

“Real time object detection”

Prof. Shobha S Biradar¹, Sunil²

¹Professor, Master of Computer Application, VTU, Kalaburagi, Karnataka, India

²Student, Master of Computer Application, VTU, Kalaburagi, Karnataka, India

ABSTRACT- Real-time object detection is a rapidly evolving field within Artificial Intelligence (AI) and Machine Learning (ML) that focuses on the automatic identification and localization of objects in images or video streams with minimal delay. Unlike traditional image classification, which only determines the presence of an object, object detection provides both classification and spatial information using bounding boxes. Recent advancements in deep learning, particularly Convolution Neural Networks (CNNs) and architectures such as YOLO (You Only Look Once), SSD (Single Shot Multibox Detector), and Faster R-CNN, have significantly improved the accuracy and speed of detection.

Keywords: Real time object detection, Deep learning, Real-time object detection, Computer vision, object detection.

1. INTRODUCTION

This unified detection approach frames the task as a single network that directly regresses object bounding boxes and class probabilities from full images, enabling extremely fast end-to-end inference suitable for real-time video. It trades some localization finesse for speed but set the standard for practical, deployable detectors on live streams. Because the model is compact and trains end-to-end, it became a popular baseline for edge and camera-side applications where throughput is critical. Its simplicity also made export and engineering tooling easier, accelerating adoption. Use this family as the first production candidate when latency and simplicity are top priorities.

This single-shot detector predicts object boxes and classes from multi-scale feature maps, handling different object sizes in one pass and offering an attractive speed/accuracy tradeoff. It demonstrated that proposal-free detectors can rival two-stage systems while remaining faster—useful when you need more accuracy than the smallest real-time models but still require high FPS. The multi-scale default-box mechanism is widely reused in mobile and embedded detectors. For mid-tier hardware, SSD is a pragmatic option balancing precision and throughput.

2. PROBLEM STATEMENT

The primary challenge lies in designing a system that can accurately detect and classify multiple objects in a video stream while maintaining real-time performance. Achieving this requires balancing speed, accuracy, and computational

efficiency. Furthermore, real-world environments often present additional difficulties such as variations in lighting, object occlusion, background clutter, and camera noise, which make detection tasks more complex. Therefore, the problem addressed in this project is the development of a reliable real-time object detection system that leverages advanced AI and ML techniques, particularly deep learning models, to overcome the shortcomings of traditional approaches and provide accurate, fast, and efficient object detection for real-world applications.

3. OBJECTIVES

The main objective of this study is to design and implement a real-time object detection system that can accurately identify and classify multiple objects in live video streams using Artificial Intelligence (AI) and Machine Learning (ML) techniques. To achieve this, the study is guided by the following specific objectives:

4. METHODOLOGY USED

The methodology of this study outlines the systematic approach followed to design and implement the real-time object detection system

1) Model Selection: Choose a suitable pre-trained deep learning model such as YOLOv8 or SSD for real-time performance.

2) System Design: Develop a pipeline where video frames are captured from a webcam or video source.

3) Implementation: Integrate the chosen model into the system using Python and OpenCV.

4) Testing and Evaluation: Test the system under different conditions such as lighting variations, moving objects, and occlusions.

5. LITERATURE SURVEY

Mobile / lightweight detectors (various, 2016–2023) — Mobile Net-SSD, Tiny-YOLO, and Efficient Det-lite variants show how architecture choices (depth wise separable convs, smaller feature maps) minimize compute and memory while preserving usable accuracy. These models are tailored for CPU/mobile/NPU inference where full-size detectors are impractical. Use them for on-device, battery-sensitive applications.

Bewley et al. (2016) — Introduced SORT, a minimal-latency tracker using Kalman filters and Hungarian assignment to link detections across frames, achieving high throughput for multi-object tracking. SORT’s simplicity makes it a popular baseline in real-time systems that need trajectories without heavy compute. It’s commonly used with fast detectors to create scalable camera pipelines.

Wojke et al. (2017) — Extended SORT with learned appearance embeddings (DeepSORT) to reduce identity switches during occlusion and reappearance, combining motion and appearance cues. Deep SORT is widely used where ID continuity matters (counting, re-id across cameras); it balances robustness with real-time constraints by offloading embedding computation to a lightweight CNN.

Lin et al. (2014) — The COCO dataset provided a large-scale, richly-annotated benchmark with instance segmentation and dense scenes, catalyzing modern detector evaluation and architecture development. COCO’s standardized metrics (mAP at multiple IoU thresholds) and diverse images are the de-facto evaluation baseline for detection research. Train and evaluate on COCO for broad generalization and standardized comparisons.

6. SYSTEM DESIGN

The real-time object detection system is designed as a standalone application that integrates computer vision techniques with deep learning models to detect and classify objects in live video streams. The system fits into the domain of AI-powered intelligent monitoring solutions, where it acts as a core module that can later be extended into larger applications such as smart surveillance systems, autonomous vehicles, or industrial automation platforms.

System Architecture Overview:

Input Subsystem: Captures live video feed from a webcam, CCTV, or pre-recorded video file.

Processing Subsystem: Uses a deeplearning model (e.g., YOLO, SSD, Faster R-CNN) to detect and classify objects.

Output Subsystem: Displays processed frames with bounding boxes, labels, and confidence scores in real-time.

User Interaction Subsystem: Allows users to start, stop, and exit the detection process with minimal effort.

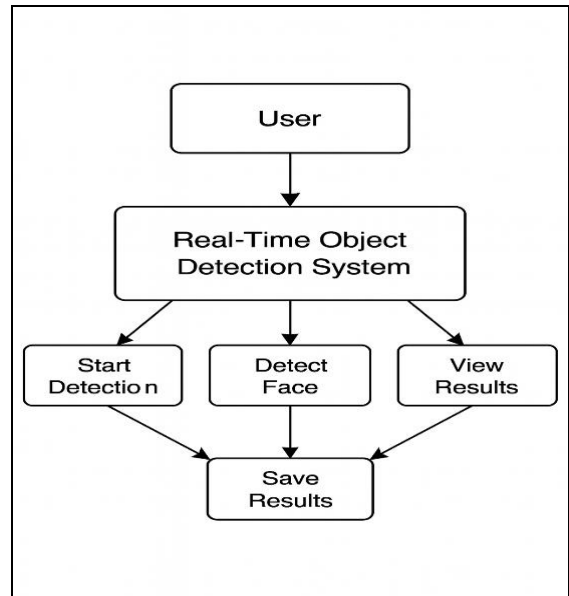
System Context:

The system interacts with hardware components (camera, GPU/CPU, display monitor) and software libraries (OpenCV, PyTorch/TensorFlow, YOLOv8).

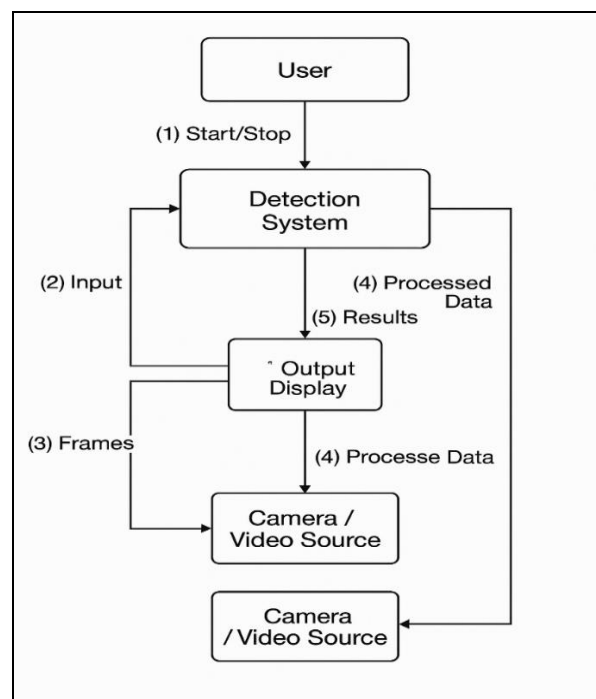
It relies on pre-trained deep learning models, which can be replaced or retrained for domain-specific applications. The

output is presented visually, but may also be extended to generate logs or alerts in advanced versions.

7. DETAILED DESIGN



Collaboration Diagrams:



8. SCREENSHOTS

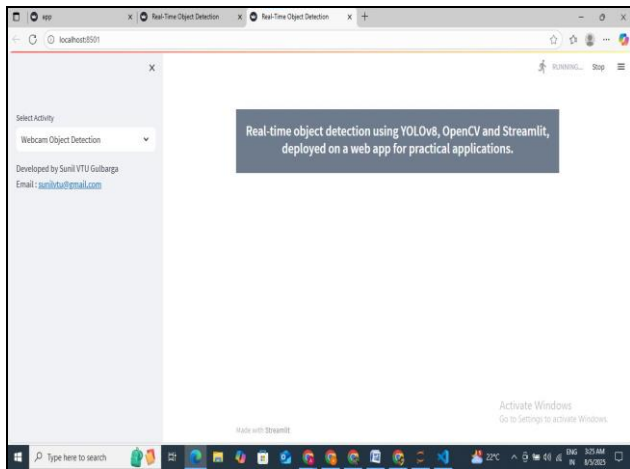


Figure 7.2: Home page

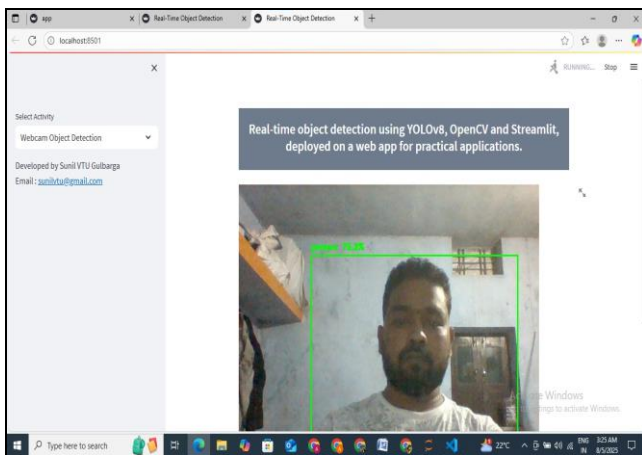


Figure 7.3: object detections a person

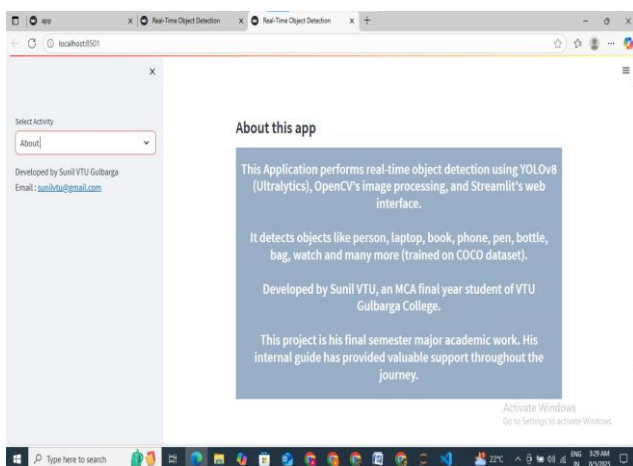


Figure 7.4: about page

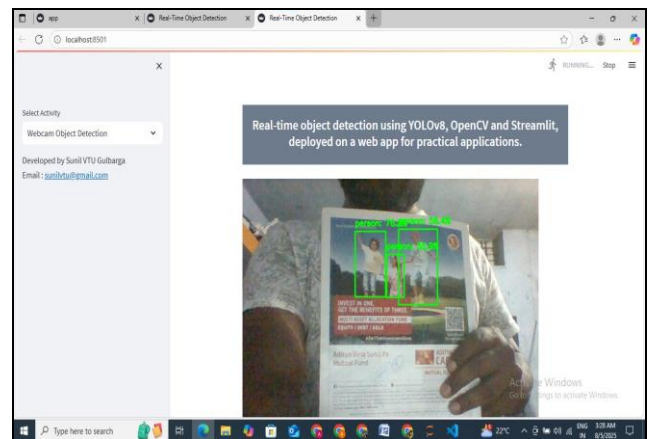


Figure 7.5: multiple object detection

9. SOFTWARE TESTING

Software testing is a crucial phase in system development that ensures the reliability, accuracy, and efficiency of the application. The **Real-Time Object detection System** was tested at multiple levels to identify errors, verify functionality, and validate results against user requirements.

Testing Strategies: The system followed a bottom-up testing strategy, starting from small units (functions, modules) and moving towards the complete system. Both manual testing (for usability and output verification) and automated testing (for functional modules) were applied. The following strategies were used:

- Black Box Testing – Focused on inputs and outputs without checking internal code.
- White Box Testing – Focused on the flow of logic and correctness of algorithms.

Integration Testing: Combined modules such as **video input + preprocessing + detection**.

- Checked if the integration allowed smooth flow of data between components.
- Example: Verified that preprocessed frames were correctly passed to the detection model.

10. CONCLUSION & FUTURE SCOPE

The **Real-Time Object detection System** successfully demonstrates the application of Artificial Intelligence and Machine Learning in solving real-world problems. By integrating state-of-the-art object detection models with live video streams, the system is capable of identifying and classifying multiple objects accurately and efficiently.

Through systematic design, development, and testing, the system has achieved its objectives of real-time performance,

reliability, and user-friendliness. The use of deep learning models such as **YOLO/SSD** ensures high accuracy, while OpenCV enables smooth video frame handling and visualization.

Although the system successfully achieves its objectives of detecting and classifying objects in real time, there are several potential areas where it can be further improved and expanded.

Improved Accuracy: Integrating advanced deep learning models (e.g., YOLOv8, EfficientDet, Transformer-based models) can enhance detection accuracy and robustness for specific domains (traffic monitoring, medical imaging, retail, etc.) can make the system more application-specific.

Faster Processing: Implementing GPU acceleration or deploying models on edge devices (NVIDIA Jetson, Google Coral, TPUs) can reduce latency and improve performance in real-time environments. Optimizing models through quantization and pruning to run efficiently on low-resource devices.

11. REFERENCES

- [1] YOLO — You Only Look Once: unified, real-time object detection. [CV Foundation](#)
- [2] SSD — Single Shot MultiBox Detector: single-stage multi-scale detector. [Computer Science](#)
- [3] Faster R-CNN — Region Proposal Network + detection backbone. [arXiv](#)
- [4] YOLO family progress & practical repos (YOLOv4 / YOLOv5 / YOLOv8). [ResearchGateACM Digital Library](#)
- [5] EfficientDet — scalable and efficient detection via BiFPN and compound scaling. [CVF Open Access](#)
- [6] SSD / MobileNet-SSD and mobile/lightweight detector variants.
- [7] SORT — Simple Online and Realtime Tracking (tracking-by-detection baseline). [ACM Digital Library](#)
- [8] DeepSORT — SORT + deep appearance embeddings for robust ID association. [arXiv](#)
- [9] COCO dataset — Common Objects in Context (standard detection/segmentation benchmark). [arXiv](#)
- [10] MOTChallenge / multi-object tracking benchmarks and metrics.