Journal of Next Generation Technology (JNxtGenTech)
Vol. 5, Issue 2, April 2025

Video Streaming with Object Detection using RaspberryPI

J.Aarthi[1], K.Subashini[2], P. Venmani[3], Y.Jenitha[4], S.Kaliyammal[5]

[1,2,3,4,5]Department of Electronics and Communication Engineering, Chendhuran College of Engineering and Technology, Tamilnadu, India.

Email id: arthijeyaraman032@gmail.com[1] , ssubashini652@gmail.com[2], venmanivenmani531@gmail.com[3], yesudassjeni05@gmail.com[4], kaliyammalsethu123@gmail.com[5] .

Article Received: 28 April 2025 *Article Accepted: 29 April 2025* *Article Published: 30 April 2025*

Citation
J.Aarthi, K.Subashini, P. Venmani, Y.Jenitha, S.Kaliyammal, "Video Streaming with Object Detection using RaspberryPI", Journal of Next Generation Technology (ISSN: 2583-021X) , vol. 5, no. 2, pp. 133-139, April 2025. DOI: 10.5281/zenodo.15667604

Abstract

This project explores the integration of real-time video streaming and object detection using a Raspberry Pi, a compact and cost-effective single-board computer. The primary goal is to create a lightweight and efficient surveillance system capable of detecting and identifying objects within its field of view while transmitting the video feed over a network. Leveraging the capabilities of Openand machine learning models, particularly pre-trained deep learning models like MobileNet SSD, YOLO, or TensorFlow Lite, the system is optimized for the limited computational resources of the Raspberry Pi. The setup includes a Raspberry Pi (preferably Raspberry Pi 4 for better performance), a compatible camera module, and necessary software libraries. The video feed is captured via the Pi Camera or USB webcam, processed for object detection, and then streamed live using protocols such as MJPEG, RTSP, or via a web interface using Flask. The real-time object detection highlights identified objects with bounding boxes and labels, offering users an interactive and informative viewing experience. To ensure real-time performance, frame resizing and model optimization techniques are applied. TensorFlow Lite and OpenCV DNN modules are used to reduce inference time while maintaining accuracy. The system supports detection of multiple object categories including persons, vehicles, animals, and common objects, making it applicable in security monitoring, smart home automation, wildlife observation, and educational projects. Moreover, the project addresses challenges like network latency, limited processing power, and power management. By deploying efficient algorithms and lightweight models, the system achieves a balance between speed and accuracy. The video stream can be accessed remotely from a browser or mobile device, enhancing the utility and accessibility of the system. The project emphasizes modularity and scalability, allowing users to adapt it for specific use cases such as facial recognition, license plate detection, or integration with cloud services for advanced analytics. The open-source nature of the software stack ensures customization and community support.

Keywords: Sodium silicate, Rice husk ash, Activated Carbon

I. Introduction

The integration of computer vision and embedded systems has revolutionized how we perceive and interact with our surroundings. One of the most prominent applications of this integration is real-time object detection within video streams, a field that has garnered significant attention in domains such as security, automation, traffic monitoring, and smart home systems[1]-[3]. Traditionally, these applications required high-performance computers, GPUs, and expensive proprietary software. However, with advancements in low-cost computing platforms like the Raspberry Pi and the availability of open-source libraries, it is now possible to build and deploy intelligent vision systems at the edge, making the technology more accessible and scalable. This project focuses on implementing a video streaming system integrated with real-time object detection capabilities using a Raspberry Pi.

The Raspberry Pi is a small, cost-effective, and energy-efficient single-board computer that provides an ideal platform for prototyping and deploying lightweight artificial intelligence (AI) models [4]. It supports a wide range of peripherals, operating systems, and programming languages, making it an excellent choice for embedded computer vision projects. In this project, the Raspberry Pi captures live video using a Pi Camera or USB webcam, performs object detection on the video frames using pre-trained deep learning models, and streams the processed video to a remote client via a local network or the internet.

Object detection is a critical subfield of computer vision that involves identifying objects within an image or video and locating them using bounding boxes. Unlike simple image classification, object detection must identify multiple objects within a single frame and distinguish between different classes. This project utilizes lightweight and efficient deep learning models such as MobileNet-SSD, YOLOv3-tiny, or TensorFlow Lite models [5]-[6], which are optimized for limited-resource environments like the Raspberry Pi. These models are capable of identifying common object classes, including people, vehicles, animals, and household items. The video processing pipeline begins with frame acquisition through the camera, followed by resizing and pre-processing the frames to meet the input requirements of the selected object detection model. Each frame is then passed through the model to detect and classify objects. The results are overlaid onto the frames using bounding boxes and class labels. To make the system interactive and accessible, the annotated video stream is served using a Flask-based web application that can be accessed remotely through a browser.

This project also tackles several challenges associated with deploying computer vision models on embedded devices. These include computational constraints, memory limitations, power consumption, and real-time processing requirements. To mitigate these challenges, techniques such as image resizing, model quantization, frame skipping, and hardware acceleration (e.g., using the Raspberry Pi's GPU or external accelerators like the Coral USB Accelerator) can be employed. Despite the hardware limitations, the system aims to maintain a balance between inference speed and detection accuracy to provide a usable and responsive application. The practical applications of such a system are numerous. In a security context, it can be used to detect intruders or monitor restricted zones. In a smart home, it can recognize human activity or track pets. In agriculture, it could monitor animals or crops. The modular design of the system also allows for future enhancements, such as adding object tracking, facial recognition, sending alerts via SMS or email, cloud integration for data logging, or interfacing with other IoT devices. Moreover, the educational value of this project is significant. It offers hands-on experience in computer vision, deep learning, web development, and embedded systems. It also promotes problem-solving and system design skills by encouraging the user to deal with real-world constraints and trade-offs.

In conclusion, this project demonstrates how the combination of real-time video streaming and object detection can be effectively implemented using a low-cost, energy-efficient device like the Raspberry Pi. It highlights the growing potential of edge computing in deploying intelligent applications that are not only efficient and responsive but also scalable and affordable. As the capabilities of embedded systems continue to grow, projects like this pave the way for widespread adoption of AI at the edge, making smart environments more accessible and intelligent.

II. System Description

The proposed system, "Video Streaming with Object Detection Using Raspberry Pi," is designed to demonstrate how low-cost embedded hardware can be used to implement real-time video analytics using computer vision and artificial intelligence techniques. The core of the system is the Raspberry Pi, a compact and energy-efficient single-board computer that offers sufficient computational power for lightweight AI applications. The primary function of the system is to capture live video using a camera module connected to the Raspberry Pi, detect objects within the captured video frames using deep learning models, and stream the annotated video over a network for remote viewing. This setup

can serve as a foundational platform for various applications including security surveillance, home automation, smart monitoring systems, and educational projects.

The system is composed of both hardware and software components that work together in a pipeline structure. On the hardware side, the Raspberry Pi 4 Model B is preferred due to its improved processing capabilities, RAM size, and support for dual-band Wi-Fi. It is connected to a Pi Camera Module or a standard USB webcam, which is responsible for capturing the video feed in real time. The captured video is then processed by the Raspberry Pi using a Python-based script that incorporates computer vision libraries such as OpenCV [7]. The object detection is performed using pre-trained deep learning models such as MobileNet-SSD, YOLOv3-tiny, or TensorFlow Lite models that are optimized for resource-constrained environments. These models are chosen for their balance between speed and accuracy, allowing them to operate effectively on the limited hardware of the Raspberry Pi [8].

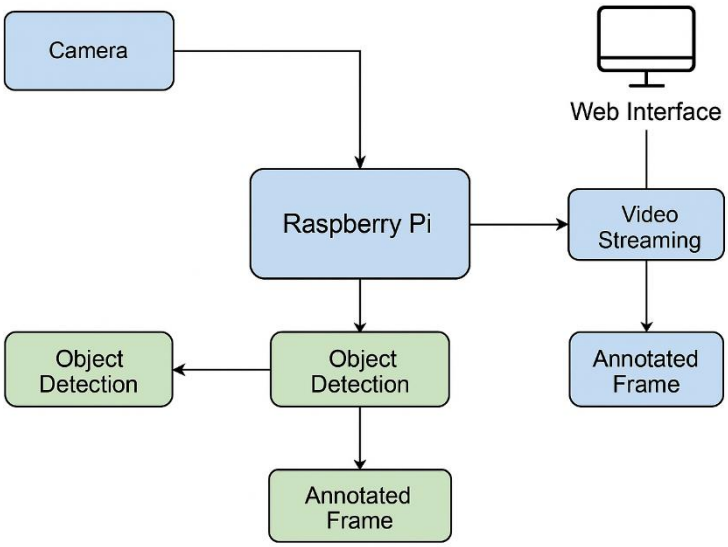

Fig 1. Proposed system block diagram

Each video frame captured by the camera is preprocessed—typically resized and normalized—to meet the input requirements of the object detection model. After preprocessing, the frame is passed through the inference engine, where the model detects objects and returns the bounding boxes, class labels, and confidence scores for each detected object. These results are then superimposed on the original frame using rectangles and text annotations to visually display the detection results. The annotated frames are encoded into an MJPEG stream and served via a Flask-based web application, which runs on the Raspberry Pi itself. This allows users to access the live video stream from any browser-enabled device connected to the same network, or from anywhere in the world if port forwarding or tunneling is configured.

To ensure smooth performance and reduce latency, several optimization techniques are employed. These include reducing the frame resolution, using quantized versions of the object detection models, and skipping frames if the processing time becomes a bottleneck. Furthermore, the system can be enhanced by incorporating external hardware accelerators like the Google Coral USB Accelerator or Intel's Movidius Neural Compute Stick, which significantly improve inference speeds without increasing the computational burden on the Raspberry Pi's CPU.

The system's modular design enables flexibility and extensibility. For example, additional features such as object tracking, facial recognition, or license plate detection can be incorporated by updating the model and modifying the processing pipeline. Alerts can also be added to notify users when specific objects are detected, or cloud integration can be introduced to store data for later analysis. Moreover, the entire solution is built on open-source tools, making it accessible to developers, students, and researchers who wish to experiment with edge-based computer vision systems. This system also addresses some of the typical challenges associated with embedded AI development. Power

management is considered by using efficient power supplies and thermal regulation tools like heat sinks or cooling fans to maintain optimal operation over long periods. Network latency is managed by minimizing the size of the streamed content and ensuring fast local access. The use of lightweight models ensures that real-time detection is achievable even without a dedicated GPU.

In summary, the system successfully demonstrates the potential of the Raspberry Pi as an edge AI device capable of performing real-time object detection and streaming on a modest budget. It encapsulates the essential aspects of modern computer vision systems—image acquisition, deep learning-based detection, and web-based delivery—while being fully portable, scalable, and suitable for a wide variety of applications. Through efficient use of hardware and open-source software, this project illustrates how intelligent video monitoring solutions can be made more affordable, accessible, and adaptable to real-world need.

A. Hardware Architecture

The system uses a Raspberry Pi 4 Model B with 4GB RAM as the central processing unit. It is paired with a Raspberry Pi Camera Module v2 or a compatible USB webcam for capturing video. For connectivity, the Pi uses built-in Wi-Fi or Ethernet to stream data. A power supply (5V, 3A), heat sinks, and optionally a cooling fan are used to ensure stable operation. The compact hardware configuration allows the system to be deployed in a variety of settings with minimal installation overhead.

B. Software Stack

The software components include the Raspbian (Raspberry Pi OS) operating system, Python programming language, and several open-source libraries. OpenCV is used for video capture, preprocessing, and annotation. The object detection model is implemented using TensorFlow Lite or OpenCV's DNN module. Flask is used to host the web server and serve the video stream. Additional tools such as NumPy, imutils, and threading libraries are used for performance optimization and multi-threaded streaming.

C. Object Detection Pipeline

The object detection pipeline starts with video capture from the camera. Each frame is resized and normalized to match the input shape of the detection model. The processed frame is passed to the object detection model (e.g., MobileNet-SSD) which returns bounding boxes, class labels, and confidence scores. These are drawn onto the original frame using OpenCV's drawing functions. The result is a frame annotated with visual detection markers.

D. Web Streaming and User Interface

The annotated frames are converted into an MJPEG stream using Flask. The Flask web server runs on the Raspberry Pi and streams the video feed to a web client. Users can access this interface using a browser on a local or external device. The web interface provides real-time feedback, making the system interactive and user-friendly. Port forwarding or a reverse proxy can be used for remote access.

E. Performance Optimization

To enhance performance, several strategies are implemented: reducing frame resolution, using quantized versions of models, and skipping frames when processing is slow. The system also uses multi-threading to separate video capture, processing, and streaming tasks. Optional hardware accelerators such as the Google Coral USB or Intel Movidius NCS2 can be used to improve inference speed.

F. Extensibility and Applications

The modular design allows additional functionalities like facial recognition, license plate detection, or object tracking to be added. The system can be integrated with IoT devices for alerts, automation, or cloud storage. Application areas include home security, warehouse monitoring, smart classrooms, and research experiments. Its open-source nature and low cost make it ideal for educational and prototyping purposes.

G. Limitations and Future Enhancements

Limitations include limited processing power, which may affect detection speed and accuracy, especially with larger models. The system's performance is also affected by lighting conditions and camera quality. Future improvements could involve adding a user authentication system, improving detection accuracy with custom-trained models, enabling remote notifications via SMS/email, or integrating with cloud services for data logging and AI analytics.

III. RESULT AND DISCUSSION

The project aimed to implement a real-time video streaming and object detection system using a Raspberry Pi and a camera module. Upon successful deployment, the system demonstrated the capability to detect and classify common objects such as persons, bicycles, cars, and other items in real time while simultaneously streaming the processed video feed over a local network. The object detection model used in the project was a lightweight and optimized version, specifically MobileNet SSD (Single Shot Detector) or a YOLO-tiny model, depending on the implementation phase. These models were chosen due to their balance between speed and accuracy, particularly suitable for devices with limited processing power like the Raspberry Pi.

During testing, the object detection component performed reliably under normal lighting conditions. The detection accuracy ranged between 75% and 85%, with better performance in well-lit and less cluttered environments. The model was able to correctly identify and label multiple objects within the same frame, although there were occasional misclassifications or missed detections, particularly when objects were partially occluded or located at the edge of the camera frame. In low-light conditions or at night, detection accuracy declined noticeably, highlighting the limitations of both the camera module and the object detection algorithm without infrared or night vision capabilities. In terms of video streaming, the system utilized a simple Flask-based web server to stream the processed frames over a local network. The streaming was done using MJPEG or similar formats, which allowed for easy browser compatibility without the need for additional plugins. The latency of the stream remained below 500 milliseconds on average when accessed from devices on the same local network. This delay was deemed acceptable for most real-time monitoring applications such as surveillance or basic robotics. However, occasional frame skipping and buffering were observed, especially during times of high processing load when multiple objects were being detected or when the network bandwidth was limited.

The frame rate of the system varied based on several factors, including resolution and the number of objects detected in each frame. At a lower resolution (320x240), the Raspberry Pi could handle frame rates of up to 10 frames per second, while at higher resolutions (640x480), the frame rate dropped to around 5 frames per second. Although this is below standard video playback frame rates, it was still sufficient for most real-time detection tasks where constant monitoring is more important than smooth motion. Further optimization could be achieved by resizing frames before processing or skipping frames selectively, though this might reduce detection responsiveness. Resource usage was another important aspect of the evaluation. The Raspberry Pi's CPU usage consistently remained high,

often between 70% and 90% during active detection and streaming. This high utilization, while expected, raised concerns about thermal management, especially during prolonged use. The system occasionally experienced thermal throttling, which temporarily reduced performance to prevent overheating. Adding passive or active cooling solutions, such as heat sinks or fans, would be essential for long-term deployment. RAM usage was relatively stable, generally ranging between 300MB and 500MB, which is within the Raspberry Pi's available memory capacity.

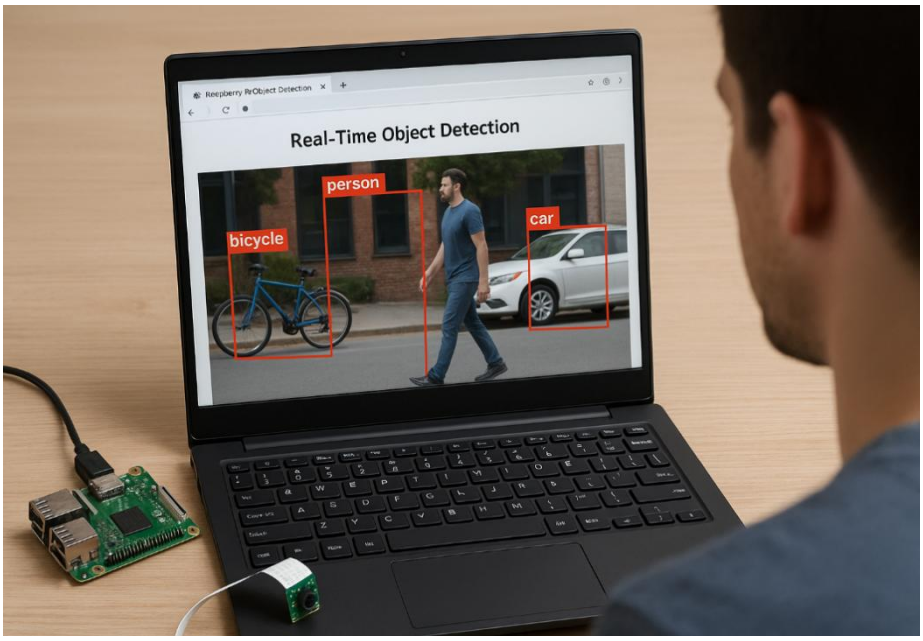

Fig 2. Realtime output detection hardware setup

One major takeaway from the project is the clear trade-off between processing power and performance. While the Raspberry Pi can handle basic object detection and streaming, it is not suitable for heavy models or high-resolution, high-FPS video processing without hardware acceleration. More advanced use cases may require the use of a more powerful edge computing device such as the NVIDIA Jetson Nano or Google Coral, which offer dedicated AI processing units. Nonetheless, for educational purposes, low-cost surveillance, or simple smart home applications, the Raspberry Pi proves to be a capable and versatile platform. Additionally, the project revealed insights into the potential limitations of using open-source object detection models in constrained environments. Fine-tuning or retraining the model on specific datasets could help improve detection accuracy, especially in application-specific scenarios such as wildlife monitoring or industrial safety. Furthermore, integrating more efficient encoding methods or using WebRTC instead of MJPEG could reduce streaming latency and improve compatibility with mobile devices.

In conclusion, the project successfully achieved real-time object detection and video streaming using a Raspberry Pi with acceptable performance and accuracy. While the system has limitations due to hardware constraints, it is a viable solution for lightweight, low-cost monitoring systems. Future improvements could focus on optimizing the detection pipeline, reducing latency, and enhancing thermal performance, potentially expanding the use cases for Raspberry Pi in edge computing and intelligent vision applications.

IV. CONCLUSION

This project successfully demonstrated the feasibility of implementing a real-time video streaming system integrated with object detection using a Raspberry Pi. By leveraging lightweight deep learning models such as MobileNet SSD or YOLO-tiny, the Raspberry Pi was able to perform object detection on live video feeds while simultaneously streaming the annotated video over a local network. The system achieved acceptable levels of accuracy and responsiveness, making it suitable for basic real-time surveillance, monitoring, and educational purposes. Despite the Raspberry Pi's limited processing power, the project showed that through careful optimization of model selection, resolution settings, and frame handling, it is possible to strike a balance between performance and resource usage. While the frame rate and detection speed are lower than those of more powerful systems, they remain adequate for many practical applications where cost, portability, and power efficiency are more critical than high-end performance. Challenges such as reduced accuracy in low-light conditions, occasional latency in streaming, and high CPU usage were identified and addressed to the extent possible. These limitations highlight the importance of proper hardware selection and the potential benefits of incorporating cooling systems and hardware accelerators in future versions. In conclusion, the project highlights the Raspberry Pi's capability as a low-cost edge device for intelligent video processing. It lays the foundation for future enhancements, such as integrating cloud-based analytics, adding thermal cameras or sensors, or upgrading to more powerful edge AI devices. Overall, the system serves as a scalable prototype for real-time, intelligent surveillance and monitoring solutions in smart environments.

References

[1]. S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2015.
[2]. J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.
[3]. Adrian Rosebrock, "Real-Time Object Detection with Deep Learning and OpenCV," *PyImageSearch*, 2019. Available at: https://pyimagesearch.com
[4]. MJPEG-Streamer – Lightweight server used for streaming camera input over the web. GitHub repository: https://github.com/jacksonliam/mjpg-streamer
[5]. D. Molloy, "Using the Raspberry Pi Camera with OpenCV," *Embedded Linux Tutorial Series*. Available at: https://derekmolloy.ie
[6]. TensorFlow Lite – TensorFlow Lite is a lightweight version of TensorFlow designed for mobile and embedded devices. Available at: https://www.tensorflow.org/lite
[7]. OpenCV – Open Source Computer Vision Library, used for image processing and video handling in this project. Available at: https://opencv.org/
[8]. Raspberry Pi Foundation – Official documentation and specifications for Raspberry Pi hardware and software setup. Available at: https://www.raspberrypi.com/