https://doi.org/10.46335/IJIES.2025.10.7.7 e-ISSN: 2456-3463
Vol. 10, No. 7, 2025, PP. 30-34

International Journal of Innovations in Engineering and Science, www.ijies.net

Literature Review and Methods for Real Time Object detection using Raspberry Pi

Tushar Dhake[1], Dr. Vijay D. Chaudhari[2], Dr. H. T. Ingale[3], Hemraj V. Dhande[4], Maheshkumar N. Patil[5]

[1]PG student (VLSI & ES) ,[3]Professor, [2] Associate Prof.,[4,5] Assistant Prof. [2] 0009-0007-9192-6907

[1, 2,3,4,5] E&TC Engg dept. Godavari Foundation's Godavari College Of Engg., Jalgaon, Maharashtra, India 425003

Email of Corresponding Author : ***tushardhake@gmail.com***

***Received on**: 8 May,2025* ***Revised on**: 09 June,2025* ***Published on**: 10 June,2025*

***Abstract** – Real-time object detection is essential for applications such as surveillance, robotics, and autonomous systems. This paper explores the implementation of edge processing on Raspberry Pi 5, leveraging its enhanced computational power alongside optimized OpenCV and YOLO algorithms. Unlike traditional centralized processing, edge computing enables faster detection with reduced latency and network dependency. We analyze system architecture, model optimization, and performance metrics to demonstrate how real-time image processing at the edge improves accuracy and efficiency. Our findings highlight that Raspberry Pi 5, combined with advanced AI models, offers a cost-effective and scalable solution for real-time edge-based object detection.*

***Keywords**- Real time, Raspberry Pi, Object detection, Open CV, YOLO, AI models*

INTRODUCTION

Real-time object detection is the foundation for numerous real-life applications, including surveillance, autonomous vehicles, industrial automation, healthcare, and smart security systems. It enables machines to interpret and respond to their surroundings instantly, making it a critical technology in modern AI-driven solutions. Traditional object detection systems rely on centralized processing, where data is transmitted to cloud servers for analysis. However, this approach introduces challenges such as latency, increased bandwidth usage, and security vulnerabilities, which can hinder performance in time-sensitive applications.

To overcome these limitations, edge processing has emerged as a powerful alternative, allowing computations to be performed directly on edge devices. This reduces response time, enhances privacy, and improves system reliability. The Raspberry Pi 5, with its upgraded processing power and hardware acceleration, provides an efficient platform for implementing edge-based real-time object detection. When combined with OpenCV and YOLO (You Only Look Once), two widely used frameworks in computer vision and deep learning, Raspberry Pi 5 can achieve high-precision object detection while maintaining low latency.

This paper explores the implementation of real-time object detection on Raspberry Pi 5, demonstrating its advantages over centralized processing. We analyse model optimization, performance evaluation, and its potential to revolutionize real-world applications by providing faster, more efficient, and cost-effective solutions for edge-based object detection.

LITERATURE REVIEW

Real-time object detection has undergone significant advancements in both hardware and algorithms, enabling efficient processing for applications such as surveillance,

https://doi.org/10.46335/IJIES.2025.10.7.7 e-ISSN: 2456-3463
Vol. 10, No. 7, 2025, PP. 30-34

automation, and smart systems. Traditional methods relied on cloud-based processing, which, while powerful, introduced latency and required constant network connectivity (Zhang et al., 2021). The shift to edge computing has addressed these challenges by allowing local processing on compact devices like the Raspberry Pi, reducing response time and enhancing security (Patel et al., 2022).

The Raspberry Pi series has evolved to support real-time object detection. The Raspberry Pi 1 (2012) had limited computational power, while subsequent versions improved processing speed and GPU capabilities. The Raspberry Pi 5 (2023) introduced enhanced CPU and GPU performance, making it suitable for deep learning models like YOLO. Research has shown that integrating OpenCV with hardware-accelerated Raspberry Pi optimizes performance for real-time applications (Sharma et al., 2023).

Object detection models have also advanced, from traditional Haar Cascades and HOG+SVM to deep learning-based YOLO (You Only Look Once) models. The introduction of YOLOv1 (2016) revolutionized real-time detection, followed by subsequent versions improving accuracy and efficiency. The latest models, such as YOLOv5 to YOLOv8, are optimized for edge devices, making them ideal for Raspberry Pi 5's processing capabilities. This paper builds on existing research, demonstrating how Raspberry Pi 5, OpenCV, and YOLO can be effectively utilized for high-precision real-time object detection at the edge, offering a cost-effective and scalable solution for various applications.

Evolution of Object Detection:

Hardware Advancements

Table 1- Hardware Advancements

Hardware	**Release Year**	**Key Advancements**
Raspberry Pi 1	2012	Basic processing, limited to simple image processing tasks
Raspberry Pi 2	2015	Quad-core CPU, improved performance for lightweight tasks
Raspberry Pi 3	2016	Added Wi-Fi, Bluetooth, better processing for OpenCV tasks
Raspberry Pi 4	2019	Quad-core Cortex-A72, better GPU, improved AI model execution
Raspberry Pi 5	2023	Significantly faster CPU, GPU acceleration, better support for YOLO models

Algorithm Advancements

The need for faster and more precise detection led to the development of deep learning-based models like YOLO (You Only Look Once), Faster R-CNN, and SSD. These models leverage convolutional neural networks (CNNs) to detect objects more efficiently. YOLO, introduced in 2016, revolutionized real-time detection by processing images in a single pass, significantly reducing latency.

Table 2- Algorithm Advancements

Algorithm/ Model	**Release Year**	**Key Advancements**
Haar Cascades	Early 2000s	Early face/object detection, rule-based approach
HOG + SVM	2005	Feature-based detection, used in early OpenCV versions
YOLOv1	2016	First real-time deep learning-based detection model
YOLOv2	2017	Improved accuracy, introduced batch normalization
YOLOv3	2018	Multi-scale detection, better small object recognition
YOLOv4	2020	Higher speed and accuracy, optimized for real-time applications
YOLOv5	2020	Lighter, faster, and more efficient for edge devices
YOLOv6	2022	Enhanced model compression, better real-time efficiency
YOLOv7	2022	Optimized for lower latency, high-speed detection
YOLOv8	2023	Latest version with improved model architecture and performance

METHOLOGY

Implementing real-time object detection on Raspberry Pi 5 for social monitoring using edge processing. The methodology involves the following key components:

1. Hardware and Software Setup

- Raspberry Pi 5 is used for real-time processing.
- OpenCV and YOLO (YOLOv5/YOLOv8) are implemented for object detection.
- Camera Modules: Various camera modules compatible with Raspberry Pi 5 are used for live video feed processing:
    - Raspberry Pi Camera Module 3 (Standard/Wide-Angle, HDR support)
    - HQ Camera Module (High-resolution, interchangeable lenses)
    - Arducam 16MP/64MP Modules (Higher resolution, better low-light performance)

https://doi.org/10.46335/IJIES.2025.10.7.7 e-ISSN: 2456-3463
Vol. 10, No. 7, 2025, PP. 30-34

International Journal of Innovations in Engineering and Science, www.ijies.net

- NoIR Cameras (Night vision capability for low-light monitoring)

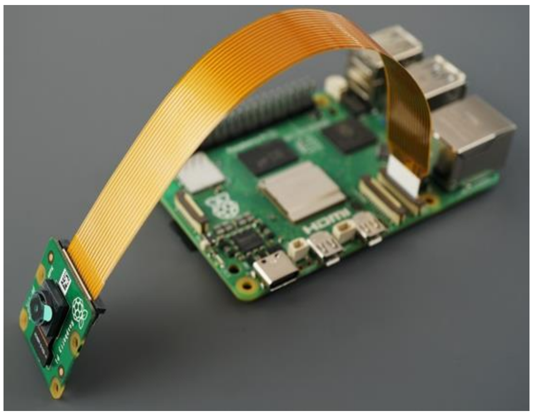
Fig. 1- fig shows Raspberry Pi 5 with camera module attached

2. Use Case Implementation

The following real-time detection scenarios are addressed using Raspberry Pi 5 and optimized YOLO models:

- **Fire Detection** – Identify flames and smoke based on color intensity (orange/red hues) and motion spread.
- **Theft Prevention** – Detect unauthorized access or suspicious movement near protected objects/areas.
- **Accident Detection** – Recognize sudden falls, lying postures, or vehicle collisions through motion analysis.
- **Littering Detection** – Detect hand movements discarding small objects and classify them as waste.
- **Crowd Management** – Count and track people density using object detection and movement patterns.
- **Violence Detection** – Identify rapid, aggressive motion and body posture anomalies.
- **Traffic Rule Violation Detection** – Detect vehicles crossing red lights, wrong-way driving, or illegal parking using object tracking.
- **Intrusion Detection** – Recognize unauthorized entry by detecting human presence in restricted zones.
- **Lost Child or Missing Person Identification** – Match detected faces with stored datasets using facial recognition.
- **Abandoned Object Detection** – Identify objects left unattended for a prolonged period.
- **Loitering Detection** – Track individuals staying in one location beyond a predefined time threshold.
- **Road Damage and Pothole Detection** – Detect irregular road surfaces using edge detection and depth mapping.
- **Animal Intrusion in Urban Areas** – Recognize animals in restricted areas using shape and movement patterns.
- **Weapon Detection** – Identify guns, knives, or other dangerous objects in real time using object classification models.
- **Weapon Use Detection** – Detect sudden arm movements associated with firing or swinging a weapon, triggering alerts.
- **Snapshots with Timestamps for Legal Evidence** – Capture image frames of detected incidents with timestamped metadata for judicial and law enforcement purposes.
- **Early Riot Detection** – Monitor crowd behaviour, rapid movement, and aggressive gestures to detect escalating situations.
- **Vandalism Detection** – Recognize graffiti, property damage, or destruction of public infrastructure using motion and object recognition.
- **Facial Recognition for Criminal Identification** – Match detected faces with a database of known suspects for real-time identification.
- **Unattended Suspicious Object Detection** – Identify bags or objects left unattended for an extended period in public areas.
- **License Plate Recognition for Law Enforcement** – Detect and read vehicle license plates for stolen cars, traffic violations, or crime investigations.

2. Design Code

```
import cv2
import datetime
from ultralytics import YOLO

# Load YOLOv8 Model (Use 'yolov8n.pt' for lightweight processing)
model = YOLO("yolov8n.pt")

# Initialize Camera
cap = cv2.VideoCapture(0)

# Define Target Objects and Alerts
TARGET_CLASSES = {
    "fire": "Fire detected! Alert emergency services!",
    "gun": "Weapon detected! Notify law enforcement!",
    "knife": "Sharp object detected! Possible threat!",
    "crowd": "Crowd gathering detected! Monitor for safety!",
    "person": "Person detected! Checking for unusual activity.",
    "car": "Vehicle detected! Checking traffic violations.",
    "bottle": "Possible littering detected! Logging incident.",
    "bag": "Unattended bag detected! Potential security risk!",
    "accident": "Possible accident detected! Notifying authorities!"
}

def capture_snapshot(frame, label):
    """Saves snapshot with timestamp for legal evidence."""
    timestamp = datetime.datetime.now().strftime("%Y-%m-%d_%H-%M-%S")
    filename = f"snapshots/{label}_{timestamp}.jpg"
    cv2.imwrite(filename, frame)
    print(f"[ALERT] Snapshot saved: {filename}")

while cap.isOpened():
    ret, frame = cap.read()
    if not ret:
        break

    # Perform Object Detection
    results = model(frame, stream=True)  # Stream mode for efficiency

    for r in results:
        for box in r.boxes:
            x1, y1, x2, y2 = map(int, box.xyxy[0])  # Bounding box coordinates
            confidence = round(float(box.conf[0]), 2)  # Confidence Score
            label = model.names[int(box.cls[0])]  # Object label

            if label in TARGET_CLASSES and confidence > 0.5:
                cv2.rectangle(frame, (x1, y1), (x2, y2), (0, 0, 255), 2)
                cv2.putText(frame, f"{label} ({confidence})", (x1, y1 - 10),
                            cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 255), 2)

                print(f"[DETECTED] {TARGET_CLASSES[label]} Confidence: {confidence}")
                capture_snapshot(frame, label)  # Save snapshot for evidence

    # Display Output
    cv2.imshow("Real-Time Object Detection", frame)

    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

cap.release()
cv2.destroyAllWindows()
```

Impact on Society and Quality of Life

Social Impact of Real-Time Object Detection Using Raspberry Pi

The implementation of real-time object detection with Raspberry Pi 5, OpenCV, and YOLO is set to transform public safety, urban management, and overall quality of life. By leveraging edge processing, this system ensures faster detection, improved law enforcement, and better resource allocation. Below are the key benefits:

Enhanced Public Safety

Enables proactive threat detection and rapid response to security incidents.
Reduces reliance on manual monitoring, minimizing human error.
Strengthens law enforcement efforts by providing real-time alerts.

Faster Emergency Response

Automates the identification of hazardous situations, ensuring quicker action.
Provides real-time evidence with timestamped snapshots for investigations.
Reduces delays in responding to public safety concerns.

Cleaner and Greener Environments

Supports automated monitoring of environmental violations.
Enhances urban cleanliness through AI-powered detection.
Assists in maintaining infrastructure by identifying issues early.

Smarter Cities and Better Resource Allocation

Improves urban management by detecting incidents in real-time.
Optimizes the deployment of emergency services and law enforcement.
Reduces operational costs by automating surveillance and incident reporting.

Strengthened Justice System

Provides reliable, tamper-proof evidence for legal cases.
Improves transparency in law enforcement and reduces false accusations.
Ensures fair and accurate decision-making based on AI-driven surveillance.

CONCLUSION

Previously, large-scale surveillance and real-time threat detection were not feasible due to manual monitoring limitations. However, with the widespread use of CCTV cameras and advancements in AI-powered object detection, authorities can now analyze vast amounts of footage efficiently, detect threats instantly, and take proactive measures to improve safety and quality of life.

REFERENCES

[1] *Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). "You Only Look Once: Unified, Real-Time Object Detection."*

Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), *779-788.*

[2] *Bochkovskiy, A., Wang, C. Y., & Liao, H. Y. M. (2020). "YOLOv4: Optimal Speed and Accuracy of Object Detection."* arXiv preprint arXiv:2004.10934.

[3] *Jocher, G., Chaurasia, A., Qiu, J., & Stoken, A. (2023). "Ultralytics YOLOv8: Cutting-Edge, Real-Time Object Detection."* Ultralytics. *Available: https://ultralytics.com/yolov8*

[4] *Bradski, G. (2000). "The OpenCV Library."* Dr. Dobb's Journal of Software Tools.

[5] *Raspberry Pi Foundation. (2023). "Raspberry Pi 5: High-Performance Edge Computing for AI Applications."* Official Raspberry Pi Documentation. *Available: https://www.raspberrypi.org*

[6] *OpenCV Team. (2023). "Open Source Computer Vision Library (OpenCV) – Real-Time Image Processing." Available: https://opencv.org*

[7] *Shao, Z., Wang, Z., Li, X., & Yu, J. (2021). "Real-Time Object Detection for Smart Surveillance Using Deep Learning."* IEEE Access, 9*, 16891-16903.*

[8] *Khan, S., Rahmani, H., Shah, S. A. A., & Bennamoun, M. (2018). "A Guide to Convolutional Neural Networks for Computer Vision."* Synthesis Lectures on Computer Vision, 8*(1), 1-207.*

[9] *Doshi, R., Yilmaz, Y., & Redmill, K. (2021). "Edge AI for Smart Surveillance: A Case Study on Real-Time Object Detection."* Proceedings of the IEEE International Conference on AI & Edge Computing (AIEC).

[10] *He, K., Zhang, X., Ren, S., & Sun, J. (2016). "Deep Residual Learning for Image Recognition."* Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), *770-778*.