Object Detection Using ESP32 CAM in Realtime

Lalitha Boddeda[1], A.Ramana Babu[2], M.Lalitha[3], S. Prakash[4], K. Vinay[5], N. Karthik[6]
[1,2]*Assistant Professor, Department of ECE, Avanthi Institute of Engineering & Technology, Narsipatnam, Anakapalle.*
[3, 4, 5,6] *Student Department of ECE, Avanthi Institute of Engineering & Technology, Narsipatnam, Anakapalle*

***Abstract* — The ESP32-CAM based Real-Time Object Detection System is a low-cost embedded vision solution designed for smart surveillance and monitoring applications. The system utilizes the ESP32-CAM module, which integrates a Wi-Fi enabled microcontroller with an OV2640 camera. It captures real-time images and streams video over a wireless network. The device processes visual data using embedded programming and supports integration with machine learning models. Lightweight object detection algorithms are implemented to identify objects in real time. The system is programmed using Arduino IDE and C/C++ language. It can transmit captured data to a web server or cloud platform for remote monitoring. The compact design makes it suitable for IoT-based applications. It consumes low power and operates efficiently in real-time environments. The system supports image capture, video streaming, and motion detection features. It can be applied in security systems, smart homes, and industrial monitoring [2] [1]**
The wireless capability eliminates the need for complex wiring. The project demonstrates the integration of embedded systems with artificial intelligence. It provides a scalable and cost-effective smart vision solution. Overall, the ESP32-CAM enables real-time intelligent monitoring in modern IoT applications [2]

***Keywords*— ESP32-CAM, Object Detection, Internet of Things (IoT), Deep Learning, Tensor Flow, Computer Vision, CNN, Real-Time Monitoring, Embedded Systems, Edge Computing, Smart Surveillance. [7]**

I. INTRODUCTION

Object detection is one of the most important applications of computer vision and artificial intelligence. It enables machines to automatically identify and locate objects within images or video streams. In recent years, embedded systems combined with artificial intelligence have made it possible to perform intelligent tasks using low-cost hardware devices. The ESP32-CAM module is a compact and powerful microcontroller equipped with a built-in camera, WiFi connectivity, and sufficient processing capability for image acquisition and IoT applications. [2]

The objective of this project is to design and develop a real-time object detection system using the ESP32-CAM module. The system captures live video frames through the camera module and streams them over a WiFi network. These frames are processed using a computer vision model implemented with Tensor Flow, which detects objects such as persons, cars, cats, and dogs. Once an object is detected, the result is transmitted back to the ESP32-CAM and displayed on a 16×2 LCD module. The system integrates hardware components such as the ESP32-CAM microcontroller, I2C LCD display, power supply, and connecting wires. The software part includes embedded programming using Arduino IDE and object detection using Python with TensorFlow and OpenCV libraries. The system also uses a web interface developed with Flask to display the live camera feed and detected objects. [7] [2]

This project demonstrates how IoT devices can be combined with artificial intelligence to build smart monitoring systems. Such systems have applications in security surveillance, smart homes, traffic monitoring, wildlife observation, and industrial automation. [2]

The proposed solution is cost-effective, portable, and easy to deploy compared to traditional computer-based vision systems. By combining embedded hardware with machine learning algorithms, this project provides a practical approach to implementing real-time object detection in everyday environments. [1]

This project focuses on developing a real-time object detection system using the ESP32-CAM module. The system captures live images using the built-in camera and processes them to detect objects

such as humans, cats, dogs, and cars. A Convolutional Neural Network (CNN) based model is used to recognize and classify objects by learning visual features from a trained dataset. [2] [1]

II. LITERATURE SURVEY

2.1 TRADITIONAL METHODS OF OBJECT DETECTION

In earlier years, object detection was primarily performed using classical image processing techniques. These methods relied on handcrafted features and mathematical models to detect patterns in images. One commonly used technique was the Haar Cascade classifier, which detects objects by analyzing Haar-like features extracted from images. This method was widely used for face detection in early surveillance systems. [1]

Another popular traditional approach involved Histogram of Oriented Gradients (HOG). This technique analyzes gradient orientation in localized portions of an image to identify object shapes. HOG was commonly used for pedestrian detection in video surveillance applications. These methods were computationally efficient and suitable for low-power devices.

Traditional detection techniques often involved multiple steps such as image pre-processing, feature extraction, and classification. Algorithms such as edge detection, thresholding, and template matching were used to identify objects within images. While these methods were effective for simple applications, they struggled to perform well in complex environments.

The major limitation of traditional computer vision techniques is their inability to adapt to variations in lighting conditions, object orientation, and background clutter. They require manual feature engineering and cannot easily generalize to new object categories. Due to these limitations, researchers began exploring machine learning and deep learning approaches to improve object detection accuracy and robustness [1]

2.2 INTERNET OF THINGS (IoT) IN SMART MONITORING SYSTEM

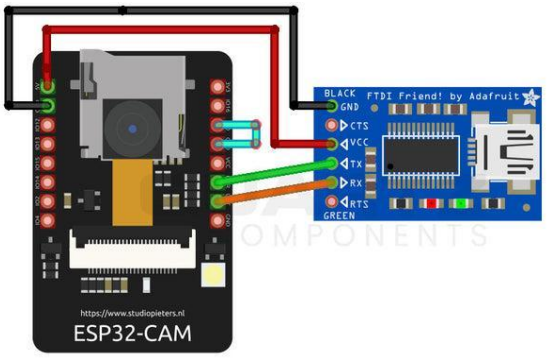


Figure 2.1: ESP32-CAM Module Architecture

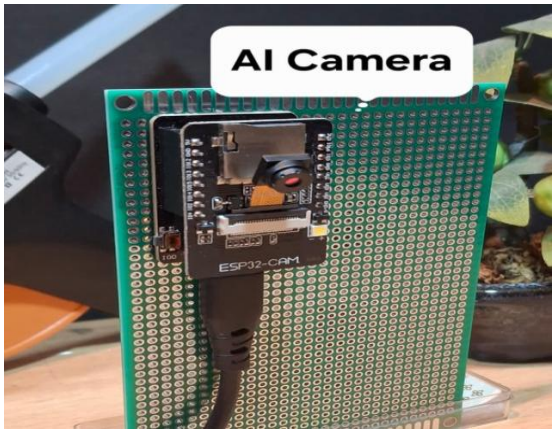


Figure 2.2: AI Camera for Smart Monitoring

The Internet of Things (IoT) has revolutionized the way devices communicate and interact with each other. IoT enables physical devices such as sensors, cameras, and microcontrollers to connect to the internet and exchange data in real time. In recent years, IoT has been widely used in smart monitoring systems, surveillance applications, and industrial automation. [2]

Embedded microcontrollers equipped with wireless connectivity allow remote monitoring of environments without the need for expensive infrastructure. Devices such as camera modules and sensors can capture data and transmit it to remote servers for processing and analysis.

In surveillance applications, IoT-based camera systems enable real-time monitoring of public spaces, buildings, and industrial facilities. These systems provide continuous video streaming and allow users to access the video feed remotely through web interfaces or mobile applications.The integration of IoT with computer vision technologies further enhances the capabilities of monitoring systems. By combining cameras with artificial intelligence algorithms, it becomes possible to automatically detect objects, track movements, and generate alerts when specific events occur. [2]

Microcontrollers such as ESP32 have become popular in IoT applications due to their built-in WiFi capability, low power consumption, and compact design. These devices allow developers to build smart camera systems that can capture images, stream video, and communicate with cloud-based or local processing systems. [2]

The Internet of Things (IoT) refers to a network of connected devices that communicate with each other through the internet to collect, send, and analyze data. Insmart monitoring systems, IoT enables real-time observation and control of environments using sensors, cameras, and embedded devices. These devices continuously collect information and transmit it to cloud platforms or monitoring applications for analysis and decision-making. [2]

In many modern applications, IoT devices such as the ESP32-CAM are used to capture images or environmental data and send them to a server or monitoring dashboard. This allows users to monitor locations remotely and receive alerts when unusual events occur. IoT-based monitoring systems are widely used in areas such as smart homes, security surveillance, traffic monitoring, industrial automation, and environmental monitoring. [2]

By integrating Artificial Intelligence and Computer Vision with IoT devices, systems can automatically detect objects, recognize patterns, and respond intelligently. This combination improves efficiency, reduces human effort, and enables faster decision-making. As a result, IoT plays a crucial role in developing intelligent and automated monitoring systems for real-world applications. [2]

2.3 DEEP LEARNING BASED OBJECT DETECTION

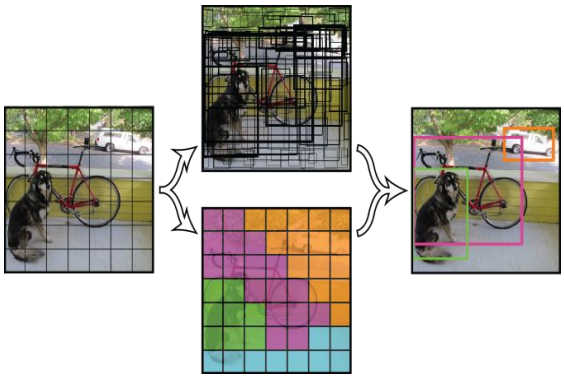


Figure 2.3: Unified Deep Learning Based Object Detection Method

Recent advancements in deep learning have significantly improved the performance of object detection systems. Deep learning models use artificial neural networks to automatically learn complex features from large datasets. These models eliminate the need for manual feature engineering and provide higher accuracy compared to traditional techniques. [1]

One of the widely used deep learning models for object detection is the Single Shot MultiBox Detector (SSD). This model performs object localization and classification in a single forward pass of the neural network. The SSD architecture is efficient and suitable for real-time applications. [6] [1]

Another popular model is YOLO (You Only Look Once), which processes the entire image at once and predicts bounding boxes and class probabilities simultaneously. YOLO is known for its high speed and abilityto perform real-time object detection in video streams. [5] [1]

Tensor Flow provides several pre-trained object detection models such as SSD MobileNet, Faster R-CNN, and EfficientDet. These models are trained on large datasets like COCO and ImageNet, enabling them to recognize a wide range of objects. [7] [6]

Deep learning-based detection systems are capable of handling complex scenes, varying lighting conditions, and multiple object categories. These capabilities make them suitable for applications such as autonomous vehicles, security surveillance, robotics, and smart cities.

However, deep learning models typically require significant computational resources, which can be challenging for low-power embedded devices. Therefore, researchers have explored hybrid systems where image capture is performed by embedded hardware while processing is carried out on more powerful computing systems.

2.4 Research Gaps

Despite significant progress in object detection technologies, several challenges still remain in implementing efficient real-time detection systems using embedded hardware. [1]

Many existing systems rely on high-performance computers or cloud-based servers to perform image processing and object detection tasks. This increases the cost of deployment and introduces latency due to network communication delays. Such systems may not be suitable for real-time applications where immediate responses are required. [1]

Another limitation is the power consumption and hardware requirements of deep learning models.

Running complex neural networks directly on microcontrollers is often difficult due to limited memory and processing capability.In addition, many traditional surveillance systems only provide video streaming without intelligent analysis. This means that human operators are still required to monitor video feeds continuously, which reduces efficiency and increases the chance of missing critical events.There is also a lack of affordable and portable object detection systems that combine embedded cameras, wireless communication, and artificial intelligence.

Therefore, there is a need for a low-cost, energy-efficient, and intelligent object detection system that can capture images using embedded hardware while leveraging powerful machine learning algorithms for object recognition. [1]

The proposed system addresses these gaps by combining an ESP32-CAM module for image acquisition with a Tensor Flow-based object detection model running on a computer. This hybrid architecture enables real-time object detection while maintaining low hardware cost and power consumption. [7] [2]

Key challenges in existing systems include: High cost due to advanced hardware like LiDAR

- Complex algorithms requiring high computational power
- Lack of simple cloud-based mission management
- Limited real-time communication and monitoring
- Absence of GPS navigation in low-cost systems

These gaps highlight the need for a low-cost, IoT-enabled autonomous rover.

2.5 EMBEDDED VISION SYSTEMS [2]

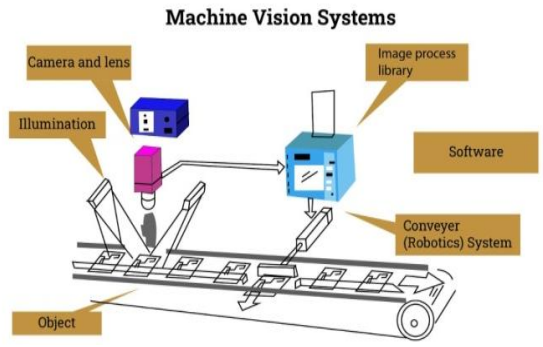


Figure 2.4: Embedded Machine Vision System

GPS enables accurate outdoor navigation using satellite signals. Modules like NEO-6M provide latitude and longitude data.Distance to waypoints is calculated using formulas such as the Haversine formula. GPS allows operation without predefined paths but may face signal issues in obstructed areas. In this project, GPS data guides the rover toward target locations in real time.

2.6 EDGE COMPUTING IN OBJECT DETECTION

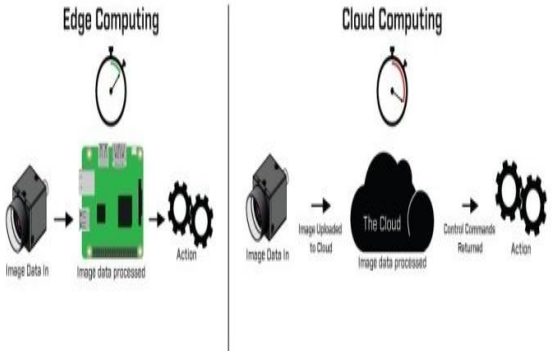


Figure 2.5: Edge Computing Based Machine Vision System

Edge computing is an emerging technology that processes data closer to the source of generation rather than sending all data to centralized cloud servers. In computer vision applications, edge computing allows devices such as cameras and embedded processors to perform partial or complete data processing locally.

One of the main advantages of edge computing is reduced latency. Since data is processed locally, the system can respond quickly to detected events without waiting for cloud processing. This is particularly important in real-time applications such as surveillance systems, autonomous vehicles, and industrial monitoring. Another advantage is improved privacy and security. Instead of transmitting raw video data to remote servers, only processed information or detection results are sent across the network. This reduces bandwidth consumption and protects sensitive data.In object detection systems, edge computing allows embedded devices to capture and preprocess images while more complex processing can be handled by nearby computing systems. The proposed project follows this hybrid approach, where the ESP32-CAM captures video frames and a computer running Tensor Flow performs object detection. This architecture provides a balance between computational efficiency and real-time performance, making it suitable for low-cost embedded AI

applications.
[7] [2]

2.7 SUMMARY OF LITERATURE SURVEY

The literature survey highlights the evolution of object detection technologies from traditional image processing techniques to modern deep learning-based approaches. Early systems relied on handcrafted feature extraction methods such as Haar cascades and Histogram of Oriented Gradients. While these methods were computationally efficient, they lacked robustness and accuracy in complex environments. [1]

With the advancement of artificial intelligence and machine learning, deep learning models such as SSD MobileNet and YOLO have significantly improved the performance of object detection systems. These models can automatically learn complex visual features and detect multiple objects within an image with high accuracy. [5] [6]

The integration of Internet of Things (IoT) technology has further enhanced the capabilities of monitoring systems by enabling wireless communication between embedded devices and computing systems. IoT-based camera modules allow real-timeimage streaming and remote monitoring through web interfaces. [2]

Embedded vision systems and edge computing technologies have also played a crucial role in enabling intelligent processing on compact and energy-efficient hardware platforms. These technologies allow visual data to be captured and processed closer to the source, reducing latency and improving system efficiency. Despite these advancements, many existing systems still rely on expensive hardware or cloud-based processing. Therefore, there is a need for cost-effective and portable object detection solutions. [1]

The proposed system addresses these challenges by combining the ESP32-CAM module for image acquisition with a deep learning-based object detection model implemented using TensorFlow. This hybrid architecture enables real-time object detection while maintaining low hardware cost and efficient system performance. [7]

III. BLOCK DIAGRAM

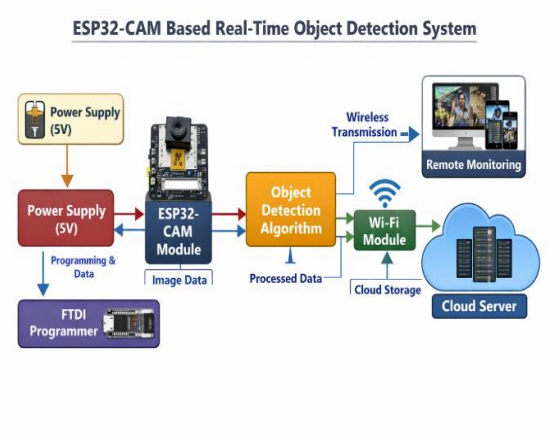


Figure 3.1: Block Diagram of ESP32-CAM Based Real-Time Object Detection System

A Data Flow Diagram (DFD) represents the flow of data within the system. It shows how data moves from input devices to processing components and finally to output devices.In the proposed system, the ESP32-CAM captures images and sends them to the Python server through a WiFi network. The server processes the received frames using the TensorFlow object detection model. The detection results are displayed on the web interface and also sent back to the ESP32-CAM. [7] [2].The ESP32-CAM receives the detected object information and displays it on the LCD module. This continuous loop allows the system to perform real-time monitoring and object recognition. [2]. In this project, the system is designed to perform real-time object detection using an embedded camera module connected to a wireless network. The ESP32-CAM module captures video frames and streams them through WiFi. These frames are processed using a computer vision model implemented with Tensor Flow. The detected objects are displayed on a web interface and also shown on a 16×2 LCD module connected to the ESP32-CAM. [2] [1]

3.1 Data Flow Steps

1. Whether the proposed system is practical and achievable within the available resources. It evaluates the technical, operational, and economic aspects of the project.Camera captures image frames
2. ESP32-CAM streams frames via WiFi
3. Python server receives the frames
4. Tensor Flow model performs object detection

5. Detected objects are displayed on the web interface
6. Detection results are sent back to ESP32-CAM

3.1.1 Technical Feasibility

Technical feasibility refers to whether the required hardware and software technologies are available to implement the proposed system. The ESP32-CAM microcontroller provides built-in WiFi connectivity and camera support, making it suitable for image capture and streaming applications. The system also uses machine learning libraries such as Tensor Flow and OpenCV to perform object detection. [2] [1]

These technologies are widely available and supported by large developer communities. The Arduino IDE provides a convenient environment for programming the ESP32-CAM module, while Python frameworks allow efficient implementation of computer vision algorithms. [2], Since all required components such as microcontrollers, camera modules, LCD displays, and development tools are readily available, the technical implementation of the system is feasible.

3.2 FUNCTIONAL REQUIREMENTS

Functional requirements describe the specific operations and features that the system must perform. These requirements define how the system processes inputs and produces outputs.

3.2.1 System Inputs

The input to the system mainly consists of video frames captured by the ESP32-CAM camera module. These frames are streamed over a wireless network to a computer system where object detection processing takes place. [2] [1]

The user can also provide input through the web interface by pressing the capture button to initiate object detection. This triggers the processing of the current frame. [1]

3.2.2 System Processing

The processing stage involves analyzing the captured frames using a deep learning model. The system uses the SSD MobileNet object detection model implemented through Tensor Flow. [7] [6]

Each captured frame is resized and converted into the required format before being passed to the machine learning model. The model analyzes the image and identifies objects based on learned patterns.Bounding boxes are generated around detected objects, and labels such as person, car, cat, or dog are assigned to them.

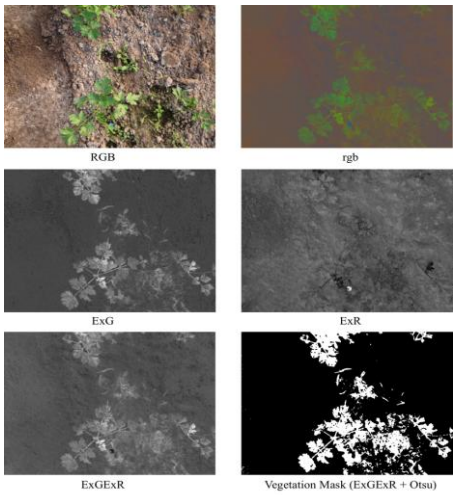
3.2.3 System Outputs

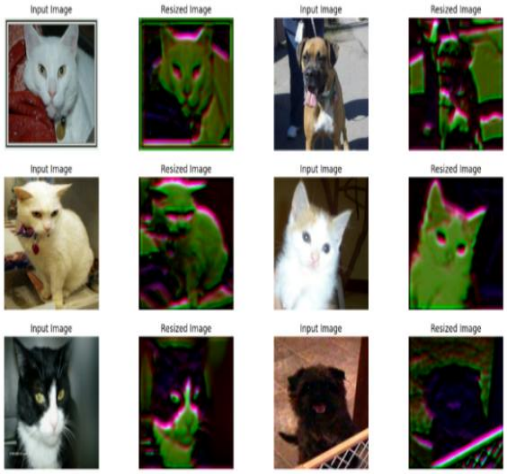
The system provides output in multiple formats to improve usability. Detected objects are displayed on a web interface along with the captured image and detection results.The system also sends the detected object names back to the ESP32-CAM module through HTTP communication. The ESP32-CAM then displays the detected object name on the 16×2 LCD module. [2]

1. Image Pre-processing Requirement

The system shall pre-process each captured frame before passing it to the CNN model. This includes resizing the image to the required input dimension (e.g., 96×96 or 160×160 pixels), normalizing pixel values, and converting the image format if necessary. Pre-processing must be completed quickly to maintain real-time performance. [1]

The pre-processing stage usually begins with image acquisition, where an image is captured using a camera or sensor, such as the ESP32-CAM module used in real-time object detection systems. After capturing the image, it is often resized to a standard resolution so that it matches the input size required by machine learning models. Resizing also reduces computational complexity and speeds up processing. Another common step is noise reduction, where unwanted distortions in the image are removed using filtering techniques like Gaussian or median filters. This helps in [2] [1]





3. Bounding Box and Output Display Requirement

The system shall generate bounding box coordinates around detected objects and display them on the live video stream. It must output detection results via Serial Monitor and host a web server for remote access. The detected object label and confidence score shall be visible in the output interface.

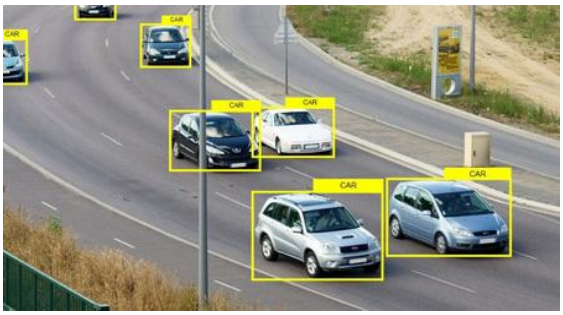


Figure 3.4: CNN-Based Image Segmentation and Detection

4. Real-Time Processing Requirement

The system shall process frames continuously without manual intervention. It must maintain acceptable frame rate (FPS) suitable for embedded applications. The inference process must operate within the memory and processing limitations of the ESP32-CAM module. [2]

IV. OUTPUTS

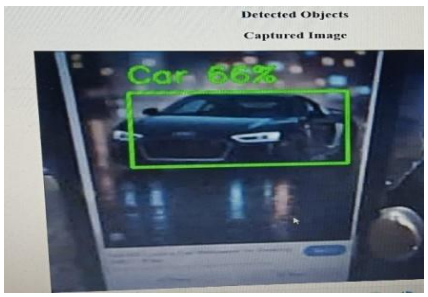


Figure 4.1(a): Input Image Sample

Input Result

Figure 4.2(b): Object Detection Output Result

V. CONCLUSION

The project “Design and Development of Object Detection using ESP32-CAM in Real Time” successfully demonstrates the integration of embedded systems with computer vision and artificial intelligence technologies. The system uses the ESP32-CAM module to capture images and stream video frames through a WiFi network. These frames are processed using a deep learning-based object detection model implemented with Tensor Flow to identify objects such as persons, cars, cats, and dogs. [2]

The developed system displays detection results through a web interface with highlighted bounding boxes and object labels. Additionally, the detected object names are transmitted back to the ESP32-CAM module and displayed on a 16×2 LCD module. This combination of hardware and software provides a simple and efficient real-time object detection system. [2]

Overall, the project provides a low-cost and practical solution for smart monitoring applications. The system can be further enhanced by adding more object categories, implementing edge AI processing directly on embedded devices, and integrating cloud-based monitoring systems for advanced applications.

REFERENCES

[1] X. Wang, A. Shrivastava, and A. Gupta, “A-Fast-RCNN: Hard Positive Generation via Adversary for Object Detection,” in Proc.

IEEE Conf. Computer Vision and Pattern Recognition (CVPR), 2017, pp. 2606–2615. [1]

[2] Veena A.,Vasanthamma H., Heenakousar M. Makandar, and Uppar Gangadevi, "Real Time Object Detection Using TensorFlow," Journal of Emerging Technologies and Innovative Research (JETIR), 2019. [7]

[3] A. Periyaswamy and K. Tejaswi, "Real Time Object Detection Using Raspberry Pi," 2020.

[4] V. S. S. R. Prakash, D. Lakshman Sai, and M. Saraswathi, "Object Detection with Audio Feedback," International Journal of Creative Research Thoughts (IJCRT), 2023.

[5] J. Redmon and S. Divvala, "You Only Look Once: Unified, Real-Time Object Detection," in Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR), 2016.

[6] W. Liu et al., "SSD: Single Shot MultiBox Detector," in Proc. European Conf. Computer Vision (ECCV), 2016. [6]

[7] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," in Advances in Neural Information Processing Systems, 2015, pp. 91–99. [1]

[8] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," in Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR), 2016, pp. 779–788.

[9] J. Redmon and A. Farhadi, "YOLO9000: Better, Faster, Stronger," in Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR), 2017, pp. 7263–7271. [5]

[10] J. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement," arXiv preprint arXiv:1804.02767, 2018. [5]

[11] R. Bharti, K. Bhadane, P. Bhadane, and A. Gadhe, "Persons Detection and Recognition for Blind Assistance," International Research Journal of Engineering and Technology (IRJET), vol. 6, 2019.

[12] J. Redmon et al., "You Only Look Once: Unified Real-Time Object Detection," IEEE Paper.