.
dY1 Calculated using the formula: 55 - (((sumY * 10) / 1.52) / 30)
dY2 Calculated using the formula: 130 - (((sumY * 10) / 1.52) / 40)
dY3 Calculated using the formula: 60 + (((sumY * 10) / 1.52) / 50)

The calculation results for points a to e can be seen from the test results based on Fig. 25, Fig. 26, Fig. 27 and Fig. 28.

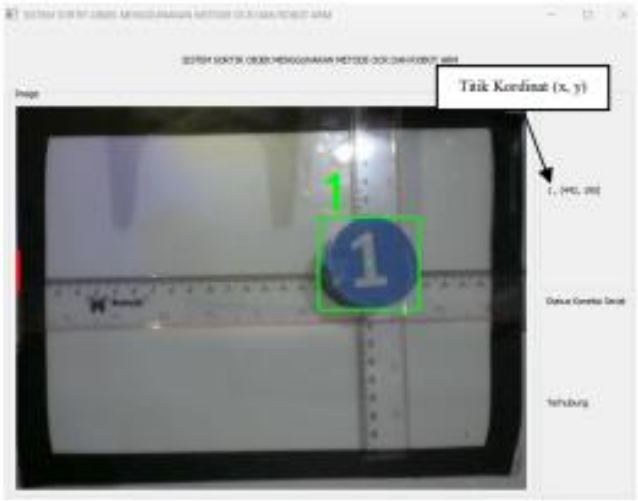

Fig. 25. Coordinate Point Testing for Number Code 1

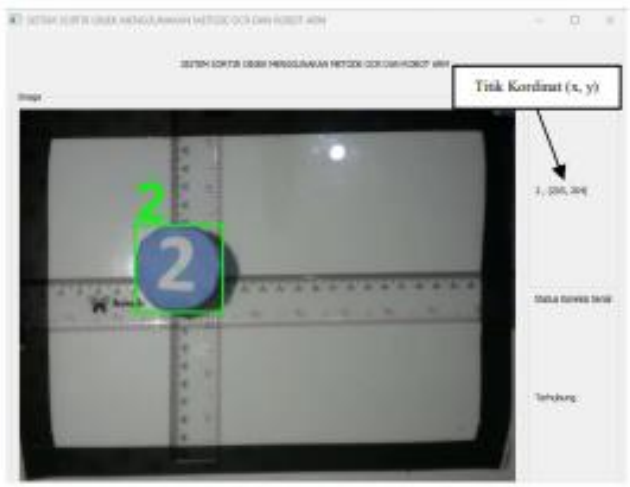

Fig. 26. Coordinate Point Testing for Number Code 2

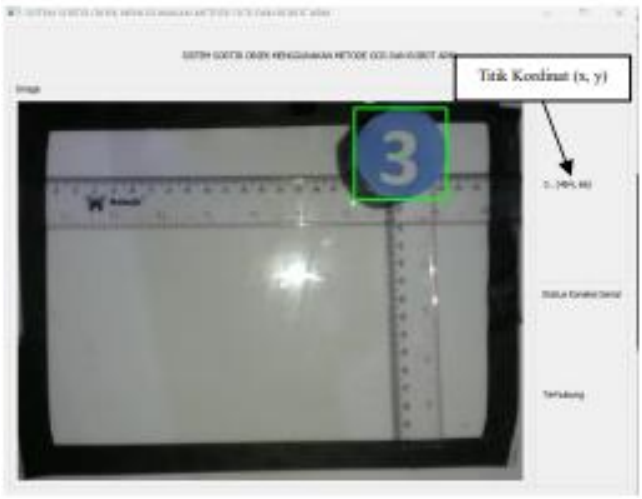

Fig. 27. Coordinate Point Testing for Number Code 3

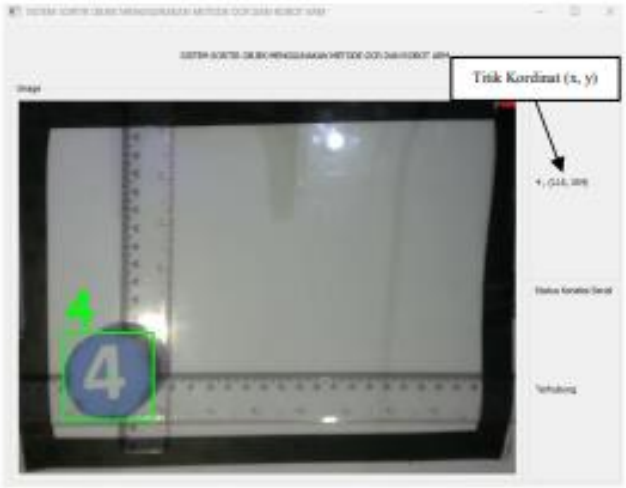

Fig. 28. Coordinate Point Testing for Number Code 4

B. *EasyOCR Data Training*

In this study, the OCR component relies on EasyOCR, which primarily uses pre-trained models. However, for specific use cases like recognizing distinct digits, it is recommended to use a custom digit recognition model with ResNet18 architecture. This section details the steps for fine-tuning the OCR model for accurate number code detection.

a) Preprocessing

Preprocessing is a foundational step that enhances the image quality to improve text detection and recognition accuracy. Images are initially converted to grayscale to reduce data complexity and focus on luminance, which is crucial for accurate OCR.

b) Mid-process with ResNet

ResNet (Residual Network) is incorporated as a feature extractor in this OCR pipeline. Known for its ability to maintain performance in deep networks, ResNet processes preprocessed images, extracting features that are crucial for accurate digit recognition.

c) Training loop

The training loop involves multiple epochs, where the dataset is processed repeatedly to optimize the model's

performance. Each epoch involves adjusting the model weights to improve accuracy in recognizing specific digits.

d) Model Storage

After training, the model is saved for reuse, ensuring consistent performance without requiring retraining. This saved model is subsequently used in the EasyOCR-based detection system.

C. *Testing EasyOCR Method for Number Code Detection*

Testing the EasyOCR-based OCR system focuses on its ability to accurately detect and recognize numerical codes (specifically digits "1" through "4"). This section elaborates on the operational architecture and steps involved in using EasyOCR, along with the results obtained from number code detection.

a) Installing EasyOCR

EasyOCR and dependencies (like PyTorch) are installed in the development environment (e.g., VSCode) using pip. Once installed, the OCR system is configured to process images for number code detection.

b) Pre-processing

Pre -processing steps include:

1. **HSV Conversion:** Employed for analyzing and isolating specific colors in images.
2. **Noise Reduction:** Filtering out unwanted image noise to enhance clarity for OCR.
3. **Converting the Image to Grayscale:** Converting color images to grayscale simplifies data, focusing on shape and intensity, critical for accurate text detection.

The resulting preprocessed images are displayed in Fig. 29 to illustrate the effectiveness of each preprocessing step and the quality of images provided to the OCR model for further processing.

Fig. 29. Pre-processing results

c) Text Detection

Text regions within the image are detected using a deep learning-based text detection model, such as CRAFT (Character Region Awareness for Text). This model generates bounding boxes that delineate the detected text areas, as shown in Fig. 30, enabling the OCR system to isolate areas of interest effectively.

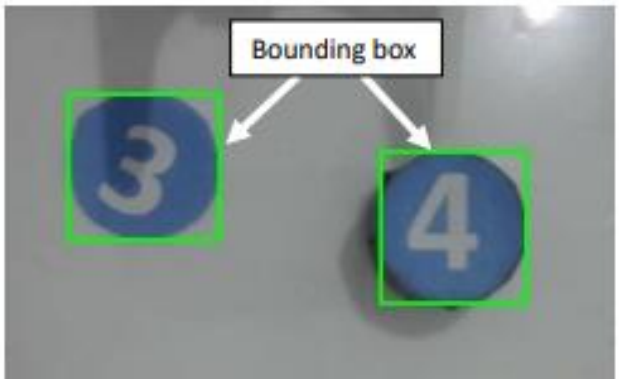

Fig. 30. Text detection

d) Feature Extraction

Feature extraction is conducted using Convolutional Neural Networks (CNNs), allowing the OCR model to recognize characters within the detected regions accurately. EasyOCR leverages these features to recognize text efficiently within VSCode, facilitating seamless character recognition without manual CNN implementation.

e) Displaying the Reading Results

The final OCR results, shown in Fig. 31, include bounding boxes around the detected text regions, with recognized text overlaid on the image. This visual representation confirms the accuracy of text detection and provides clear insight into the performance of EasyOCR in real-time number code recognition.

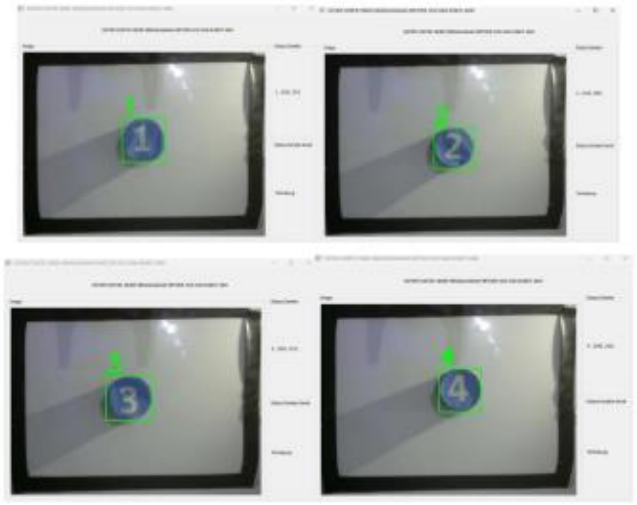

Fig. 31. Reading results

The number code reading test involved presenting digits "1" through "4" in front of the camera to verify the accuracy and reliability of EasyOCR in detecting these codes. The test outcomes, shown in Table I, confirm successful detection for each presented digit.

TABLE I. NUMBER DETECTION TESTING

No	Number Code	Status
1	1	Detected
2	2	Detected
3	3	Detected
4	4	Detected

D. *Testing the Inverse Kinematics Method for Object Position*

The implementation of the inverse kinematics method on the robot is a key step where theoretical calculations are translated into executable code, enabling precise robotic movement based on input coordinates. This process involves applying several calculations and adjustments to ensure that the robot responds correctly to various (x) and (y) coordinate inputs. Specific conditioning steps are included to calibrate the coordinates, allowing the robot to move right and left smoothly and accurately based on application commands. This approach ensures that the robot accurately reaches the desired positions in the workspace, improving overall performance in object manipulation tasks.

E. *Overall System Testing*

Testing of the ARM robot's movement is conducted once the number codes and object coordinates have been obtained. The purpose of this testing is to verify whether the ARM robot's movement aligns with the provided input.

a) Testing Robot Movement with Number Code '1'

The initial robot movement test uses the number code "1" with the object positioned on the workspace sheet (Fig. 32, Fig. 33, and Table II). The object's position is placed randomly, as long as it remains within the defined workspace.

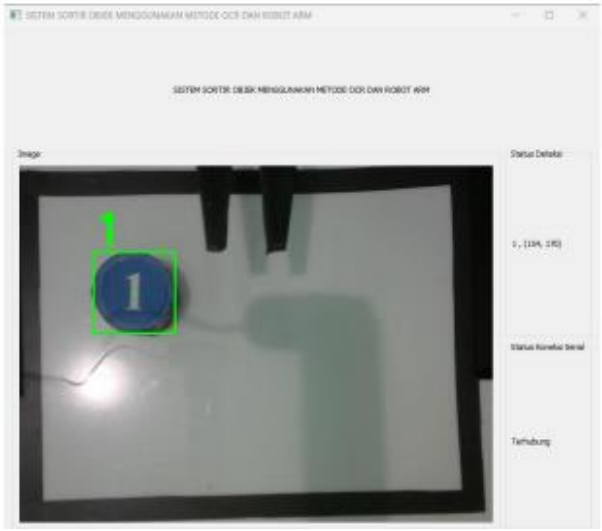

Fig. 32. First trial with number code '1'

In the first test, the ARM robot was able to detect and move the object to the specified location. During this initial test, the ARM robot performed as expected. The ARM robot moved the object from the coordinates (154, 170) pixels, which was converted to centimeters resulting in (6.6, 7.3) cm. To determine the servo motor movement values, the researcher used the inverse kinematics method that has been implemented in the programming language with the following formula:

Sumbu x = sX
Sumbu y = sY/23
y = 170/23
y = 7.3
dX = Sumbu x/6.6
dX = 154/6.6
dX = 23.3
L1 = dX + 50
L1 = 23.3 + 50
L1 = 73.3
L2 = 85 - ((sumY*10)/1.52)/30)
L2 = 85 – ((23.3 x 10 / 1.52) / 30)
L2 = 80.1
L3 =165 - (((sumY*10)/1.52)/40)
L3 = 165 – ((23.3 x 10 / 1.52) / 40)
L3 = 163.8 L4 = 65 + (((sumY*10)/1.52)/50)
L4 = 65 + ((23.3 x 10 / 1.52) / 50)
L4 = 66

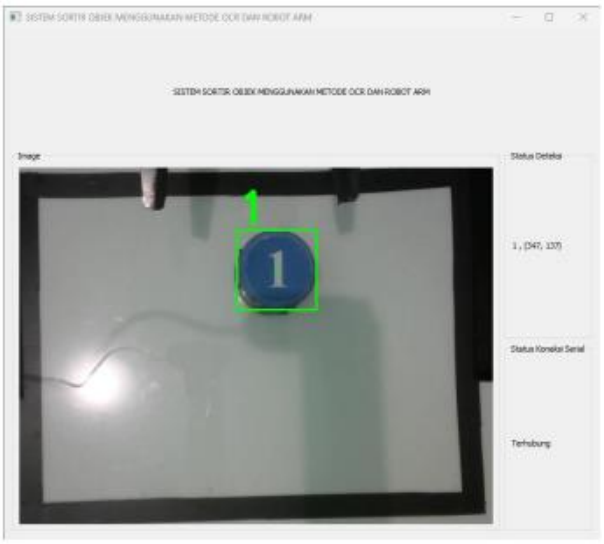

Fig. 33. Second trial with number code '1'

In the second test, the ARM robot was able to detect and move the object to the specified location. In this test, the ARM robot performed as expected. The ARM robot moved the object from the coordinates (347, 137) pixels, which was converted to cm resulting in (15.1, 5.9) cm. To determine the servo motor movement values, the researcher used the inverse kinematics method that has been implemented in the programming language with the following formula:

Axis x = sX
Axis y = sY/23
y = 137/23
y = 5.9
dX = Axisx/6.6
dX = 347/6.6
dX = 52.5
L1 = dX + 53
L1 = 52.5 + 50
L1 = 102.5
L2 = 55 - ((sumY*10)/1.52)/30)
L2 = 55 – ((23.3 x 10 / 1.52) / 30)
L2 = 83.4
L3 =130 - (((sumY*10)/1.52)/40)
L3 = 130 – ((23.3 x 10 / 1.52) / 40)
L3 = 179
L4 = 60 + (((sumY*10)/1.52)/50)
L4 = 60 + ((23.3 x 10 / 1.52) / 50)
L4 = 50.8

TABLE II. DATA COLLECTION TABLE FOR NUMBER CODE 1

Test No.	Code Number	Status	Position Coordinates (Px)		Position Output (Cm)		Servo Motor Movement				Testing Results
			X	Y	X	Y	L1	L2	L3	L4	
1	1	Detected	154	170	6.6	7.3	73.3	83.4	163.8	66.0	Successful
2	1	Detected	347	137	15.1	5.9	97.6	103.7	179.0	50.8	Successful
3	1	Detected	468	146	20.3	6.3	115.9	103.6	179.0	50.8	Successful
4	1	Detected	144	346	6.2	15.1	74.8	67.5	125.0	50.0	Successful
5	1	Detected	575	417	25	18.1	130.1	51.0	127.0	62.4	Failed
6	1	Detected	210	279	9.1	12.1	81.8	82.3	163.0	66.6	Successful
7	1	Detected	476	107	20.6	4.6	117.1	104.0	179.2	50.6	Successful
8	1	Detected	84	398	3.6	17.3	65.7	51.0	127.0	62.4	Successful
9	1	Detected	304	335	13.2	14.5	89.1	51.8	127.6	61.9	Successful
10	1	Detected	490	135	21.3	5.8	119.2	103.7	179.0	50.8	Successful
11	1	Detected	333	191	14.5	8.3	90.5	83.2	163.6	66.1	Successful
12	1	Detected	77	240	3.9	10.4	61.7	82.7	163.3	66.4	Successful
13	1	Detected	320	108	13.9	4.7	93.5	104.0	179.2	50.6	Successful
14	1	Detected	320	108	13.9	4.7	93.5	104.0	179.2	50.6	Successful
15	1	Detected	157	282	6.8	12.3	73.8	82.3	163.0	66.6	Successful
16	1	Detected	210	279	9.1	12.1	81.8	82.3	163.0	66.6	Successful
17	1	Detected	451	282	19.6	12.2	108.3	82.3	163.0	66.6	Successful
Success Level											94.11%

b) Testing Robot Movement with Number Code '2'

The first test of robot movement using number code '2' involved placing the object within the workspace area (Fig. 34, Fig. 35, Table III). The object's position was placed randomly as long as it remained within the defined workspace.

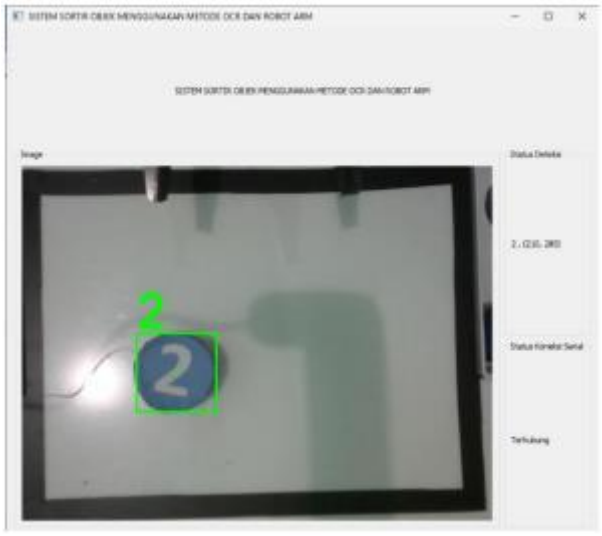

Fig. 34. First trial with number code '2'

In the first test, the ARM robot was able to detect and move the object to the specified location. In this test, the ARM robot performed as expected. The ARM robot moved the object from the coordinates (210, 279) pixels, which was converted to centimeters resulting in (9.1, 12.1) cm. To determine the servo motor movement values, the researcher used the inverse kinematics method that has been implemented in the programming language with the following formula:

Axis x = sX
Axis y = sY/23
y = 279/23
y = 12.1
dX = Axis x/6.6
dX = 210/6.6
dX = 31.8
L1 = dX + 50
L1 = 31.8 + 50
L1 = 81.8
L2 = 85 - ((sumY*10)/1.52)/30)
L2 = 85 – ((12.1 x 10 / 1.52) / 30)
L2 = 82,3
L3 =165 - (((sumY*10)/1.52)/40)
L3 = 165 – ((12.1 x 10 / 1.52) / 40)
L3 = 163
L4 = 65 + (((sumY*10)/1.52)/50)
L4 = 65 + ((12.1 x 10 / 1.52) / 50)
L4 = 66.5

Fig. 35. Second trial with number code '2'

In the second test, the ARM robot was able to detect and move the object to the specified location. In this test, the ARM robot performed as expected. The ARM robot moved the object from the coordinates (490, 135) pixels, which was converted to centimeters resulting in (21.3, 5.8) cm. To determine the servo motor movement values, the researcher used the inverse kinematics method that has been

implemented in the programming language with the following formula:

Axis x = sX
Axis y = sY / 23
y = 135 / 23
y = 5.8
dX = Axis x / 6.6
dX = 490 / 6.6
dX = 74.2
L1 = dX + 45
L1 = 74.2 + 45
L1 = 119.2
L2 = 105 - ((sumY*10)/1.52)/30)
L2 = 105 – ((5.8 x 10 / 1.52) / 30)
L2 = 179 L3 =180 - (((sumY*10)/1.52)/30)
L3 = 180 – ((5.8 x 10 / 1.52) / 30)
L3 = 179
L4 = 50 + (((sumY*10)/1.52)/50)
L4 = 50 + ((5l8 x 10 / 1.52) / 50)
L4 = 50

c) Robot Movement Testing with Code "3"

The first robot movement test uses code "3" with the object positioned within the workspace sheet (Fig. 36, Fig. 37, and Table IV). The object's position is placed randomly, as long as it remains within the defined workspace.

Fig. 36. First trial with code "3"

In the first trial, the ARM robot was able to detect and move the object to the specified location. In this test, the ARM robot is said to have performed as expected. The robot moved the object with coordinates (490, 135) pixels, which was converted to (21.3, 5.8) cm. To determine the servo motor movement values, the researcher used the inverse kinematics method, which was input into the programming language with the following formulas:

Axis x = sX
Axis y = sY/23
y = 135/23
y = 5.8
dX = Axis x/6.6
dX = 490/6.6
dX = 3.2
L1 = dX + 50 65
L1 = 3.2 + 45
L1 = 92.7
L2 = 105 - ((sumY*10)/1.52)/30)
L2 = 105 – ((23,3 x 10 / 1.52) / 40)
L2 = 103.6
L3 =180 - (((sumY*10)/1.52)/40)
L3 = 180 – ((23.3 x 10 / 1.52) / 40)
L3 = 179
L4 = 65 + (((sumY*10)/1.52)/50)
L4 = 65 + ((23.3 x 10 / 1.52) / 50)
L4 = 50.8

Fig. 37. Second trial with code "3"

In the second trial, the ARM robot was able to detect and move the object to the designated location as intended. In this trial, the robot ARM was considered to be functioning as desired. The robot ARM moved an object with coordinates (157, 335) pixels, which was converted to (6.8, 14.6) cm. To determine the servo motor movement values, the researchers used the inverse kinematics method, which was already implemented in the programming language with the following formula:

Axis x = sX
Axis y = sY / 23
y = 335 / 23
y = 14.5
dX = Axis x / 6.6
dX = 157/ 6.6 dX = 23.7
L1 = dX + 50
L1 = 23.7 + 50
L1 = 76.8
L2 = 85 - ((sumY*10)/1.52)/30)
L2 = 85 – ((14,5 x 10 / 1.52) / 30)
L2 = 81.8 L3 =165 - (((sumY*10)/1.52)/40)
L3 = 165 – ((14.5 x 10 / 1.52) / 40)
L3 = 161.8
L4 = 65 + (((sumY*10)/1.52)/50)
L4 = 65 + ((14.5 x 10 / 1.52) / 50)
L4 = 66.9

TABLE III. DATA COLLECTION FOR CODE NUMBER 2

Test No.	Code Number	Status	Position Coordinates (Px)		Position Output (Cm)		Servo Motor Movement				Testing Results
			X	Y	X	Y	L1	L2	L3	L4	
1	2	Detected	210	279	9.1	12.1	81.8	82.3	163.0	66.6	Successful
2	2	Detected	490	135	21.3	5.7	119.2	103.7	179.0	50.8	Successful
3	2	Detected	210	284	9.1	12.3	81.8	81.3	163.0	66.6	Successful
4	2	Detected	77	201	3.3	8.7	61.7	83.1	163.6	66.1	Successful
5	2	Detected	470	230	20.4	10	111.2	82.8	163.4	66.3	Successful
6	2	Detected	232	165	10	7.1	85.2	83.4	163.8	65.9	Successful
7	2	Detected	451	282	19.6	12.2	108.3	82.3	163.0	66.6	Successful
8	2	Detected	198	264	8.6	11.4	80.0	82.5	163.1	66.5	Successful
9	2	Detected	372	307	16.1	13.3	96.4	82.1	162.8	66.8	Successful
10	2	Detected	320	162	13.9	7.0	88.5	82.5	163.8	65.9	Successful
11	2	Detected	529	355	23.0	15.4	123.2	51.6	127.5	62.0	Successful
12	2	Detected	373	104	16.2	4.5	101.5	104.0	179.3	50.6	Successful
13	2	Detected	211	301	9.2	13.1	82.0	82.1	162.8	66.7	Failed
14	2	Detected	210	279	9.1	12.1	81.8	82.3	163.0	66.6	Successful
15	2	Detected	476	107	20.6	4.6	117.1	104.0	179.2	50.6	Successful
16	2	Detected	216	218	9.4	9.5	82.7	82.9	163.4	66.2	Successful
17	2	Detected	333	191	14.5	8.3	90.5	83.2	163.6	66.1	Successful
18	2	Detected	151	137	6.6	6	67.9	103.7	179.0	50.8	Successful
Success Level											94.11%

TABLE IV. DATA COLLECTION TABLE FOR NUMBER CODE 3

Test No	Code Number	Status	Position Coordinates (Px)		Position Output (Cm)		Servo Motor Movement				Testing Results
			X	Y	X	Y	L1	L2	L3	L4	
1	3	Detected	315	143	13.7	6.2	92.7	103.6	179.0	50.8	Successful
2	3	Detected	157	335	6.8	14.6	76.8	81.8	161.8	66.9	Successful
3	3	Detected	117	164	5.1	7.1	67.7	83.4	163.8	65.9	Successful
4	3	Detected	585	403	25.4	17.5	131.6	51.2	127.1	62.3	Failed
5	3	Detected	333	191	14.5	8.3	90.5	83.2	163.6	66.1	Successful
6	3	Detected	77	240	3.9	10.4	61.7	82.7	163.3	66.4	Successful
7	3	Detected	320	108	13.9	4.7	93.5	104.0	179.2	50.6	Successful
8	3	Detected	157	282	6.8	12.3	73.8	82.3	163.0	66.6	Successful
9	3	Detected	216	218	9.4	9.5	82.7	82.9	163.4	66.2	Successful
10	3	Detected	151	137	6.6	6	67.9	103.7	179.0	50.8	Successful
11	3	Detected	310	209	13.5	9.1	87.0	83.0	163.5	66.2	Successful
12	3	Detected	198	264	8.6	11.4	80.0	82.5	163.1	66.5	Successful
13	3	Detected	451	282	19.6	12.2	108.3	82.3	163.0	66.6	Successful
14	3	Detected	232	165	10	7.1	85.2	83.4	163.9	65.9	Successful
16	3	Detected	529	355	23.0	15.4	123.2	51.6	127.5	62.0	Successful
17	3	Detected	373	104	16.2	4.5	101.5	104.0	179.3	50.6	Successful
18	3	Detected	320	108	13.9	4.7	93.5	104.0	179.2	50.6	Successful
19	3	Detected	372	307	16.1	13.3	96.4	82.1	162.8	66.8	Successful
Success Level											94.7%

d) Testing Robot Movement with Code Number "4"

The first test involves using code number "4" for the robot's movement (Fig. 38, Fig. 39, and Table V). The object is positioned randomly within the defined workspace dimensions.

In the first trial, the ARM robot was able to detect and move the object to the designated location. In this test, the ARM robot performed as expected. The robot moved the object from the coordinates (480, 283) pixels, which were converted to (20.9, 12.3) cm. To calculate the servo motor movement values, the researcher used the inverse kinematics method, which was implemented in the programming language with the following formulas:

Axis x = sX
Axis y = sY / 23
y = 283 / 23
y = 12.3

dX = Axis x / 6.6
dX = 480/ 6.6
dX = 72.7
L1 = dX + 50
L1 = 72.7 + 50
L1 = 112.7
L2 = 85 - ((sumY*10)/1.52)/30)
L2 = 85 – ((12,3 x 10 / 1.52) / 30)
L2 = 82.3 L3 =165 - (((sumY*10)/1.52)/40)
L3 = 165 – ((12.3 x 10 / 1.52) / 40)
L3 = 163
L4 = 65 + (((sumY*10)/1.52)/50)
L4 = 65 + ((12,3 x 10 / 1.52) / 50)
L4 = 66.6

Fig. 38. Second trial with code number "4"

In the first trial, the ARM robot was able to detect and move the object to the designated location. In this test, the ARM robot performed as expected. The robot moved the object from the coordinates (480, 283) pixels, which were converted to (20.9, 12.3) cm. To calculate the servo motor movement values, the researcher used the inverse kinematics method, which was implemented in the programming language with the following formulas:

Axis x = sX
Axis y = sY / 23
y = 215 / 23
y = 9.3
dX = Axis x/6.6
dX = 197/ 6.6
dX = 29.8
L1 = dX + 50
L1 = 29.8 + 50
L1 = 79.8
L2 = 85 - ((sumY*10)/1.52)/30)
L2 = 85 – ((9.3 x 10 / 1.52) / 30)
L2 = 83 70
L3 =165 - (((sumY*10)/1.52)/40)
L3 = 165 – ((9.3 x 10 / 1.52) / 40)
L3 = 163.5
L4 = 65 + (((sumY*10)/1.52)/50)
L4 = 65 + ((9.3 x 10 / 1.52) / 50)
L4 = 66.2

Fig. 39. Second trial with number code "4"

TABLE V. DATA COLLECTION TABLE FOR NUMBER CODE 4

Test No	Code Number	Status	Position Coordinates (Px)		Position Output (Cm)		Servo Motor Movement				Testing Results
			X	Y	X	Y	L1	L2	L3	L4	
1	4	Detected	480	283	20.9	12.3	112.7	82.3	163.0	66.6	Successful
2	4	Detected	197	215	8.6	9.3	79.8	83.0	163.5	65.2	Successful
3	4	Detected	138	408	6.0	17.7	73.9	83.4	162.8	65.9	Successful
4	4	Detected	210	209	13.5	9.1	87.0	83.0	163.5	66.2	Successful
5	4	Detected	529	355	23.0	15.4	123.2	51.6	127.5	62.0	Successful
6	4	Detected	373	104	16.2	4.5	101.5	104.0	179.3	50.6	Successful
7	4	Detected	211	301	9.2	13.1	82.0	82.1	162.8	66.7	Failed
8	4	Detected	436	134	19.0	5.8	111.1	103.7	179.0	50.8	Successful
9	4	Detected	372	307	16.2	13.3	96.4	82.1	162.8	66.8	Successful
10	4	Detected	232	165	10	7.1	85.2	83.4	163.8	65.9	Successful
11	4	Detected	157	282	6.8	12.3	73.8	82.3	163.0	66.6	Successful
12	4	Detected	77	201	3.3	8.7	61.7	83.1	163.6	66.1	Successful
13	4	Detected	320	108	13.9	4.7	93.5	104.0	179.2	50.6	Successful
14	4	Detected	347	137	15.1	5.9	97.6	103.7	179.0	50.8	Successful
16	4	Detected	468	146	20.3	6.3	115.9	103.6	179.0	50.8	Successful
17	4	Detected	117	164	5.1	7.1	67.7	83.4	163.8	65.9	Successful
18	4	Detected	451	282	19.6	12.2	108.3	82.3	163.0	66.6	Successful
Success Level											94.44%

V. CONCLUSION

This study successfully developed a 4-DOF robotic arm system capable of recognizing and manipulating objects based on numerical codes with high precision, achieved through the integration of EasyOCR for code recognition and a control mechanism ensuring accurate robotic arm positioning. The testing results indicate a success rate exceeding 94% in detecting and positioning objects with numerical codes 1 through 4, with an average positioning error below 1.5 degrees. The system adjusts arm length by approximately 30 to 50 mm to optimize positioning. These findings suggest that combining computer vision-based OCR with inverse kinematics enhances accuracy in robotic tasks requiring precision.

However, the system's performance is affected by environmental factors, such as lighting conditions, which impact code readability. Minor discrepancies in servo angle positioning necessitate further calibration in kinematic calculations to maintain precise object positioning. These observations underscore the need for further refinement in calibration to enhance the system's adaptability in real-world conditions.

Future research should focus on developing more robust code recognition algorithms using deep learning models specifically trained for various lighting conditions. Expanding the system's functionality to handle a broader range of object shapes and sizes would also increase its applicability in industries requiring automated, high-precision object handling, such as manufacturing and logistics.

The contribution of this research lies in presenting a reliable and adaptable robotic system that effectively integrates OCR-based code recognition with robotic arm control for precise object manipulation. These findings serve as a foundation for further research in computer vision-based robotic automation, demonstrating the potential of this system to improve operational efficiency across various industrial sectors.

REFERENCES

[1] K. Suphalak, N. Klanpet, N. Sikaressakul, and S. Prongnuch, "Robot Arm Control System via Ethernet with Kinect V2 Camera for use in Hazardous Areas," in *2024 1st International Conference on Robotics, Engineering, Science, and Technology (RESTCON)*, pp. 175–180, 2024, doi: 10.1109/RESTCON60981.2024.10463582.

[2] J. W. Lee and S. Jung, "Design of a Foldable Robot Arm for a Hybrid Robot Manipulator," in *2022 22nd International Conference on Control, Automation and Systems (ICCAS)*, pp. 325–327, 2022, doi: 10.23919/ICCAS55662.2022.10003830.

[3] J. Lin, Z. Gu, A. M. Amir, X. Chen, K. Ashim, and K. Shi, "A fast humanoid robot arm for boxing based on servo motors," in *2021 International Conference on High Performance Big Data and Intelligent Systems (HPBD&IS)*, pp. 252–255, 2021, doi: 10.1109/HPBDIS53214.2021.9658471.

[4] Z. Ben Hazem, R. Ince, and S. Dilibal, "Joint Control Implementation of 4-DOF Robotic Arm Using Robot Operating System," in *2022 International Conference on Theoretical and Applied Computer Science and Engineering (ICTASCE)*, pp. 72–77, 2022, doi: 10.1109/ICTACSE50438.2022.10009733.

[5] S. Mori, K. Tanaka, S. Nishikawa, R. Niiyama, and Y. Kuniyoshi, "High-Speed Humanoid Robot Arm for Badminton Using Pneumatic-Electric Hybrid Actuators," *IEEE Robot Autom Lett*, vol. 4, no. 4, pp. 3601–3608, 2019, doi: 10.1109/LRA.2019.2928778.

[6] A. R. Al Tahtawi, M. Agni, and T. D. Hendrawati, "Small-scale robot arm design with pick and place mission based on inverse kinematics," *Journal of Robotics and Control (JRC)*, vol. 2, no. 6, pp. 469–475, Nov. 2021, doi: 10.18196/jrc.26124.

[7] W. Wei, K. Kurita, J. Kuang, and A. Gao, "Real-Time 3D Arm Motion Tracking Using the 6-axis IMU Sensor of a Smartwatch," in *2021 IEEE 17th International Conference on Wearable and Implantable Body Sensor Networks (BSN)*, pp. 1–4, 2021, doi: 10.1109/BSN51625.2021.9507012.

[8] P. Rosenberger *et al.*, "Object-Independent Human-to-Robot Handovers Using Real Time Robotic Vision," *IEEE Robot Autom Lett*, vol. 6, no. 1, pp. 17–23, 2021, doi: 10.1109/LRA.2020.3026970.

[9] Y. Yun, S. J. Lee, and S.-J. Kang, "Motion Recognition-Based Robot Arm Control System Using Head Mounted Display," *IEEE Access*, vol. 8, pp. 15017–15026, 2020, doi: 10.1109/ACCESS.2020.2964801.

[10] F. Sun, Y. Chen, Y. Wu, L. Li, and X. Ren, "Motion Planning and Cooperative Manipulation for Mobile Robots With Dual Arms," *IEEE Trans Emerg Top Comput Intell*, vol. 6, no. 6, pp. 1345–1356, 2022, doi: 10.1109/TETCI.2022.3146387.

[11] Z. Wang *et al.*, "Vision-Based Calibration of Dual RCM-Based Robot Arms in Human-Robot Collaborative Minimally Invasive Surgery," *IEEE Robot Autom Lett*, vol. 3, no. 2, pp. 672–679, 2018, doi: 10.1109/LRA.2017.2737485.

[12] Y.-S. L.-K. Cio, M. Raison, C. Leblond Ménard, and S. Achiche, "Proof of Concept of an Assistive Robotic Arm Control Using Artificial Stereovision and Eye-Tracking," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 27, no. 12, pp. 2344–2352, 2019, doi: 10.1109/TNSRE.2019.2950619.

[13] J. Sigut, M. Castro, R. Arnay, and M. Sigut, "OpenCV Basics: A Mobile Application to Support the Teaching of Computer Vision Concepts," *IEEE Transactions on Education*, vol. 63, no. 4, pp. 328–335, 2020, doi: 10.1109/TE.2020.2993013.

[14] Y. Xu, H. Zhang, L. Cao, X. Shu, and D. Zhang, "A Shared Control Strategy for Reach and Grasp of Multiple Objects Using Robot Vision and Noninvasive Brain–Computer Interface," *IEEE Transactions on Automation Science and Engineering*, vol. 19, no. 1, pp. 360–372, 2022, doi: 10.1109/TASE.2020.3034826.

[15] S. Li, N. Hendrich, H. Liang, P. Ruppel, C. Zhang, and J. Zhang, "A Dexterous Hand-Arm Teleoperation System Based on Hand Pose Estimation and Active Vision," *IEEE Trans Cybern*, vol. 54, no. 3, pp. 1417–1428, 2024, doi: 10.1109/TCYB.2022.3207290.

[16] X. Chen, X. Huang, Y. Wang, and X. Gao, "Combination of Augmented Reality Based Brain- Computer Interface and Computer Vision for High-Level Control of a Robotic Arm," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 28, no. 12, pp. 3140–3147, 2020, doi: 10.1109/TNSRE.2020.3038209.

[17] Y. Zhou *et al.*, "Shared Three-Dimensional Robotic Arm Control Based on Asynchronous BCI and Computer Vision," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 31, pp. 3163–3175, 2023, doi: 10.1109/TNSRE.2023.3299350.

[18] M. F. El-Khatib and S. A. Maged, "Low level position control for 4-DOF arm robot using fuzzy logic controller and 2-DOF PID controller," in *2021 International Mobile, Intelligent, and Ubiquitous Computing Conference (MIUCC)*, pp. 258–262, 2021, doi: 10.1109/MIUCC52538.2021.9447617.

[19] T. A. Mai, T. S. Dang, D. N. Anisimov, and E. Fedorova, "Fuzzy-PID Controller for Two Wheels Balancing Robot Based on STM32 Microcontroller," in *2019 International Conference on Engineering Technologies and Computer Science (EnT)*, pp. 20–24, 2019, doi: 10.1109/EnT.2019.00009.

[20] B. Rouzbeh, G. M. Bone, G. Ashby, and E. Li, "Design, Implementation and Control of an Improved Hybrid Pneumatic-Electric Actuator for Robot Arms," *IEEE Access*, vol. 7, pp. 14699–14713, 2019, doi: 10.1109/ACCESS.2019.2891532.

[21] M. Syakir, E. S. Ningrum, and I. Adji Sulistijono, "Teleoperation Robot Arm using Depth Sensor," in *2019 International Electronics Symposium (IES)*, pp. 394–399, 2019, doi: 10.1109/ELECSYM.2019.8901679.

[22] N. Tan, P. Yu, Z. Zhong, and F. Ni, "A New Noise-Tolerant Dual-Neural-Network Scheme for Robust Kinematic Control of Robotic

Arms With Unknown Models," *IEEE/CAA Journal of Automatica Sinica*, vol. 9, no. 10, pp. 1778–1791, 2022, doi: 10.1109/JAS.2022.105869.

[23] A. W. L. Yao and H. C. Chen, "An Intelligent Color Image Recognition and Mobile Control System for Robotic Arm," *International Journal of Robotics and Control Systems*, vol. 2, no. 1, pp. 97–104, 2022, doi: 10.31763/ijrcs.v2i1.557.

[24] Y. Zhang, L. Sun, and Y. Zhang, "Research on Algorithm of Humanoid Robot Arm Control System Based on Fuzzy PID Control," in *2022 International Conference on Artificial Intelligence and Autonomous Robot Systems (AIARS)*, pp. 337–341, 2022, doi: 10.1109/AIARS57204.2022.00082.

[25] T. P. Cabré, M. T. Cairol, D. F. Calafell, M. T. Ribes, and J. P. Roca, "Project-Based Learning Example: Controlling an Educational Robotic Arm With Computer Vision," *IEEE Revista Iberoamericana de Tecnologias del Aprendizaje*, vol. 8, no. 3, pp. 135–142, 2013, doi: 10.1109/RITA.2013.2273114.

[26] Md. K. Raihan, H. H. Alvee, and M. R. Tanvir Hossain, "Manipulation and Analysis of a 4-DOF Robotic Arm Using a Peer-Peer Messaging System," in *2019 International Conference on Electrical, Computer and Communication Engineering (ECCE)*, pp. 1–4, 2019, doi: 10.1109/ECACE.2019.8679497.

[27] A. Tai, M. Chun, Y. Gan, M. Selamet, and H. Lipson, "PARA: A one-meter reach, two-kg payload, three-DoF open source robotic arm with customizable end effector," *HardwareX*, vol. 10, p. e00209, 2021, doi: https://doi.org/10.1016/j.ohx.2021.e00209.

[28] H. Gokul, S. V Kanna, H. Akshay Kumar, and V. Ravikumar, "Design of Imitative Control Modalities for a 3 Degree of Freedom Robotic Arm," in *2020 4th International Conference on Computer, Communication and Signal Processing (ICCCSP)*, pp. 1–6, 2020, doi: 10.1109/ICCCSP49186.2020.9315273.

[29] S. P, S. S. D, A. A. M, P. M, and V. Peroumal, "Design of Robotic Arm with Three Degree of Freedom (DOF) operated by Bluetooth enabled Smartphones," in *2024 International Conference on Intelligent and Innovative Technologies in Computing, Electrical and Electronics (IITCEE)*, pp. 1–6, 2024, doi: 10.1109/IITCEE59897.2024.10467660.

[30] S. Bouzoualegh, E.-H. Guechi, and Y. Zennir, "Model Predictive Control of a Three Degrees of Freedom Manipulator Robot," in *2019 International Conference on Advanced Systems and Emergent Technologies (IC_ASET)*, pp. 84–89, 2019, doi: 10.1109/ASET.2019.8870999.

[31] A. Matthew, T. A. Tamba, and A. Sadiyoko, "Construction and Path Generation Algorithm Design of a Four Degrees of Freedom Robot Arm," in *2023 3rd International Conference on Smart Cities, Automation & Intelligent Computing Systems (ICON-SONICS)*, pp. 165–170, 2023, doi: 10.1109/ICON-SONICS59898.2023.10435330.

[32] Y.-C. Lin and C.-C. Peng, "Command Generation of a High Degree of Freedom Robot Manipulator on a Moving Platform," in *2019 IEEE International Conference on Cybernetics and Intelligent Systems (CIS) and IEEE Conference on Robotics, Automation and Mechatronics (RAM)*, pp. 512–517, 2019, doi: 10.1109/CIS-RAM47153.2019.9095787.

[33] A. A. Mohammed, H. R. Abdul Ameer, and D. S. Abdul-Zahra, "Design of a Linear Mathematical Model to Control the Manipulator of a Robotic Arm with a Hexagonal Degree of Freedom," in *2022 3rd Information Technology To Enhance e-learning and Other Application (IT-ELA)*, pp. 181–185, 2022, doi: 10.1109/IT-ELA57378.2022.10107941.

[34] A. Elmogy, Y. Bouteraa, and W. Elawady, "An adaptive fuzzy self-tuning inverse kinematics approach for robot manipulators," *Journal of control engineering and applied informatics*, vol. 22, no. 4, pp. 43-51, 2020.

[35] S. Fuenzalida, K. Toapanta, J. Paillacho, and D. Paillacho, "Forward and Inverse Kinematics of a Humanoid Robot Head for Social Human Robot-Interaction," in *2019 IEEE Fourth Ecuador Technical Chapters Meeting (ETCM)*, pp. 1–4, 2019, doi: 10.1109/ETCM48019.2019.9014887.

[36] Y. Pan, J. Tang, Q. Zhao, and S. Zhu, "Forward and Inverse Kinematics Modeling and Simulation of Six-axis Joint Robot Arm Based on Exponential Product Method," in *2020 IEEE 3rd International Conference on Automation, Electronics and Electrical Engineering (AUTEEE)*, pp. 372–375, 2020, doi: 10.1109/AUTEEE50969.2020.9315719.

[37] N. Zivkovic, J. Vidakovic, S. Mitrovic, and M. Lazarevic, "Implementation of Dual Quaternion-based Robot Forward Kinematics Algorithm in ROS," in *2022 11th Mediterranean Conference on Embedded Computing (MECO)*, pp. 1–4, 2022, doi: 10.1109/MECO55406.2022.9797160.

[38] H. P. Nurba, D. Hadian, N. Lestari, K. A. Munastha, H. Mistialustina, and E. Rachmawati, "Performance Evaluation of 3 DOF Arm Robot With Forward Kinematics Denavit-Hartenberg Method For Coffee Maker Machine," in *2022 16th International Conference on Telecommunication Systems, Services, and Applications (TSSA)*, pp. 1–6, 2022, doi: 10.1109/TSSA56819.2022.10063918.

[39] S. Savin, O. Balakhnov, and A. Klimchik, "Energy-based local forward and inverse kinematics methods for tensegrity robots," in *2020 Fourth IEEE International Conference on Robotic Computing (IRC)*, pp. 280–284, 2020, doi: 10.1109/IRC.2020.00051.

[40] J. Lai, J. Yang, and J. Yan, "Application of Improved GA-BPNN Algorithm for Forward Kinematics Problem Solving of Parallel Robots," in *2022 7th International Conference on Mechatronics System and Robots (ICMSR)*, pp. 9–14, 2022, doi: 10.1109/ICMSR2020.2022.00010.

[41] Y. Jusman *et al.*, "Comparison between Support Vector Machine and K-Nearest Neighbor Algorithms for Leukemia Images Classification Using Shape Features," in *2021 25th International Computer Science and Engineering Conference (ICSEC)*, pp. 70–74, 2021, doi: 10.1109/ICSEC53205.2021.9684585.

[42] F. Amoros, L. Paya, W. Mayol-Cuevas, L. M. Jimenez, and O. Reinoso, "Holistic Descriptors of Omnidirectional Color Images and Their Performance in Estimation of Position and Orientation," *IEEE Access*, vol. 8, pp. 81822–81848, 2020, doi: 10.1109/ACCESS.2020.2990996.

[43] A. F. Rizky, N. Yudistira, and E. Santoso, "Text recognition on images using pre-trained CNN," *arXiv preprint arXiv:2302.05105*, 2023.

[44] E. Turajlic, "Multilevel Image Thresholding Based on Otsu's Method and Multi-swarm Particle Swarm Optimization Algorithm," in *2024 47th MIPRO ICT and Electronics Convention (MIPRO)*, pp. 43–47, 2024, doi: 10.1109/MIPRO60963.2024.10569522.

[45] L. Lin and S. Yang, "Weighted Two-dimensional Otsu Threshold Approached for Image Segmentation," in *2019 IEEE Symposium Series on Computational Intelligence (SSCI)*, pp. 1002–1006, 2019, doi: 10.1109/SSCI44817.2019.9002689.

[46] K. Li, L. Bai, Y. Li, and M. Feng, "Improved Otsu Multi-Threshold Image Segmentation Method based on Sailfish Optimization," in *2021 33rd Chinese Control and Decision Conference (CCDC)*, pp. 1869–1874, 2021, doi: 10.1109/CCDC52312.2021.9601664.

[47] C. Chen, B. Ye, J. Wu, X. Wang, W. Deng, and J. Bao, "Eddy Current C-scan Image Segmentation Based on Otsu Threshold Method," in *2019 IEEE 8th Data Driven Control and Learning Systems Conference (DDCLS)*, pp. 754–75, 2019, doi: 10.1109/DDCLS.2019.8908855.

[48] H. El Khoukhi, Y. Filali, A. Yahyaouy, M. A. Sabri, and A. Aarab, "A hardware Implementation of OTSU Thresholding Method for Skin Cancer Image Segmentation," in *2019 International Conference on Wireless Technologies, Embedded and Intelligent Systems (WITS)*, pp. 1–5, 2019, doi: 10.1109/WITS.2019.8723815.

[49] J. Chao, Y. Xiaoxiao, and W. Xiaohai, "Algorithm of Double Threshold Image Segmentation Combined QGA with Two-Dimensional Otsu," in *2020 5th International Conference on Mechanical, Control and Computer Engineering (ICMCCE)*, pp. 2219–2223, 2020, doi: 10.1109/ICMCCE51767.2020.00481.

[50] R. Chen, "A PCB Image Self-adaption Threshold Segmentation Method Fusing Color Information and OTSU Theory," in *2019 IEEE 8th Joint International Information Technology and Artificial Intelligence Conference (ITAIC)*, pp. 652–656, 2019, doi: 10.1109/ITAIC.2019.8785606.

[51] Y.-S. Lee and H.-J. Kim, "High-Speed Multilevel Binary Imaging CMOS Image Sensor for Object Feature Extraction," *IEEE Sens J*, vol. 22, no. 16, pp. 15934–15943, 2022, doi: 10.1109/JSEN.2022.3189653.

[52] R. Hooda and W. D. Pan, "Tree Based Search Algorithm for Binary Image Compression," in *2019 SoutheastCon*, pp. 1–6, 2019, doi: 10.1109/SoutheastCon42311.2019.9020376.

[53] I. R. Paulino, "Practical Bulk Denoising Of Large Binary Images," in *2022 IEEE International Conference on Image Processing (ICIP)*, pp. 196–200, 2022, doi: 10.1109/ICIP46576.2022.9897678.

[54] B. Madoš, A. Baláž, N. Ádám, J. Hurtuk, and Z. Bilanová, "Algorithm Design for User Experience Enhancement of Volume Dataset Reading from Storage Using 3D Binary Image as the Metadata," in *2019 IEEE 17th World Symposium on Applied Machine Intelligence and Informatics (SAMI)*, pp. 269–274, 2019, doi: 10.1109/SAMI.2019.8782726.

[55] A. A. Y. Mustafa, "An Image Mapping Approach for Quick Dissimilarity Detection of Binary Images," in *2019 International Conference on Image and Vision Computing New Zealand (IVCNZ)*, pp. 1–4, 2019, doi: 10.1109/IVCNZ48456.2019.8961029.

[56] R. Duan, Y. Liao, and S. Wang, "Adaptive Morphological Analysis Method and Its Application for Bearing Fault Diagnosis," *IEEE Trans Instrum Meas*, vol. 70, pp. 1–10, 2021, doi: 10.1109/TIM.2021.3072116.

[57] J. Yu, J. Huang, C. Liu, and B. Xia, "Fault Feature of Gearbox Vibration Signals Based on Morphological Filter Dynamic Convolution Autoencoder," *IEEE Sens J*, vol. 22, no. 23, pp. 22931–22942, 2022, doi: 10.1109/JSEN.2022.3213783.

[58] Y. Li *et al.*, "A Gradient-Constrained Morphological Operation for Retrieving Subcanopy Topography Over Densely Forested Areas From ICESat-2/ATL03 Data," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 62, pp. 1–13, 2024, doi: 10.1109/TGRS.2024.3404007.

[59] K. Cho, S.-E. Park, J.-H. Cho, H. Moon, and S.-H. Han, "Automatic Urban Area Extraction From SAR Image Based on Morphological Operator," *IEEE Geoscience and Remote Sensing Letters*, vol. 18, no. 5, pp. 831–835, 2021, doi: 10.1109/LGRS.2020.2989461.

[60] M. Zhang, W. Li, X. Zhao, H. Liu, R. Tao, and Q. Du, "Morphological Transformation and Spatial-Logical Aggregation for Tree Species Classification Using Hyperspectral Imagery," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 61, pp. 1–12, 2023, doi: 10.1109/TGRS.2022.3233847.

[61] H. Zhao *et al.*, "Automatic Lumen Segmentation in Intravascular Optical Coherence Tomography Using Morphological Features," *IEEE Access*, vol. 7, pp. 88859–88869, 2019, doi: 10.1109/ACCESS.2019.2925917.

[62] Z. Wu, C. Fang, G. Wu, Z. Lin, and W. Chen, "A CNN-Regression-Based Contact Erosion Measurement Method for AC Contactors," *IEEE Trans Instrum Meas*, vol. 71, pp. 1–10, 2022, doi: 10.1109/TIM.2022.3192282.

[63] Y. Yang, L. Wang, H. Yu, G. Lu, and Z. Xiao, "Image segmentation method based on RGB - D fusion," in *2020 International Conference on Virtual Reality and Visualization (ICVRV)*, pp. 225–230, 2020, doi: 10.1109/ICVRV51359.2020.00053.

[64] T. Singh and S. Karanchery, "Universal Image Segmentation Technique for Cancer Detection in Medical Images," in *2019 10th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, pp. 1–7, 2019, doi: 10.1109/ICCCNT45670.2019.8944598.

[65] Z. Liang, "Fast Image Segmentation and Animation Generation Algorithm Based on Depth Image Sequence," in *2022 2nd International Conference on Networking, Communications and Information Technology (NetCIT)*, pp. 648–651, 2022, doi: 10.1109/NetCIT57419.2022.00150.

[66] A. Abdulrahman and S. Varol, "A Review of Image Segmentation Using MATLAB Environment," in *2020 8th International Symposium on Digital Forensics and Security (ISDFS)*, pp. 1–5, 2020, doi: 10.1109/ISDFS49300.2020.9116191.

[67] H. Yang, "Graphic Image Segmentation Method based on High-Precision and Fast Algorithm," in *2023 Global Conference on Information Technologies and Communications (GCITC)*, pp. 1–4, 2023, doi: 10.1109/GCITC60406.2023.10425884.

[68] W. Honghui and L. Xiaojing, "Summary on Thangka Image Segmentation," in *2020 International Conference on Intelligent Computing, Automation and Systems (ICICAS)*, pp. 72–77, 2020, doi: 10.1109/ICICAS51530.2020.00022.

[69] Y. Cheng and B. Li, "Image Segmentation Technology and Its Application in Digital Image Processing," in *2021 IEEE Asia-Pacific Conference on Image Processing, Electronics and Computers (IPEC)*, pp. 1174–1177, 2021, doi: 10.1109/IPEC51340.2021.9421206.

[70] W. Zhang, J. Wang, and X. Liu, "Adaptive Sidelobe Suppression of SAR Images with Arbitrary Doppler Centroids and Bandwidths," in *IGARSS 2020 - 2020 IEEE International Geoscience and Remote Sensing Symposium*, pp. 909–912, 2020, doi: 10.1109/IGARSS39084.2020.9323697.

[71] A. Kaluthantrige, J. Feng, J. Gíl-Fernández, and A. Pellacani, "Centroid regression using CNN-based Image Processing Algorithm with application to a binary asteroid system," in *2022 IEEE Congress on Evolutionary Computation (CEC)*, pp. 1–7, 2022, doi: 10.1109/CEC55065.2022.9870217.

[72] K. Imane, A. Adnane, and G. Zouhair, "Sun Centroid Extraction Algorithm for Satellite based on Black Sun Effect," in *2022 IEEE 3rd International Conference on Electronics, Control, Optimization and Computer Science (ICECOCS)*, pp. 1–4, 2022, doi: 10.1109/ICECOCS55148.2022.9983232.

[73] R. M, S. S.B, A. N, A. G. A B, and V. Patil, "Centroiding and Connected Component Labeling for Radar Images Using Image Processing Algorithms," in *2020 International Conference on Smart Technologies in Computing, Electrical and Electronics (ICSTCEE)*, pp. 50–54, 2020, doi: 10.1109/ICSTCEE49637.2020.9277394.

[74] S. Ganesh, P. T. Bhatti, M. Alkhalaf, S. Gupta, A. J. Shah, and S. Tridandapani, "Combining Optical Character Recognition With Paper ECG Digitization," *IEEE J Transl Eng Health Med*, vol. 9, pp. 1–9, 2021, doi: 10.1109/JTEHM.2021.3083482.

[75] X. Wang *et al.*, "Intelligent Micron Optical Character Recognition of DFB Chip Using Deep Convolutional Neural Network," *IEEE Trans Instrum Meas*, vol. 71, pp. 1–9, 2022, doi: 10.1109/TIM.2022.3154831.

[76] J. Galvis, S. Morales, C. Kasmi, and F. Vega, "Denoising of Video Frames Resulting From Video Interface Leakage Using Deep Learning for Efficient Optical Character Recognition," *IEEE Letters on Electromagnetic Compatibility Practice and Applications*, vol. 3, no. 2, pp. 82–86, 2021, doi: 10.1109/LEMCPA.2021.3073663.

[77] C. Zhang *et al.*, "A Machine Vision-Based Character Recognition System for Suspension Insulator Iron Caps," *IEEE Trans Instrum Meas*, vol. 72, pp. 1–13, 2023, doi: 10.1109/TIM.2023.3300474.

[78] R. Buoy, M. Iwamura, S. Srun, and K. Kise, "Toward a Low-Resource Non-Latin-Complete Baseline: An Exploration of Khmer Optical Character Recognition," *IEEE Access*, vol. 11, pp. 128044–128060, 2023, doi: 10.1109/ACCESS.2023.3332361.

[79] J. Memon, M. Sami, R. A. Khan, and M. Uddin, "Handwritten Optical Character Recognition (OCR): A Comprehensive Systematic Literature Review (SLR)," *IEEE Access*, vol. 8, pp. 142642–142668, 2020, doi: 10.1109/ACCESS.2020.3012542.

[80] A. T. Sahlol, M. Abd Elaziz, M. A. A. Al-Qaness, and S. Kim, "Handwritten Arabic Optical Character Recognition Approach Based on Hybrid Whale Optimization Algorithm With Neighborhood Rough Set," *IEEE Access*, vol. 8, pp. 23011–23021, 2020, doi: 10.1109/ACCESS.2020.2970438.

[81] R. E. Nalawati, D. Y. Liliana, R. W. Iswara, M. D. Nashshar, M. S. Rumika Damayanti, and R. M. Shaliha, "Vehicle Number Plate Recognition in Intelligent Transportation System using YOLO v8 and EasyOCR," in *2023 11th International Conference on Cyber and IT Service Management (CITSM)*, pp. 1–5, 2023, doi: 10.1109/CITSM60085.2023.10455441.

[82] A. D. Iriawan and A. Sunyoto, "Automatic License Plate Recognition System in Indonesia Using YOLOv8 and EasyOCR Algorithm," in *2023 6th International Conference on Information and Communications Technology (ICOIACT)*, pp. 384–3882023, doi: 10.1109/ICOIACT59844.2023.10455908.

[83] S. Dhyani and V. Kumar, "Real-Time License Plate Detection and Recognition System using YOLOv7x and EasyOCR," in *2023 Global Conference on Information Technologies and Communications (GCITC)*, pp. 1–5, 2023, doi: 10.1109/GCITC60406.2023.10425814.

[84] D. R. Vedhaviyassh, R. Sudhan, G. Saranya, M. Safa, and D. Arun, "Comparative Analysis of EasyOCR and TesseractOCR for Automatic License Plate Recognition using Deep Learning Algorithm," in *2022 6th International Conference on Electronics, Communication and Aerospace Technology*, pp. 966–971, 2022, doi: 10.1109/ICECA55336.2022.10009215.

[85] U. Kulkarni, S. Agasimani, P. P. Kulkarni, S. Kabadi, P. S. Aditya, and R. Ujawane, "Vision based Roughness Average Value Detection using YOLOv5 and EasyOCR," in *2023 IEEE 8th International Conference for Convergence in Technology (I2CT)*, pp. 1–7, 2023, doi: 10.1109/I2CT57861.2023.10126305.

[86] G. Septian, D. Wahiddin, H. Y. Novita, H. H. Handayani, A. R. Juwita, and A. F. Nur Masruriyah, "The Implementation of Real-ESRGAN as An Anticipation to Reduce CER Value in Plate Number Extraction Results Employing EasyOCR," in *2022 Seventh International Conference on Informatics and Computing (ICIC)*, pp. 1–5, 2022, doi: 10.1109/ICIC56845.2022.10006900.

[87] E. Mythili, S. Vanithamani, R. Kanna P, R. G, K. Gayathri, and R. Harsha, "AMLPDS: An Automatic Multi-Regional License Plate Detection System based on EasyOCR and CNN Algorithm," in *2023 2nd International Conference on Edge Computing and Applications (ICECAA)*, pp. 667–673, 2023, doi: 10.1109/ICECAA58104.2023.10212354.