Computer Engineering and Applications Vol. 15, No. 1, February 2026

Implementation of Color Matching in Ball Image Processing Using OpenCV

Sirmayanti[1*], Awah Rizqi Tanzil Haq Azami[2]

[1,2] *Department of Electrical Engineering, Politeknik Negeri Ujung Pandang*
** sirmayanti.sirmayanti@poliupg.ac.id*

ABSTRACT

This research focuses on object detection through color matching, enabling machines to detect objects based on specific colors using OpenCV, a Python library widely used in computer vision projects. As a form of Machine Learning and Artificial Intelligence, this method allows machines to automatically learn and classify objects by simulating the human visual system. The study aims to enable a machine to detect and locate a ball through digital image processing using a webcam. The research method includes digital image processing, implementation on a Raspberry Pi, and testing on a robot, where logic is applied to guide the robot toward the ball by detecting its color. The outcome is an object detection system that identifies the ball's position in two dimensions based on its specific color. In this case, the RGB code (164, 122, 0) and a minimum ball size of 10 radians were successfully implemented on the robot. However, the system has limitations under certain conditions. Future improvements will involve integrating TensorFlow for dataset processing and OpenCV for real-time object detection to achieve more accurate results.

Keywords: Artificial Intelligence, Color Matching, Computer Vision, OpenCV, Digital Image Processing.

1. INTRODUCTION

Research in Artificial Intelligence (AI), including image processing, language recognition, and robotics, often leverages two key subfields: Machine Learning (ML) and computer vision. ML involves developing algorithms that can automatically learn and solve problems based on input data and various techniques to aid the learning process [1], [2]. This system is widely used in fields like logistics and construction [3]. ML and computer vision are frequently combined to enable machines to perform real-time analysis. A common example is self-driving cars, where computer vision gathers data through cameras, while ML processes information related to safe driving.

This concept is also gaining traction in Indonesia, particularly in the Indonesian Robot Contest (KRSBI), which incorporates computer vision and ML, using a webcam as a visualization tool for digital image processing, similar to the human eye. Additionally, this technology is frequently applied in the healthcare sector for tasks such as detecting diabetic wounds [4], object counting [5], and even human detection in security systems [6].

Computer vision is a technology that can assist human work across various domains. Consequently, this research focuses on developing a system for detecting an

orange ball using the color matching method as an application of this technology. According to Utama & Riki (2017), object detection can be implemented by recognizing the colors and patterns of the target [7]. The most commonly used color parameters are red, green, and blue (RGB) [8]. In fact, utilizing both color and shape parameters can yield more accurate results and is particularly effective for tracking an orange ball on a green field [9], [10].

2. MATERIAL AND METHODS

2.1 ARTIFICIAL INTELLIGENCE, MACHINE LEARNING & COMPUTER VISION

Intelligence refers to the human ability to acquire knowledge about objects and apply it in real-world situations [11]. In the field of computer science, AI enables machines to imitate the functions of the human brain, allowing them to act, think, and make decisions like humans [12], [13]. For real-time object detection, a relevant technology is computer vision. Computer vision is a subfield of AI and ML that utilizes specialized methods and algorithms without relying on large datasets or complex ML algorithms. The relationship between these concepts is illustrated in Figure 1 [14].

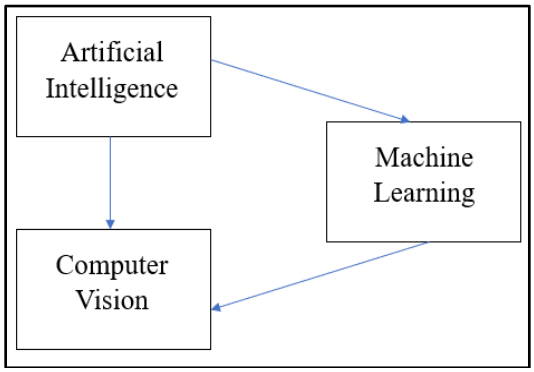


FIGURE 1. Relationship between AI, ML, and Computer Vision

As beings endowed with superior intelligence, humans can recognize and classify objects, enabling them to detect differences between one object and another. ML present the human visual system serves as a model for machines to imitate [6]. While computer vision doesn't explicitly involve training data, it focuses on understanding object features. Techniques such as image processing, color segmentation, contour detection, and tracking the movement of objects (like a ball) based on previous positions are methods used to analyze basic ML principles. These methods can still be linked to the ML subfield focused on analyzing and understanding object features in image or video data.

2.2 IMAGE PROCESSING

The technique of processing and processing images by importing photos or webcam captures is known as digital image processing [15], [16]. According to Umam and Negara (2016), image capture is defined as when there is a reflection of light beams that illuminate an object caught by the eye or other optical devices, so that an image in the form of an object shadow is recorded [6]. In a color image, each pixel consists of RGB colors that have a range of 0-255, so the total is 2553 or equal to

ISSN: 2252-4274 (Print)
ISSN: 2252-5459 (Online)

16581374 different variations in the image. Furthermore, the color will be converted into hue, saturation, value (HSV) to produce a binary image. A binary image is a color model consisting of black and white that becomes the output image so that the robot can recognize objects that have been determined [15]. After the object is recognized, morphological filtering is applied to fill the gaps in the object and smooth the object image, this method is also referred to as dilation and erosion [6].

2.3 RED GREEN BLUE (RGB)

TABLE 1.
Intensity Range of RGB [15]

Color	Color Intensity		
	Red	Green	Blue
Green	0-173	100-255	0-170
Blue	0-240	0-248	112-255
Red	128-255	0-160	0-128
Yellow	102-255	102-255	0-50
Magenta	75-255	0-230	128-255
Cyan	0-244	128-255	20-255

The RGB concept is a color model that adds strong primary light, which are red, green, and blue. In other words, if a room is given red light, then objects in the room will turn red. Likewise, if you add another primary color, for example adding green, it will produce a blue color which is a combination of green and red [12]. The RGB intensity range of primary and secondary colors is shown in Table 1.

2.4 HUE, SATURATION, VALUE (HSV)

Kartika and Rochmatika (2021) stated that HSV images are composed of hue, saturation, and value [17]. Hue represents colors that correspond to different wavelengths of light and can be perceived by the human eye, which is capable of detecting RGB. Saturation indicates how much white light is mixed with the hue and ranges from 0 to 100%. Value, or brightness, refers to the level of light perceived by the eye [18], [19]. Essentially, RGB is linked to color luminance, while HSV is used to separate image luminance from color information to adjust color intensity [20] without altering other color components [4]. Prior to converting RGB to HSV, color normalization is necessary to obtain values between 0 and 1, as indicated in the equation as follows.

$$r = \frac{R}{R+G+B}, g = \frac{G}{R+G+B}, b = \frac{B}{R+G+B} \quad (1)$$

$$\text{so} : r + g + b = 1 \quad (2)$$

Furthermore, the mathematical equation of the HSV process can be written as follows [14]:

$$V = max\ (r,g,b) \quad (3)$$

$$S = \begin{cases} 0, V = 0 \\ 1 - \frac{min\,(r,g,b)}{V}, V > 0 \end{cases} \quad (4)$$

$$H = \begin{cases} \frac{60 \times (g-b)}{S \times V}, jika\ V = r \\ 60 \times \left[2 + \frac{b-r}{S \times V}\right], jika\ V = g \\ 60 \times \left[4 + \frac{r-g}{S \times V}\right], jika\ V = b \end{cases} \quad (5)$$

2.5 METHODS

This research employs the color matching method, utilizing the Python programming language in conjunction with the OpenCV library. The materials used include a Raspberry Pi microcomputer equipped with a microSD card, a webcam, and a robot to test the effectiveness of this research. The research process is illustrated in Figure 2.

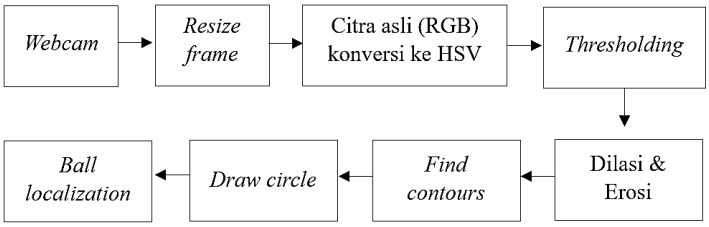


FIGURE 2. Block diagram of research method

The first step of this research involves programming the image processing on a laptop before implementing it on the Raspberry Pi. This is due to the laptop's superior reading and processing speed. The research block diagram, shown in Figure 1, illustrates that the image captured by the webcam is transmitted and displayed in a smaller size to enhance the data transmission process. Subsequently, the image is converted into an HSV model, serving as a threshold value to facilitate the color segmentation process.

According to a previous experiment by Umam (2016), dilation and erosion techniques are employed to achieve a more solid segmentation of the object (ball), allowing for the identification of the edge lines (contours) of the object. Additionally, based on the ball's data, a bounding shape in the form of a circle is required as a reference to confirm that the ball has been detected, as well as to provide positional reference data for the ball [6].

If the ball has been detected properly, the next step is to apply the program to the Raspberry Pi by first configuring it so that the microcomputer can be controlled remotely by a laptop using virtual network computing (VNC) Viewer. VNC is a client application that can be used to access or monitor a computer remotely, such as executing or sending files from one computer (remote) as well as executing or sending files from one computer to another to another computer [21].

Research conducted in 2021 indicated that using PWM allows for easy control of the average power delivered to a load, enabling adjustment to the desired speed of the robot [22]. To prove that the program works well, the experiment was carried out on a robot with logic and activated PWM pin as the speed controller of the robot's driver, the robot will go to the position of the ball as the final stage of the research.

ISSN: 2252-4274 (Print)
ISSN: 2252-5459 (Online)

3. RESULT AND DICUSSIONS

3.1 DIGITAL IMAGE PROCESSING TESTING

In this experiment, the RGB code utilized is (164, 122, 0), representing one of the orange color codes identified in the tracking results. Mathematically, prior to converting RGB to HSV, color normalization is conducted to obtain values ranging from 0 to 1, as shown in equation (1).

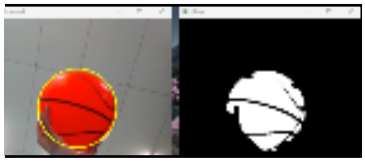
$$
\begin{aligned}
r &= \frac{R}{R+G+B} \\
&= \frac{164}{164+122+0} \\
&= 0{,}5734265734265734 \\
g &= \frac{G}{R+G+B} \\
&= \frac{122}{286} \\
&= 0{,}4265734265734266 \\
b &= \frac{B}{R+G+B} \\
&= \frac{0}{164+122+0} \\
&= 0
\end{aligned}
$$

Theoretically, the output value generated by the HSV method is a binary image consisting solely of 1s and 0s, essentially representing black and white. These results are derived from the HSV image processing, which is compared against the color components of each pixel in the RGB image to determine whether a color is acceptable. To achieve the HSV conversion value, equations (3), (4), and (5) are utilized, with the previously obtained RGB normalization as input.

$$
\begin{aligned}
V &= max\ (r,g,b) \\
&= 0{,}5734265734265734 \\
S &= 1 - \frac{min\ (r,g,b)}{V} \\
&= 1 - \frac{0}{0{,}5734265734265734} \\
&= 1 \\
H &= 60 \times \frac{(g-b)}{S \times V} \\
&= 60 \times \frac{(0{,}4265734265734266 - 0)}{1 \times 0{,}5734265734265734} \\
&= 44{,}634^{\circ}
\end{aligned}
$$

The value mentioned above serves as the lowest threshold value. Consequently, if the object falls outside this range, it will appear black with a value of 0, while it will be white if it is within the threshold range, as shown in Table 2(a).

TABLE 2.
Digital image processing in OpenCV

No	Step	Figure
a)	*Thresholding*	
b)	*Erode and Dilate*	
c)	*Draw Circle*	

The morphological filtering results are shown in Table 2(b). The table illustrates that, when comparing the images before and after the method is applied, the white dots that were previously scattered in the thresholding table become more organized and concentrated on the ball object after applying the method. This improvement occurs because erosion is utilized to eliminate small white noise from the image, while dilation serves to enlarge the objects. As more iterations are executed, additional noise is removed.

One limitation of the color-matching method is its sensitivity to lighting conditions [5]. The amount of light hitting the ball varies on different sides, depending on the direction of the light source. To address this issue, we use the cv2.findContours() function to identify the edges of the ball shape based on the previous steps and employ cv2.circle() to draw a circle around the detected ball object. As demonstrated in the masking window shown in Table 2(c), even though a hand partially obscures one side of the ball, the ball is still accurately detected.

3.2 BALL DETECTION TESTING

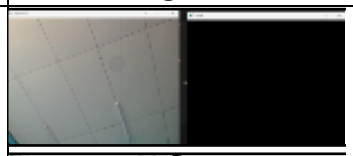
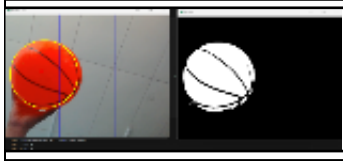
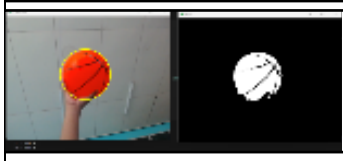
In this experiment, to aid the robot's logic, the frame is divided into three sections, allowing the robot to move toward the ball based on its position within the frame. The window for displaying objects is set to a size of 600 px horizontally and 450 px vertically, so the following calculation is performed to achieve the correct frame division.

$$\boldsymbol{Frame} = \frac{\boldsymbol{Object\ display\ window\ width}}{\mathbf{3}} = \frac{600\ px}{3} = 200\ px \tag{6}$$

The image illustrating the execution of the calculation can be found in Table 3. The blue vertical line shown in Table 3 represents the outcome of the previously performed frame division, where the first line is positioned 200 px horizontally to the right of point 0, and the second line is located 200 px to the right of the first line.

ISSN: 2252-4274 (Print)
ISSN: 2252-5459 (Online)

TABLE 3.
Ball Position Testing

No	Figure	Information
a)		The ball isn't detected
b)		The ball is on the left side of frame
c)		The ball is at the center of frame
d)		The ball is on the right side of frame

According to the data in Table 3(a), there are no indications that the camera is detecting the object, neither in the original frame on the left nor in the masking frame on the right. In Table 3(b), the ball object is located in the left frame, yielding output values of x and y at (41.5) px and (217.5) px, respectively. For the robot's movement reference, the x value, which represents the horizontal position, is particularly important. The obtained data shows that the ball's position in the frame aligns with the previously established range for the left frame, which is from 0 to 200 px.

Additionally, in Table 3(c), the ball object is positioned at the center of the frame, resulting in output coordinates of x and y at (290.5) px and (233.0) px, respectively. This aligns with the logic established in the programming, where the x coordinate for the center frame is set to range from greater than 200 px to 400 px. Table 3(d) illustrates the position of the ball in the right frame, with horizontal coordinates of (472, 1534765625) px and vertical coordinates of (270, 4808654785156) px.

3.3 BALL DETECTION DISTANCE TESTING

In this experiment, the data in Table 4 was analyzed to determine the maximum distance at which the ball can be detected. In the first entry of Table 4, the ball is positioned at the zero point because being too close to the camera results in a dark image, making the ball unreadable as it's obscured by its own body. At this position, the camera detects the ball with a maximum size of 336.32 px in radians. The ball's position is then incrementally moved until it is 25 cm closer to the camera. As shown in the second entry of Table 4, the ball is still detectable with a radius measurement of 135.92 px.

TABLE 4.
Ball detection distance testing

Figure	Distance to webcam (cm)	Radius (px)	Status
	0	336,32	Detected
	50	72,85	Detected
	150	29,17	Detected
	200	22,20	Detected
	265	<9	Not Detected

To further assess the accuracy of the readings, the distance from the ball to the camera was increased to 150 cm, resulting in a radius value of 42.70 px. In the final experiment, the distance from the ball to the camera was set at 230 cm, or 2 meters and 30 cm, due to space constraints. Under these conditions, the ball was still detected, yielding a radius value of 18.85 px. The results of the experiments indicate that the ball object can be detected because the programming script includes logic that enables detection of the ball when its radius value exceeds 10 radians.

3.4 IMPLEMENTATION TESTING ON ROBOT

In this test, the robot's speed is modified by adjusting the reading delay on the microSD card, and the experimental data is presented in Table 5.

TABLE 5.
Implementation testing on robot

Figure	Information
76%	Failed
25%	Failed
20%	Failed

ISSN: 2252-4274 (Print)
ISSN: 2252-5459 (Online)

15%	Failed
13%	Succeed

The test experiment conducted at 76% speed was unsuccessful. This failure was attributed to the processing speed of the microSD card, which was unable to keep up with the driver's speed. In the subsequent experiment, the velocity was reduced to 25%; however, the robot's speed still appeared too high, preventing it from effectively adjusting the motor driver's speed.

In the previous experiment, it was found that the robot needed a further speed reduction, resulting in a 20% speed setting. Despite this, the microSD card still couldn't synchronize properly with the robot's driver. A PWM value of 15% was then tested with an additional 5% reduction, and it was observed that while the robot could occasionally track the rotating ball, a slight speed reduction was still necessary. In the final experiment, with a PWM value of 13%, the robot moved at a relatively slow speed but was still able to successfully follow the ball.

4. CONCLUSION

Object detection can be achieved through the color matching method, which has been configured for the orange color range, using OpenCV as an additional library within the Python programming language. The creation of contours and the identification of center points on objects help accurately determine the position of the ball in two dimensions.

REFERENCES

[1] A. Ahmad, "Mengenal Artificial Intelligence, Machine Learning, Neural Network, dan Deep Learning," *Journal Teknologi. Indonesia., No. October*, 2017.

[2] M. A. Ponti, L. S. F. Ribeiro, T. S. Nazare, T. Bui, and J. Collomosse, "Everything you wanted to know about Deep Learning for Computer Vision but were afraid to ask," In *IEEE 30th SIBGRAPI conference on graphics, patterns and images tutorials (SIBGRAPI-T)*, pp. 17–41, 2017.

[3] Weka, "Computer Vision vs. Machine Learning | How Do They Relate?", [Online], https://www.weka.io/learn/ai-ml/computer-vision-vs-machine-learning/ (Accessed: Aug. 14, 2023).

[4] W. F. Roshandri, E. Utami, and A. B. Prasetio, "Diabetic Wound Segmentation Using Masking Contour Image Processing," *SISFOTENIKA*, vol. 11, pp. 111–123, 2021.

[5] D. A. Prabowo, D. Abdullah, and A. Manik, "Object Detection and Calculation Based on Color Using Color Object Tracking," *Pseudocode*. Vol. 5, No. 2, pp. 85–91. DOI:https://doi.org/10.33369/pseudocode.5.2.85-91, Sep 2018.

[6] K. Umam and B. S. Negara, "Human Object Detection on Video Database Using Background Subtraction Method and Morphological Operation," *Core*

IT. Vol. 2, No. 2, pp. 31–40. DOI:http://dx.doi.org/10.24014/coreit.v2i2.2391, Dec 2016.

[7] J. Utama and D. Riki, "Implementation of Target Detection System Based on Color and Pattern Recognition for Ball Follower Robot," *Telekontran.* Vol. *5,* No. 2, pp. 107–117, Oct 2017.

[8] Y. Yudha, D. Ardhiyanta, L. Haris, D. Anastasia, and R. Widiarti, "Basic Color Image Recognition Application," *Widya Teknik*. Vol. 15, No. 1, pp. 54–57, 2016.

[9] M. R. Kurniawan, "Implementation of Object Tracking Based on Color Filtering on Pixy CMUcam 5 Camera Sensor," *Fokus Elektroda*. Vol. 3, No. 3, pp 1–4, DOI:http://dx.doi.org/10.33772/jfe.v3i3.6586, Aug 2018.

[10] M. I. Bustami, A. Siswanto, N. Toscany, M. F. Ramdhani, and C. Saputra, "Detecting Ball Shape and Color on Humanoid Soccer Robot Using Raspberry Pi," *STIKOM Dinamika Bangsa Jambi*, 2014.

[11] V. Amrizal and Q. Aini, *Artificial Intelligence*, Jakarta:Halaman Moeka Publishing, 2013.

[12] J. Jenis, J. Ondriga, S. Hrcek, F. Brumercik, M. Cuchor, and E. Sadovsky, "Engineering Applications of Artificial Intelligence in Mechanical Design and Optimization," *Machines*, vol. 11, No. 6, pp. 577–616, DOI:https://doi.org/10.3390/machines11060577, May 2023.

[13] L. Rahadiantino, A. Fahmi, H. Wirawasista Aparamarta, S. Kustanti Moerad, and A. Mazharuddin Shiddiqi, "Implementation of Artificial Intelligence Learning for Elementary School Students in Batu City, Malang, East Java," *Journal of Innovation in Primary School Education and Learning*, Vol. 6, No. 1, pp. 92–101, July 2022.

[14] F. R. R. Geri, "Design of a Learning Material Delivery Recommendation Application Based on Facial Emotion Detection," Komputer Indonesia University, Bandung, Indonesia, 2020.

[15] R. D. Kusumanto and A. N. Tompunu, "Digital Image Processing for Object Detection Using RGB Normalized Color Model," in *National Seminar on Applied Information & Communication Technology*, 2011.

[16] F. Jalled and I. Voronkov, "Object Detection using Image Processing," pp. 1–6, DOI:10.48550/arXiv.1611.07791, Nov 2016.

[17] V. S. Kartika and R. A. Rochmatika, "Color Object Tracking to Control 2 Axis Servo with HSV Filtering Method," *Journal of Applied Electrical Engineering (JTET)*, Vol. 10, No. 1, pp. 20–26, DOI: http://dx.doi.org/10.32497/jtet.v10i1.2554, Apr 2021.

[18] S. Mutrofin and E. Kurniawan, "Color Feature Extraction Using Hue Saturation Value for Plant Classification Based on Leaf Image," DOI:10.13140/RG.2.2.24846.66884, April 2015.

[19] A. Zarkasi, H. Ubaya, and M. Fajar, "Implementation Color Filtering and Harris Corner Method on Pattern Recognition System for Underwater Objects," *Computer Engineering and Applications*, Vol. 6, No. 3, pp. 139–144, 2017.

ISSN: 2252-4274 (Print)
ISSN: 2252-5459 (Online)

[20] O. M. Gazal, A. E. Adedokun, I. J. Umoh, and O. Momoh, "LWT-CLAHE Based Color Image Enhancement Technique: An Improved Design," *Computer Engineering and Applications*, Vol. 9, No. 2, pp.117– 126, June 2020.
[21] RealVNC, *VNC Connect Security Whitepaper*, 1.4 ed. 2021.
[22] F. B. Setiawan, O. J. Aldo Wijaya, L. H. Pratomo, and S. Riyadi, "Computer Vision Based Automated Guided Vehicle Navigation System and Implementation on Raspberry Pi," *Rekayasa Elektrika*, Vol. 17, No. 1, pp. 7–14, Mar 2021.