Evolution in Electrical and Electronic Engineering Vol. 4 No. 2 (2023) 158-164
© Universiti Tun Hussein Onn Malaysia Publisher's Office

EEEE

Homepage: http://publisher.uthm.edu.my/periodicals/index.php/eeee
e-ISSN : 2756-8458

3D Geometric Shape Recognition System using YOLO v8 Implemented on Raspberry Pi

Chek Xiou Rue [1], Chessda Uttraphan[1]*

[1]Faculty of Electrical and Electronic Engineering,
Universiti Tun Hussein Onn Malaysia, Batu Pahat, Johor, 86400, MALAYSIA

*Corresponding Author Designation

DOI: https://doi.org/10.30880/eeee.2023.04.02.021
Received 20 June 2023; Accepted 30 August 2023; Available online 30 October 2023

Abstract: Recently, there has been a growing trend in the popularity of robots capable of performing complex tasks without human intervention, such as differentiating and picking up products with various shapes. These robots have found applications in diverse fields like manufacturing and servicing. To enable object recognition, artificial intelligence (AI) technology is essential for the robots. However, implementing AI algorithms is challenging and requires significant computational power. Additionally, for mobile robots, the use of microcontrollers is not an optimal choice; embedded processors are preferred. This paper introduces a system for recognizing 3D geometric shapes, employing YOLO v8 implemented on a Raspberry Pi. The proposed system combines computer vision techniques, a webcam, and a Raspberry Pi to identify two classes of 3D geometric shapes: cubes and spheres. The experimental results demonstrate that the system achieves an impressive mean average precision (mAP) of 98.3%, precision of 97.3% and recall of 94.4%. It is worth noting that the system is cost-effective, easily integrated into existing setups due to its compact form factor, high mobility, and low power consumption. In summary, the proposed 3D geometric shape recognition system demonstrates the potential of employing YOLO v8 on Raspberry Pi for real-time, affordable, and highly accurate 3D object recognition.

Keywords: 3D Geometric Shapes, Object Recognition, Raspberry Pi, You Only Look Once (YOLO)

1. Introduction

Object recognition is an area of computer vision where a robot can detect objects in an environment using a camera or sensor that is capable of extracting images of the robot's surroundings [1]. Object recognition based on their shapes is a crucial task in computer vision particularly in understanding digital images [2]. If a robot can recognize its environment and acquire this information using object recognition, it will be able to perform more complex tasks, such as grabbing objects or moving around in an environment. There are a lot of application areas depending on shape recognition that may include robotics, medical applications, and assistance for the impaired [2].

**Corresponding author: chessda@uthm.edu.my*

Robots commonly perform simple and repetitive tasks in various industries. However, their effectiveness is reliant on human intervention. Complex tasks, such as distinguishing and selecting products with diverse shapes, remain challenging for robots to accomplish autonomously. Several object recognition systems utilizing the You Only Look Once (YOLO) approach have been proposed. Notably, existing works have suggested the implementation of an object recognition system using YOLO on a Raspberry Pi, as documented in [4-7]. This paper aims to design a system capable of identifying and recognizing various types of 3D geometric shapes using YOLO v8. The approach involves training the model on a publicly available dataset and subsequently implementing and validating the design on a Raspberry Pi. This system boasts cost-effectiveness, easy integration into existing setups due to its compact form factor, high mobility, and low power consumption.

2. Methodology

2.1 Overview of the work

Figure 1 illustrates the overview of the work. A total of 200 images categorized to 100 cubes and 100 spheres are selected for the training of the model from M. Wielgosz [8]. The annotation of dataset was using Roboflow annotation tool to label the dataset. The number of images has increased to 420 images after the preprocessing and augmentation in Roboflow with the image size 640 pixels by 640 pixels. The dataset is exported from Roboflow in a snippet of YOLO v8 format generated by Roboflow and is utilized within the Jupyter notebook in Google Colab for the purposes of training, validating, and testing. Once the dataset achieves an optimized mean average precision (mAP), precision and recall score of over 90% respectively, the model is deployed to the Raspberry Pi. Subsequently, experimental work involving real objects is conducted using a webcam.

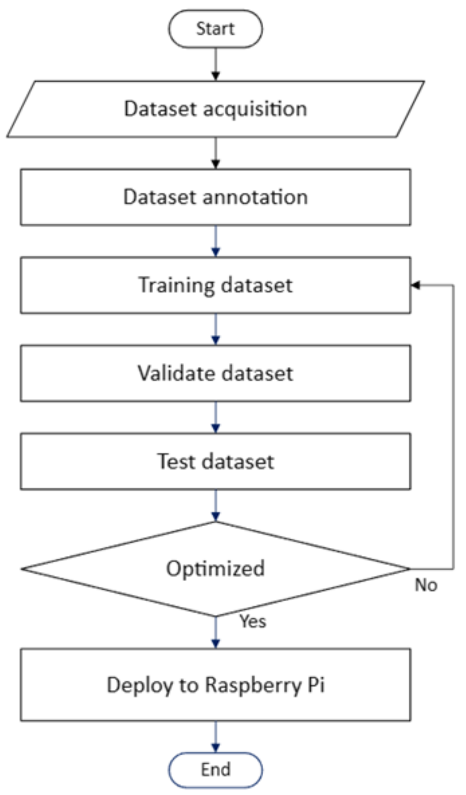


Figure 1: Overview of the work

2.2 Working principle

Figure 2 shows the working principle of the 3D object recognition system explained in the form of block diagram. YOLO v8 algorithm is used to train the model and Python is used for the coding of the system. The camera model Logitech C310 is used to detect the 3D objects such as cube and sphere in real time and the results are displayed in a window. In addition, the Raspberry Pi Model 4B with 8GB RAM is utilized for the system implementation.

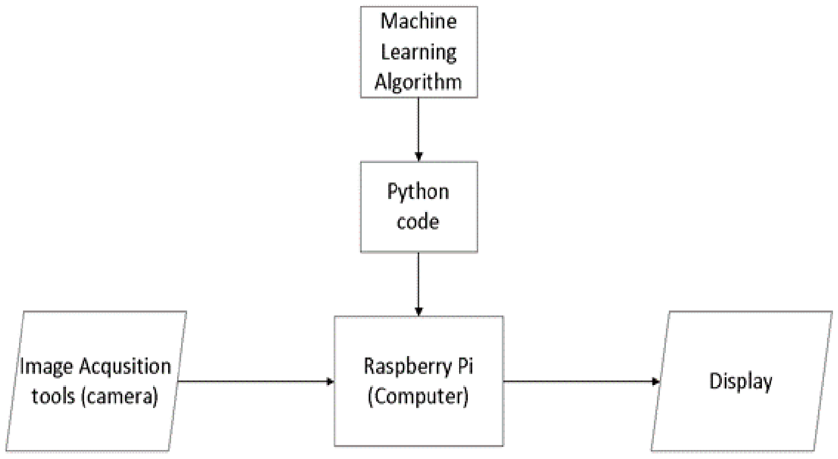


Figure 2: Block diagram of the system

2.3 Hardware components

2.3.1 Raspberry Pi

Raspberry Pi 4 Model B (Figure 3), a compact computer board, is ideal for processor industrial works. It comes with an 8GB RAM, 64-bit ARM Cortex-A72 and quad core. It also features a convenient micro-SD card slot for loading operating system and data storage. With built-in Wi-Fi and Bluetooth, it facilitates wireless connectivity for external communication.

Figure 3: Raspberry Pi 4

2.3.2 Webcam

The Logitech C310 HD Webcam (Figure 4) is used in this work for capturing real time objects such as cube and sphere with 720 pixels. It is a portable and great tool for taking pictures via USB port in the Raspberry Pi. With its 60° diagonal field of view and auto light correction, this will elevate the visual output.

Figure 4: Logitech C310 HD webcam

2.3.3 Software setup

A total of 200 images categorized to 100 cubes and 100 spheres are selected for the training of the model from M. Wielgosz [8]. Figure 5 shows the pseudocode of the system, which consists of seven parts. Firstly, the system needs to import necessary modules such as base64, json, time, as well as libraries including cv2 and requests. Then, it reads and loads a JSON configuration file that contains the API key, Roboflow model, Roboflow model ID, and Roboflow version number. A URL is constructed to call the Roboflow API for object detection in the system.

```
Import necessary modules such as base64, json, time and also libraries
namely cv2 and requests.

# Load configuration from a JSON file
With open("the location of the configuration file") as f:
    config = json.load(f)
    # Extract configuration values
    ROBOFLOW_API_KEY = config["ROBOFLOW_API_KEY"]
    ROBOFLOW_SIZE = config["ROBOFLOW_SIZE"]
    ROBOFLOW_MODEL_ID = config["ROBOFLOW_MODEL_ID"]
    ROBOFLOW_VERSION_NUMBER = config["ROBOFLOW_VERSION_NUMBER"]

# Construct the Roboflow Infer URL
upload_url = ConstructURL(ROBOFLOW_MODEL_ID, ROBOFLOW_VERSION_NUMBER,
ROBOFLOW_API_KEY)

# Get webcam interface
video = cv2.VideoCapture(0)

# Function to perform inference
Function infer():
    ret, img = video.read()
    img = ResizeImage(img, ROBOFLOW_SIZE)
    img_str = EncodeImageToString(img)
    resp = SendRequest(upload_url, img_str)
    predictions = ParseResponse(resp)
    DrawBoundingBoxes(img, predictions)
    return img, predictions

# Main loop
While True:
    If KeyPressed('q'):
        break
    start = GetCurrentTime()
    image, detections = infer()
    DisplayImage(image)
    PrintFPS(start)
    PrintDetections(detections)

# Release resources
video.release()
cv2.destroyAllWindows()
```

Figure 5: The pseudocode of the system

Next, the system initializes the access to the webcam by creating a video capture object. A function named 'infer' is defined to capture objects from the webcam, resize and encode them, and send them to the Roboflow Infer API for prediction. The API response is parsed, and the bounding boxes and class labels of the objects are drawn on the image using OpenCV functions.

The main loop continuously captures images, performs inference, and displays the detected objects' results. Additionally, it calculates and prints the achieved frames per second (FPS). The loop continues until the user presses 'q'. Upon pressing 'q', the system stops executing, and all open windows are closed.

2.5 Performance metrics

To evaluate YOLO models, mean average precision (mAP) is involved as a performance metric. mAP is defined in Eq. 1 and commonly used as a metric to measure object detection performance, encompassing precision-recall (PR) area under the curve (AUC), and Intersection over Union (IoU) [9].

$$mAP = \frac{1}{N}\sum_{i=1}^{N} AP_i \qquad \text{Eq.1}$$

AP_i = Average precision of each class

N = Total number of classes

$$Recall = \frac{True\ Positive\ (TP)}{True\ Positive\ (TP) + False\ Negative\ (FN)} \times 100\% \qquad \text{Eq.2}$$

$$Precision = \frac{True\ Positive\ (TP)}{True\ Positive\ (TP) + False\ Positive\ (FP)} \times 100\% \qquad \text{Eq.3}$$

Recall reveals the model ability to identify positive instances while precision shows the accuracy of the positive predictions made by the model. Recall and precision can be calculated by using Eq.2 and Eq.3, respectively.

3. Results and Discussion

Using the YOLO v8s with default parameters, the model is trained through 100 epochs and the result of mAP50 is considered for performance analysis as the Intersection of Union (IoU) is within 50%. Figure 6 presents an epochs curve that shows the relationship between the number of training epochs and the performance of the machine learning model. Number of training epochs is represented as x-axis which implies the quantity of the model iteration over the training dataset while y-axis portrays the evaluation metric. Ideally, the graph was expected to follow the orange dotted line. However, the graph shows a sudden spike at the early first epoch and a dip at the third epoch and 21st epoch. This scenario is common during the initial stage of training as it is also depicted in This scenario is common during the initial stage of training as it is also depicted in [10] - [11]. This is attributed to in the early stage of the training, the model undergoes parameter adjustments in order to determine the most favorable configuration [11]. As the training progresses, the model is converging towards optimal performance where it follows the ideal shape of the curve due to the model becomes familiar with the data [11].

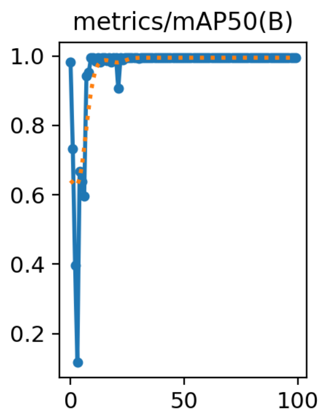


Figure 6: mAP 50 of the trained results

Based on Figure 7, a curve of precision against recall is presented and it shows that the model is a near to perfect model as point one corresponds to threshold [0, 1] and point two corresponds to threshold= 0 and the area under the curve (AUC), AUC = 1. For cube, the precision is about 0.995 and for sphere, the precision is about 0.995. In short, the model is a near to perfect model because the use of YOLO v8s that increase 20.05 % for detection and increase 1.12% for classification [12].

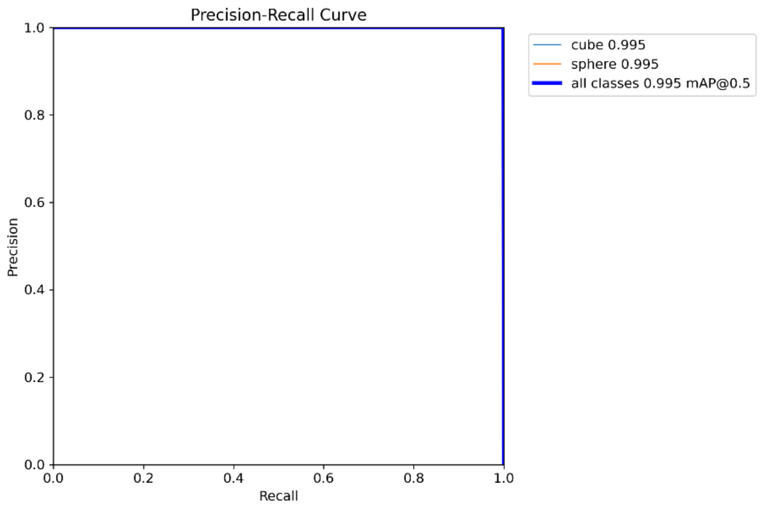
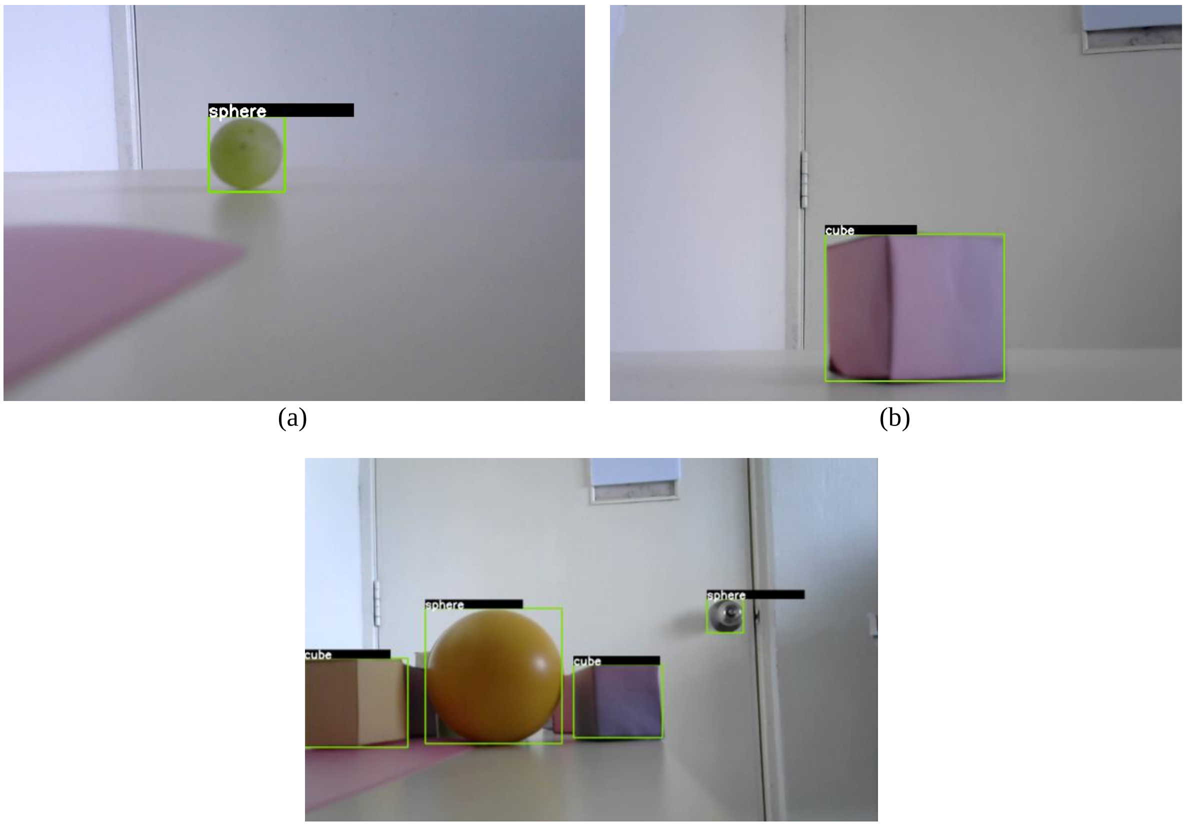


Figure 7: Precision and Recall curve

Additionally, to demonstrate real-time functionality, the proposed system is implemented on the Raspberry Pi, as depicted in Figure 8.



(a) (b)

(c)

Figure 8: Tested in real time of (a) spherical, (b) cubical and (c) multiple shapes objects

4. Conclusion

In conclusion, this study successfully implemented a real-time 3D geometric shape recognition system using YOLO v8 on the Raspberry Pi platform. The main contribution of this work is demonstrating the Raspberry Pi's capability to utilize machine learning algorithms and achieve an impressive mAP value of 98.3%, precision value of 97.3% and recall value of 94.4%.

Acknowledgement

The Faculty of Electrical and Electronic Engineering (FKEE), Universiti Tun Hussein Onn Malaysia (UTHM), in particular, deserves our sincere gratitude for their valuable contributions that were essential to the success of our project.

References

[1] Alberola, Álvaro Morena Gallego, Gonzalo Molina Maestre, Unai Garay. (2019). Artificial Vision and Language Processing for Robotics - 8.2 Multiple Object Recognition and Detection. (pp. 260). Packt Publishing.

[2] L. Fernandes and B. R. Shivakumar, "Identification and Sorting of Objects based on Shape and Colour using robotic arm," 2020 Fourth International Conference on Inventive Systems and Control (ICISC), 2020, pp. 866-871, doi: 10.1109/ICISC47916.2020.9171196.

[3] A. Salari, A. Djavadifar, X. Liu, and H. Najjaran, “Object recognition datasets and challenges: A review,” Neurocomputing, vol. 495, pp. 129–152, Jul. 2022, doi: 10.1016/j.neucom.2022.01.022.

[4] H. H. Nguyen, T. N. Ta, N. C. Nguyen, V. T. Bui, H. M. Pham and D. M. Nguyen, "YOLO Based Real-Time Human Detection for Smart Video Surveillance at the Edge," 2020 IEEE Eighth International Conference on Communications and Electronics (ICCE), Phu Quoc Island, Vietnam, 2021, pp. 439-444, doi: 10.1109/ICCE48956.2021.9352144.

[5] A. Narendra, S. R. Choudhury, A. Mishraand I. Misra, “Chaurah: A Smart Raspberry Pi based Parking System”. TechRxiv, 24-Jan-2023, doi: 10.36227/techrxiv. 21931398.v1.

[6] Matthew Danish, Justas Brazauskas, Rob Bricheno, Ian Lewis, and Richard Mortier, "DeepDish: multi-object tracking with an off-the-shelf Raspberry Pi," Proceedings of the Third ACM International Workshop on Edge Systems, Analytics and Networking (EdgeSys '20). Association for Computing Machinery, New York, NY, USA, 37–42. 2020, doi: 10.1145/3378679.3394535

[7] A. B. Wahyutama and M. Hwang, “YOLO-Based Object Detection for Separate Collection of Recyclables and Capacity Monitoring of Trash Bins,” Electronics, vol. 11, no. 9, p. 1323, Apr. 2022, doi: 10.3390/electronics11091323.

[8] M. Wielgosz, “Primitives dataset,” Wielgosz.info, 2017. https://data.wielgosz.info/.

[9] Ang, et al., “A novel application for real-time arrhythmia detection using YOLOv8,” arXiv.org, 2023. https://arxiv.org/abs/2305.16727 (accessed Jun. 10, 2023).

[10] S. Ali, Abdullah, A. Athar, M. Ali, A. Hussain and H. -C. Kim, "Computer Vision-Based Military Tank Recognition Using Object Detection Technique: An application of the YOLO Framework," 2023 1st International Conference on Advanced Innovations in Smart Cities (ICAISC), Jeddah, Saudi Arabia, 2023, pp. 1-6, doi: 10.1109/ICAISC56366.2023.10085552.

[11] Y. Zhang et al., “YOLO-Sp: A Novel Transformer-Based Deep Learning Model for Achnatherum splendens Detection,” Agriculture, vol. 13, no. 6, p. 1197, Jun. 2023, doi: 10.3390/agriculture13061197.

[12] YOLOv8 Ultralytics: State-of-the-Art YOLO Models,” LearnOpenCV – Learn OpenCV, PyTorch, Keras, Tensorflow with examples and tutorials, Jan. 10, 2023. https://learnopencv.com/ultralytics-yolov8/