An Automated Realtime Image Driven Quality Control : A vision based Solution for Industrial Object Classification

1st Nagesh A. Patil
Principal Product Manager
Veritas Software.
nagesh.patil@veritas.com

2nd Nutan V. Bansode
Dept. Of E&TC
MIT Academy of Engineering
Pune, India
nvbansode@mitaoe.ac.in

3rd Sonali Sawant
Dept. Of E&TC
PCCOE
Pune, India
sonali.sawant@pccoepune.org

4th Sumit Kumar
Dept. Of E&TC
MIT Academy of Engineering
Pune, India
202301070208@mitaoe.ac.in

5th Aryan Kumar
Dept. Of E&TC
MIT Academy of Engineering
Pune, India
202301070164@mitaoe.ac.in

6th Amir Furquani
Dept. Of E&TC
MIT Academy of Engineering
Pune, India
202301070165@mitaoe.ac.in

7th Umesh V. Kale
Chief Technology Officer
Concept Systems India Pvt Ltd.
Wakad, Pune,
411057,Maharashtra,India

Abstract— **Maintaining a high standard of quality throughout a product's lifespan is an objective for all manufacturing businesses. It covers every step of the process, from sourcing raw materials to final delivery, including packing, installation, and after-sale support. In industrial environments, the capability of recognizing and classifying objects automatically according to their physical attributes is crucial for automating procedures like sorting, inspection, and quality control. The proposed concept enables industry to identify and report defects at the appropriate time, allowing firms to implement corrective measures effectively. The concept employs specialized technology to minimize human participation and provide optimal quality standards. 24/7. This research introduces a software-based method for detecting the shape, size, and color of objects in images with the aim of offering a contactless and cost-effective solution to different industrial processes. The system processes visual features by eliminating the background, object boundary detection, image scale-based estimation of dimensions, and determination of object color from average pixel values with 94.34% accuracy. This article identifies the proposed algorithm as the optimal approach for developing a Flask application, which offers a cohesive platform for recognizing the shape, size, and color of objects. It processes the data purely based on images without any hardware installation, making it perfect for use in situations where accuracy, effectiveness, and flexibility are paramount. The method can be used in real-time or offline processing and has potential to be incorporated into larger industrial automation processes.**

Keywords— *Shape, Size, Color Detection, Image Processing, OpenCV, Dimension Measurement*

I. INTRODUCTION

Every manufacturing business aim for a high-quality bar for the entire lifecycle of the product. It includes everything from raw material selection, production, packaging, and transportation to installation and after-sales services.

The implementation of strict quality metrics through various standards, such as Six Sigma and ISO, has made it more difficult for companies to achieve the objective of controlling cost elements. Most industries use human resources for quality checks in every stage of manufacturing as an alternative to expensive inspection systems. Errors are human. Many defects go unnoticed, and those cost the industries a lot. Imagine the loss if a product is produced and shipped thousands of miles in the wrong size or color or altogether a incorrect product.

Over the past few years, the need for intelligent systems that can detect and analyze physical features of things shape, size, and color has witnessed a strong surge in various sectors like manufacturing, logistics and healthcare. Real time analysis of objects facilitates automation in key applications such as sorting, inspection, and quality control, which in turn improves overall efficiency and minimizes dependency on manual tasks.

The proposed idea discussed here helps industries to detect and notify the defect at the right stage so that the companies can take corrective actions to mitigate those at the right stage.

The idea uses the niche technologies to reduce the human intervention and to give the best quality standards 24x7.

Unlike conventional methods based on reference objects for measuring size, this system uses intrinsic camera parameters like focal length, sensor size, and object distance to convert pixel sizes into real measurements directly. Along with this, color detection is achieved by sampling average pixel values within object regions and comparing them against a pre-existing dataset of typical colors. By combining the two skills under one image processing pipeline, there is a capability to examine objects as a whole based on an image alone, without the need for costly equipment or complicated setups. This paper is an implementation system in Python that employs libraries such as OpenCV and NumPy to inspect static images for object shape recognition, size measurement, and color classification. The system employs background subtraction, contour approximation, and morphological processing to separate objects from the background and extract their features, hence being applicable for real-world use involving low-cost, contactless, and efficient inspection**.**

II. Related Work

For this study, a comprehensive literature review was conducted, encompassing 30 research papers focusing on using various algorithms and systems for object detection with shape , size and color prediction.

Object detection and measurement using computer vision have seen significant advancements across various domains such as automation, education, robotics, and surveillance. Most methods focus on detecting specific object attributes color, shape, or size using classical image processing or deep learning approaches

In this literature survey, considered three Approaches:

1) *Color Detection*

Color detection systems based on the RGB model were proposed by Ranjini Arali et al. [1] and Deborah T. Joy et al. [2], utilizing OpenCV and GUI elements for user interaction. These systems demonstrated real-time performance using mouse-based input and text-to-speech for feedback. More complex techniques such as genetic algorithm-based color segmentation across multiple color spaces (RGB, HSV, YIQ, etc.) were implemented by Yacine Messai et al. [7] to improve classification accuracy and robustness. Saliency-based approaches were explored by Guang-Hai Liu and Jing-Yu Yang [12], leveraging the Lab color space and SLIC segmentation for precise foreground-background distinction. Additionally, Mohamed Abdellatif [13] introduced spectral sharpening in pre-processing, improving object visibility under varied lighting conditions.

2) ***Shape and Size Detection***

Shape and size recognition using contour analysis and Euclidean distance was discussed by Madhavi Karanam et al. [14] and Dr. M. Mahesh et al. [15], with OpenCV being central to real-time measurement systems. Nashwan Adnan Othman et al. [8] used Canny edge detection and morphological transformations to extract and measure object contours. Unique approaches such as tactile sensing by Shreyasi Datta et al. [18] used MEMS-based pressure images to classify object shapes and sizes with machine learning classifiers. Zhang and Li [9] incorporated double thresholding and HSI color space for improved object segmentation under challenging lighting.

3) ***Hybrid and Deep Learning-Based Systems***

YOLOv5-based frameworks were presented by Azra Nasreen et al. [16] and Basavaraj M. U. et al. [19], applying object detection with bounding box normalization for size estimation from video streams. Edge-Color Distribution Transform (ECDS) proposed by Song et al. [10] tackled overlapping object detection by mapping edge points into 3D color spaces. Meanwhile, systems like those of Wicaksono et al. [4] and Pejlekar et al. [17] integrated Raspberry Pi and marker-based tracking for educational applications.

While significant research has been conducted in the field of object detection using image processing and computer vision, notable gaps remain: Unified Detection Approach, High Resource Demand of Deep Learning Models, Limited Deployment Flexibility, Inconsistent Performance in Real World Conditions. This paper proposes an optimal approach for developing a Flask-based web application that provides a cohesive platform for recognizing the shape, size, and color of objects using only image inputs. The proposed system is designed for flexibility, accuracy, efficiency and scalability. This system addresses key limitations of previous research by offering a hardware-independent**,** fully integrated**,** and deployable solution, ideal for industrial inspections, educational tools, and general-purpose object analysis.

III. Methodology

The dataset is collected from a controlled industrial environment containing various objects of different shapes of different objects, colors, and sizes. We categorize the dataset based on physical features including shape (e.g., circle, rectangle), color (e.g., light brown, blue, gray), and size (in cm). A systematic, organized approach based on the literature and build a reliable OpenCV-based detection model.

1. Pre-processing data for image enhancement using OpenCV: Background removal, grayscale conversion, Gaussian blurring, thresholding, and Canny edge detection are applied to enhance image quality and isolate the object region.
2. Contour detection and shape analysis: Using OpenCV’s find Contours() and geometric feature analysis (e.g., contour approximation and circularity metrics), the system identifies object shapes.
3. Color detection using HSV analysis: The object region is masked using contours. The image is converted to HSV color space, and the average HSV value within the contour is used for color classification.
4. Size estimation using pixel-to-mm calibration: Dimensions such as radius or bounding box size

are measured in pixels and converted to centimeters using a predefined scale factor.

5. Create a complete image processing pipeline using OpenCV: Integrate all steps into a robust pipeline using Python and OpenCV, capable of processing batches of images in real-time or near-real-time.
6. Simulate and test the pipeline on sample datasets: Test the performance of the model on new images to verify detection accuracy for shape, color, and size.
7. Verify and optimize detection results to improve accuracy: Fine-tune thresholding, contour filtering, and color classification parameters to enhance detection robustness and reduce false positives.

Figure 1 shows a graphical representation of the proposed system. This research explores the nature of Object detection' using different datasets with the assistance of a Open CV model. This paper proposes a Flask-based application as an optimal solution for developing a unified platform that recognizes the shape, size, and color of objects. The prototype is implemented using OpenCV, which acts as a powerful engine for image processing and accurate prediction of object characteristics.

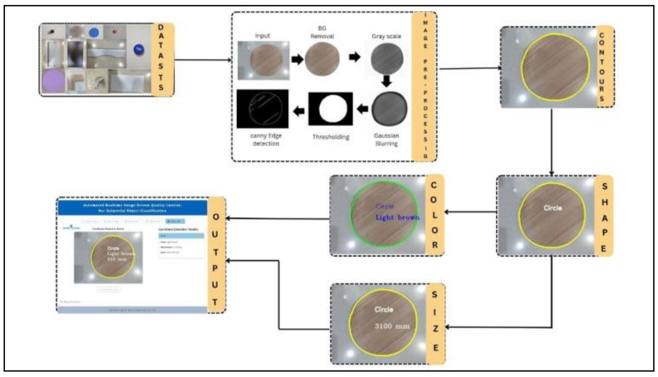


Fig 1. Graphical Abstract of Proposed System

IV. Proposed work

Fig2 shows flow chart of a system to recognize the shape, color, and true size of an object from the input image by computer vision algorithms. The image is first loaded and then the background removal is performed, which separates the object from the scene and improves accuracy for subsequent processing. Grayscale conversion simplifies the image to single-color data, and blurring with a Gaussian filter eliminates noise and smoothes the image to avoid spurious edge detection. Thresholding later transforms the image to binary mode, highlighting object edges for accurate contour detection. All these preprocessing methods together enhance object visibility to allow precise contour detection to be used for shape recognition. Then, the system computes the mean color of the detected contour and cross-checks it with a pre learned database of colors to classify colors. For size estimation, pixel sizes of the image are converted into real sizes using known camera parameters independent of a reference object. Lastly, the image is labeled with the recognized shape, color, and size, and it provides an inclusive and functional solution for application in automated inspection, object classification, and real-time examination in industrial environments.

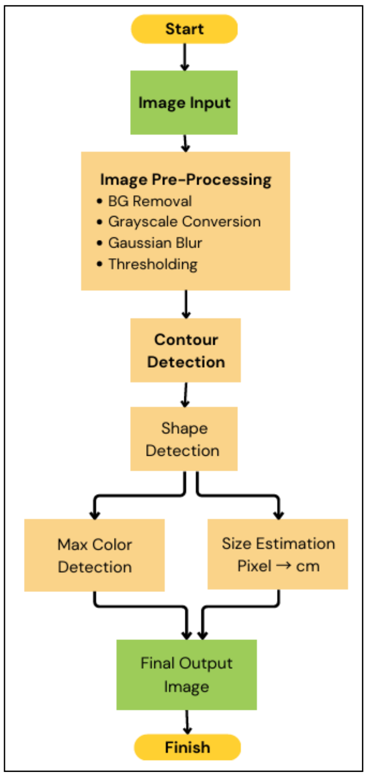


Fig. 2. Flow Chart

A. *Image Input*

The process starts by capturing or uploading an image containing various objects of different shapes, sizes, and colors.

B. *Image Preprocessing*

Image preprocessing is an important process that conditions the input image to be ready for appropriate shape, color, and size detection. It starts with resizing the image to the standard scale (in this case, reduced to 30% of the original size), thus minimizing computational complexity between various resolutions of the image. Background removal is done by utilizing the rembg library, which isolates foreground objects from irrelevant background elements. This helps to reduce noise and concentrate on the real objects of interest.

Once the background has been removed, the image is then converted to a grayscale, down-sampling data by discarding color information and transforming the image to intensity values. This aids in easier processing of subsequent operations like thresholding. The grayscale image is then processed with the application of Gaussian blur to eliminate noise and make the image smooth. Lastly, thresholding (with Otsu's algorithm) is done to transform the grayscale image into a binary image in which objects are properly separated from the background. The resulting binary image is crucial for proper contour detection since it specifies the object contours with high contrast. Overall, the preprocessing pipeline improves the image quality and clarity, allowing more accurate shape analysis, color separation, and size measurement in the subsequent steps**.**

C. *Contours Detection*

Contour detection is an important process employed to derive the shape boundary of objects from the binary image obtained after pre-processing. Contour detection helps the system determine separate regions depicting possible objects of interest. Contour detection, by employing OpenCV's findContours() method, follows the boundary of white areas in the binary image and provides a list of contours—each as a collection of points defining the object boundary. These contours are subsequently filtered by area in order to eliminate noise or redundant small areas. The contours thereby identified serve as the basis for subsequent operations such as shape classification, color analysis, and estimation of size. The primary library used for the same is OpenCV, which provides effective contour detection, drawing, and manipulation capabilities through tools such as cv2.findContours() and cv2.drawContours().

D. *Shape Detection*

Shape Detection is the Process of determining the geometric shape of objects that have been detected within the image from their contours. Once the contours are obtained, shape detection is performed by approximating each of the contours to a polygon using OpenCV's **cv2.approxPolyDP()** function. This process diminishes the points in a contour to its most basic polygonal representation so that it can be classified more easily based on the number of sides. For example, if the approximated polygon is three-sided, it is a Triangle; four-sided would mean a Rectangle (or Square); and five or greater sides would tend to mean a Circle. Such reasoning is used to correctly classify simple geometric shapes. OpenCV is the primary library used in doing this, namely functions like cv2.approxPolyDP() for polygon approximation and cv2.arcLength() for contour perimeter. These operations together enable efficient detection and classification of shapes based on contour properties.

E. *Size estimation*

Size Estimation is the calculation of the real size of objects in the world based on their size in pixels from the image. Once the object's contour is detected, the code employs OpenCV's cv2.minAreaRect() to draw the minimum area rectangle around the object and obtain its angle and size in pixels. The corner points of the rectangle are then determined via cv2.boxPoints() and shifted to accommodate any resizing which was previously completed in preprocessing. The side lengths (in pixels) are obtained via the math.hypot() function to determine the distance between two points in a straight line. These pixel lengths are then translated to real-world dimensions (centimeters) via a pinhole camera formula based on known quantities such as focal length, sensor width, and camera-object distance. The conversion is accomplished via a custom function named **calculate_real_size()**. The primary libraries employed in this step are OpenCV (for box and contour detection) and math (for distance calculation), which collectively assist in estimating the real size of the objects with precision.

F. *Average Color detection*

Color Detection is the process of determining the dominantcolor of detected object. Once the contour of the object is discovered, a mask is generated through OpenCV to extract only the region within the contour. The mask is then applied to the original image through cv2.bitwise_and() such that only the pixels of the object are chosen. Next, all the pixel values in the masked area are averaged using NumPy to get the mean color. The resulting RGB value is then matched with the nearest color name by comparing it with pre-defined colors from an Excel file using a color distance formula. This helps in assigning to every object's color a human-readable name (i.e., "Red" or "Blue"). The major libraries employed here are OpenCV (for contouring and masking functions), NumPy (for determining the average color), and Pandas (for reading the color values from the Excel file). In this manner, every object will be labeled with a proper and accurate color label.

G. *Software Requirements*

1. Open CV platform
2. Programming language-:Python
3. Web application framework-:Flask
4. IDE-:Visual Studio Code
5. Operating System-:Windows

H. *Web Page design using Flask and React:*

1. Webpage architecture:- This application is a client-server program with a specific Backend and Frontend directory , the back-end processing performs the image processing while the Frontend gives a graphical user interface from which the images are uploaded and the shape detection result is displayed
2. Back-end tech stack: we utilized the backend with the flask version 201 a lightweight Python web Framework it offers restful API endpoints used for uploading an image shape detection, size measurement and color analysis the core image processing functionality is built heavily upon OpenCv,
   - numpy used for speedy math operations on matrix images
   - pillow generally used for basic image processing tasks
   - flask-cors it helps in handling cross-origin requests from the frontend
   - werkzeug used for secure file uploads

- rembg to extract backgrounds from images uploaded to get the accurate size of the image

3. Frontend technology stack: Frontend uses react with bootstrap for responsive design, the dependencies required
    - React and ReactDOM employed for building the component based UI.
    - react bootstrap for responsive design and styling
    - Axios for http requests to the backend API

V. RESULT

I.Part A: Software Implementation using OPENCV Model

1. Shape Detection Techniques:
- Edge Detection: e.g., using the Canny edge detector.
- Contour Detection: Using OpenCV's findContours() function.
- Hough Transform: For detecting circles and lines.

2. Shape Identification:
   Shapes are classified based on geometric features like number of edges, aspect ratio, circularity, etc.
   For circle detection:

$$circularity = 4\pi * \frac{Area}{perimete^2} \qquad (1)$$

Aspect Ratio for rectangles and squares detection:

$$Aspect\ Ratio = \frac{width}{hight} \qquad (2)$$

$$Aspect\ Rati \approx 1 \rightarrow square$$
$$Aspect\ Rati \neq 1 \rightarrow rectangle$$

Table1: Objecte detection using parametrs Shape,Size & color

Sl. No	Object name and shape	Size AD (cm^2)	Size PD(cm^2)	Color	Accuracy (%)
1	Wooden Rectangle 1	60x30	59.09x28.87	Dim gray	94.85
2	Wooden Circle	33.5	31	Light brown	92.60
3	Wooden Rectangle 2	60x30	61.37x31.22	Alice blue	93.54
4	Wooden Rectangle 3	65x25	63.54x23.40	Alice blue	91.52
5	Wooden Square	28x28	28.58x29.31	Light brown	93.14
6	Perfume Cap	4.5	4.66	Black	96.57
7	Bottle Cap	4.5	4.75	Snow	94.61
8	Breadboard	16.5x5	16.93x5.19	Mid cream	93.49
9	Nivea Cream	7	6.80	Indigo	97.27
10	Boroline	5.5	5.23	Dark green	95.21
11	Aluminium Foil	12.5x12.8	11.68x13	Silver	94.92

3. Color Detection Techniques:
- Convert image from **RGB to HSV** (Hue, Saturation, Value) using:
  \text{cv2.cvtColor(image, cv2.COLOR_BGR2HSV)}
- Define color ranges in HSV for classification.

4. Size Detection Techniques:
- Use pixel-to-real-world scaling based on a known reference object or camera calibration.
- Calculate object area in pixels, then convert to cm² using scale factor.
  If P_area = area in pixels, and 1 cm = s pixels, then:

$$Real\ Area(cm^2) = \frac{Parea}{s^2} \qquad (3)$$

5. Dimension Detection (Length & Width):
   w and h are width and height in a pixel

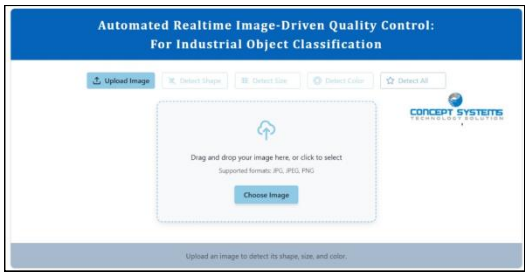
$$Real\ width = \frac{w}{s} \qquad (4)$$

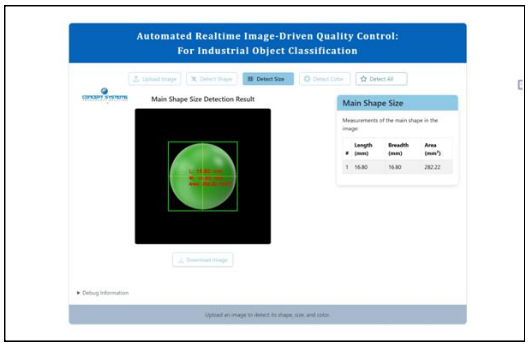
$$Real\ height = \frac{h}{s} \qquad (5)$$

Every object objects in database is verified by shape**,** color**,** size**, and** dimension detection typically works in an image processing or computer vision system, along with relevant formulas or methods used in each detection type as shown in Table 1

Part B: Web Page Implementation using Flask and React

This paper identifies the proposed algorithm as the optimal approach for developing a Flask application, which offers a cohesive platform for recognizing the shape, size, and color of objects as shown in figure 3,4,and 5.

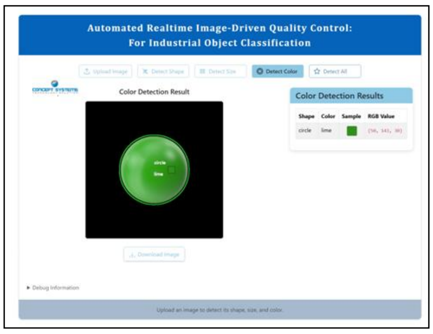

Automated Realtime Image-Driven Quality Control:
For Industrial Object Classification
Upload Image
Detect Shape
Detect Size
Detect Color
Detect All
CONCEPT SYSTEMS
TECHNOLOGY SOLUTION
Drag and drop your image here, or click to select
Supported formats: JPG, JPEG, PNG
Choose Image
Upload an image to detect its shape, size, and color.
Fig3: Web page for Proposed model


Automated Realtime Image-Driven Quality Control:
For Industrial Object Classification
Upload Image
Detect Shape
Detect Size
Detect Color
Detect All
CONCEPT SYSTEMS
Main Shape Size Detection Result
Download Image
Debug Information
Upload an image to detect its shape, size, and color.
Main Shape Size

Measurements of the main shape in the image:

#	Length (mm)	Breadth (mm)	Area (mm^2)
1	16.80	16.80	282.22

Fig4: Web page for Shape and Size detection


Automated Realtime Image-Driven Quality Control:
For Industrial Object Classification

Color Detection Result

Color Detection Results

Shape	Color	Sample	RGB Value
circle	lime	🟩	(56, 141, 30)

Upload an image to detect its shape, size, and color.

Fig5: Web page for color detection

II. FUTURE SCOPE

Expanding the system to include 3D imaging or stereo vision could provide more accurate measurements and allow for the detection of complex geometries or surface irregularities. We can explore the integration of deep learning models to improve object classification, adapt to new object types, and enhance decision-making for complex tasks such as defect detection or material type recognition.

III. CONCLUSION

The proposed system discussed here helps industries to detect and notify the defect at the right stage so that the companies can take corrective actions to mitigate those at the right stage. The idea uses the niche technologies to reduce the human intervention and to give the best quality standards 24x7. In this paper, we introduce a simple, contactless way to identify objects using image processing and OpenCV. The system can recognize an object's shape, size, and color with a high level of accuracy about 94.34% and it doesn't need any extra hardware to work. This article identifies the proposed algorithm as the optimal approach for developing a Flask application, which offers a cohesive platform for recognizing the shape, size, and color of objects.

ACKNOWLEDGMENT

The authors would like to thank all previous writers for their contributions to this field.

REFERENCES

[1] Arali, Ranjini, et al. "Colour Detection from Image." *International Journal of Advances in Engineering and Management (IJAEM)* 3.7 (2021): 2164-2171.

[2] Joy, Deborah T., et al. "Computer vision for color detection." *Int. J. Innov. Res. Comput. Sci. Technol* 9.3 (2021): 53-59.

[3] Abdullah-Al-Noman, Md, et al. "Computer vision-based robotic arm for object color, shape, and size detection." *Journal of Robotics and Control (JRC)* 3.2 (2022): 180-186.

[4] Wicaksono, M. F., and M. D. Rahmatya. "3D Geometric Shape and Colors Interactive Learning Media using Raspberry Pi, OpenCV, and TensorFlow Lite." *International Journal on Advanced Science, Engineering & Information Technology* 13.5 (2023).

[5] Pavithra, G., J. Jency Jose, and T. A. Chandrappa. "Real-time color classification of objects from video streams." *2017 2nd IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT)*. IEEE, 2017.

[6] Gustafsson, Simon, and Andreas Persson. "Detecting small and fast objects using image processing techniques: A project study within sport analysis." (2021).

[7] Messai, Yacine, et al. "Object tracking platform for color object detection using genetic algorithm optimization." *2020 International Conference on Electrical, Communication, and Computer Engineering (ICECCE)*. IEEE, 2020.

[8] Othman, Nashwan Adnan, et al. "An embedded real-time object detection and measurement of its size." *2018 International Conference on Artificial Intelligence and Data Processing (IDAP)*. IEEE, 2018.

[9] Zhang, YanJie, and XingHua Li. "A new object detection method in color image processing." *2011 4th International Congress on Image and Signal Processing*. Vol. 2. IEEE, 2011.

[10] Song, Jiqiang, Min Cai, and Michael R. Lyu. "Edge color distribution transform: an efficient tool for object detection in images." *2002 International Conference on Pattern Recognition*. Vol. 1. IEEE, 2002.

[11] Karma, I. Gede Made, I. Made Dwi Jendra Sulastra, and Jeni Susanti. "Object Detection Using Color Dissimilarity Based Segmentation Method." *2020 International Conference on Applied Science and Technology (iCAST)*. IEEE, 2020.

[12] Liu, Guang-Hai, and Jing-Yu Yang. "Exploiting color volume and color difference for salient region detection." *IEEE Transactions on Image Processing* 28.1 (2018): 6-16.

[13] Abdellatif, Mohamed. "Effect of color pre-processing on color-based object detection." *2008 SICE Annual Conference*. IEEE, 2008.

[14] Karanam, Madhavi, et al. "Object and it's dimension detection in real time." *E3S Web of Conferences*. Vol. 391. EDP Sciences, 2023.

[15] Ankad, Sushma S., et al. "Object Size Measurement from CCTV footage using deep learning." 2021 IEEE International Conference on Computation System and Information Technology for Sustainable Solutions (CSITSS). IEEE, 2021.

[16] Datta, Shreyasi, et al. "Object shape and size recognition from tactile images." *2013 International Conference on Control Communication and Computing (ICCC)*. IEEE, 2013.

[17] Basavaraj, M. U., and H. Raghuram. "Real Time Object Distance and Dimension Measurement using Deep Learning and OpenCV." *2023 Third International Conference on Artificial Intelligence and Smart Energy (ICAIS)*. IEEE, 2023.

[18] Anjanayya, S., et al. "Object size measurement using open CV (Computer Vision)." 2023 International Conference on Research Methodologies in Knowledge Management, Artificial Intelligence and Telecommunication Engineering (RMKMATE). IEEE, 2023.

[19] Basavaraj, M. U., and H. Raghuram. "Real Time Object Distance and Dimension Measurement using Deep Learning and OpenCV." *2023 Third International Conference on Artificial Intelligence and Smart Energy (ICAIS)*. IEEE, 2023.

[20] Anjanayya, S., et al. "Object size measurement using open CV (Computer Vision)." 2023 International Conference on Research Methodologies in Knowledge Management, Artificial Intelligence and Telecommunication Engineering (RMKMATE). IEEE, 2023.