Department of Electrical and Electronics Engineering

Third Year B.Tech (ECE/ECE-AIML)

Robotics and Automation

Course Code: ECE3002B

PBL REPORT

Made by

Group No.	PRN No.	Name of Student	Branch(ECE/ECE-AI-ML/E&CE)	Contact No.	Email ID
7	1032222545	Ninad Lomte	ECE	7057289365	1032222545@mitwpu.edu.in
7	1032222548	Mahak Khemani	ECE	7058147601	1032222548@mitwpu.edu.in
7	1032222557	Satwik Wakankar	ECE	9112809110	1032222557@mitwpu.edu.in
7	1032233482	Aashi Panchal	ECE	9979851031	aashi.panchal@mitwpu.edu.in

Under the guidance of: Dr.P P. Gundewar

- ## Problem Statement

Traffic congestion remains a significant issue in urban areas, leading to increased travel time, fuel consumption, and air pollution. Traditional fixed-time traffic signal systems are often inefficient in adapting to varying traffic conditions.

To address this problem, we propose an Intelligent Traffic Management System that leverages real-time image processing and ultrasonic distance measurement to dynamically adjust traffic signal timings. By analysing video feeds from cameras and utilizing ultrasonic sensors to gauge vehicle density and spacing, the system aims to optimize traffic flow and minimize congestion.

The primary goal is to develop a system that can effectively detect and track vehicles**,** estimate traffic density**,** and adjust signal timings in real-time to ensure smooth and efficient traffic movement.

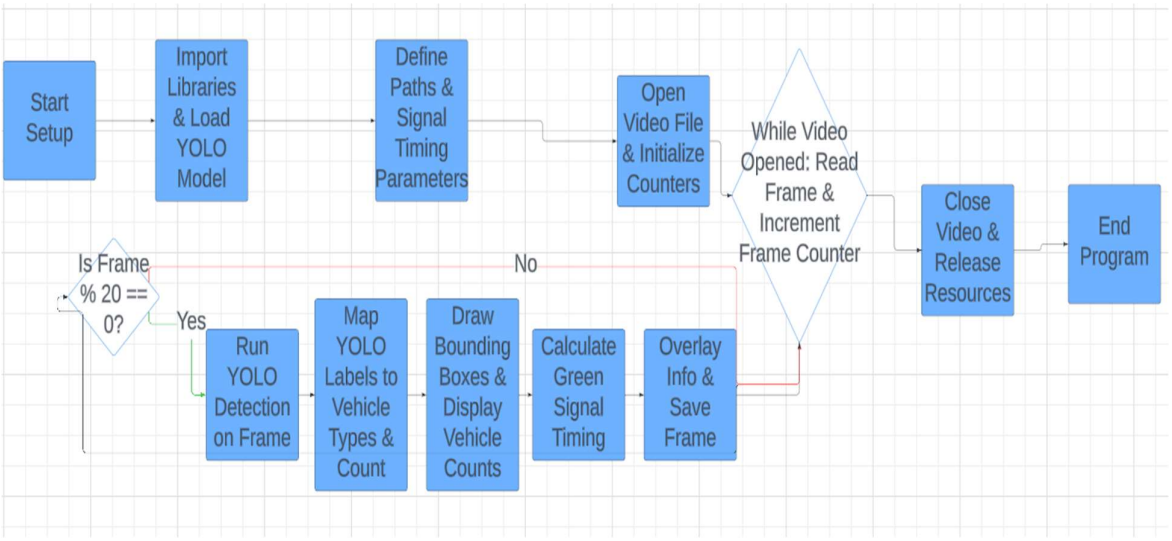
- ## Objectives

1. Capture Images: Use a camera for real-time traffic image acquisition.
2. Process Data: Apply OpenCV and YOLO for vehicle detection.
3. Adjust Signals: Dynamically modify green signal duration based on traffic conditions

- ## Methodology

1. Software Installation: Install OpenCV, YOLO, RPi.GPIO libraries on Raspberry Pi.
2. Camera Calibration: Adjust camera position and settings for optimal traffic capture.
3. Real-Time Image Capture: Capture live traffic images using the Camera.
4. Image Processing: Use OpenCV for image pre-processing (e.g., grayscale, resizing).
5. Object Detection: Apply YOLO for vehicle detection and classification.
6. Traffic Analysis: Combine object detection and distance data to assess traffic density.
7. Signal Control: Adjust green signal duration dynamically based on analysed traffic data.

Block Diagram:-



- ## Specifications

1. Raspberry Pi is our Control Unit.
2. A 50 MP Camera Module for Image Capturing.
3. OpenCV and YOLO for Image Processing &Object Detection.
4. Image Processing to count the number of vehicles in a single frame.

- ## Algorithm for Intelligent Traffic Management System

1. Setup and Initialization

- **Import Libraries:** Import `cv2`, `torch`, and `os` for OpenCV processing, YOLO model loading, and file operations.
- **Load YOLO Model:** Load the YOLOv5s model from `ultralytics` with a low confidence threshold (0.1) to increase detection sensitivity.
- **Set Video and Output Paths:** Define `video_path` for input video and `output_dir` for saving frames. Create `output_dir` if it doesn't exist.

2. Define Traffic Signal Timing Parameters

- **Base Green Signal Time:** Set `BASE_GREEN_TIME` to 30 seconds, the minimum green signal duration.
- **Extra Time per Vehicle:** Define `EXTRA_TIME_PER_VEHICLE` to add 2 seconds per detected vehicle.

3. Initialize Video and Frame Processing

- **Open Video Capture:** Open the video using OpenCV.
- **Initialize Counters:** Set `frame_count` and `total_vehicle_count` to zero.
- **Set Frame Save Interval:** Save every 20th frame by setting `save_frame_interval` to 20.

4. Process Each Frame in the Video

- **Read Video Frame:** While the video is open, read each frame in a loop.
- **Increment Frame Count:** Increase `frame_count` for each read frame.
- **Process and Save Every 20th Frame:**
    - If the frame count is a multiple of 20, proceed with processing and saving the frame.

5. Vehicle Detection and Classification

- **Run YOLO Detection:** Use YOLOv5 model to detect objects in the frame.
- **Extract Detection Results:** Retrieve bounding boxes, confidence scores, and class labels.

6. Map Labels and Count Vehicles

- **Initialize Vehicle Type Counter:** Set up a dictionary to count different vehicle types (e.g., Car, Motorbike).
- **Classify Detected Objects:** Loop through detections:
    - Based on class label, identify the vehicle type (e.g., Car, Motorbike).
    - Increment the count for each detected vehicle type.

7. Display Detection Results on Frame

- **Draw Bounding Boxes:** For each vehicle detected, draw a bounding box with label and confidence score.
- **Calculate Total Vehicle Count:** Sum values in `vehicle_types` dictionary for the total vehicle count.
- **Calculate Green Signal Timing:** Use the function `calculate_green_signal_time()` to determine the green signal time based on the vehicle count.

8. Overlay Text Information on Frame

- **Display Total Vehicle Count and Signal Timing:** Show the total vehicle count and green signal timing on the frame.
- **Display Individual Vehicle Counts:** For each vehicle type, display its count at a specific offset position.

9. Save the Processed Frame

- **Save Frame as Image:** Save the frame as an image file in the output directory with a unique filename based on `frame_count`.

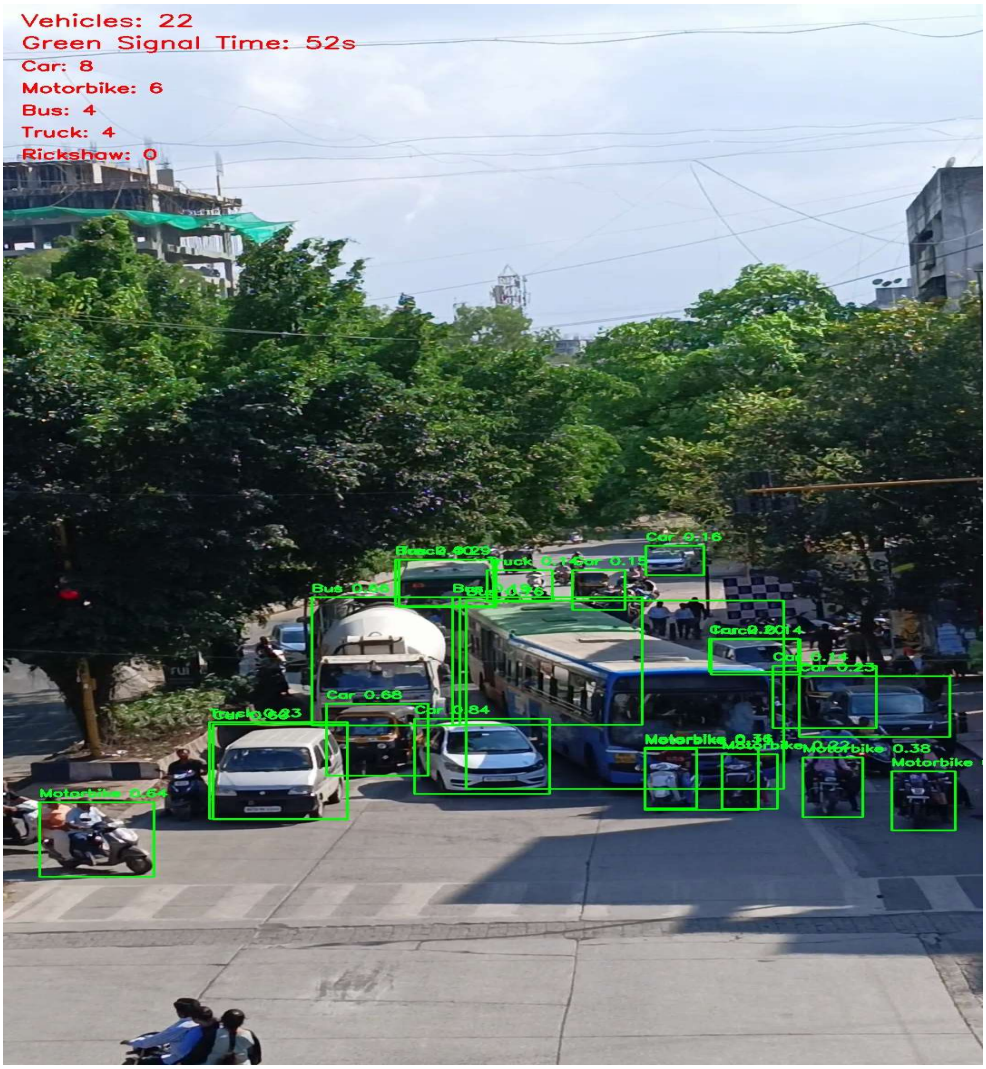
- **Print Frame Save Status:** Print a message confirming the saved frame.

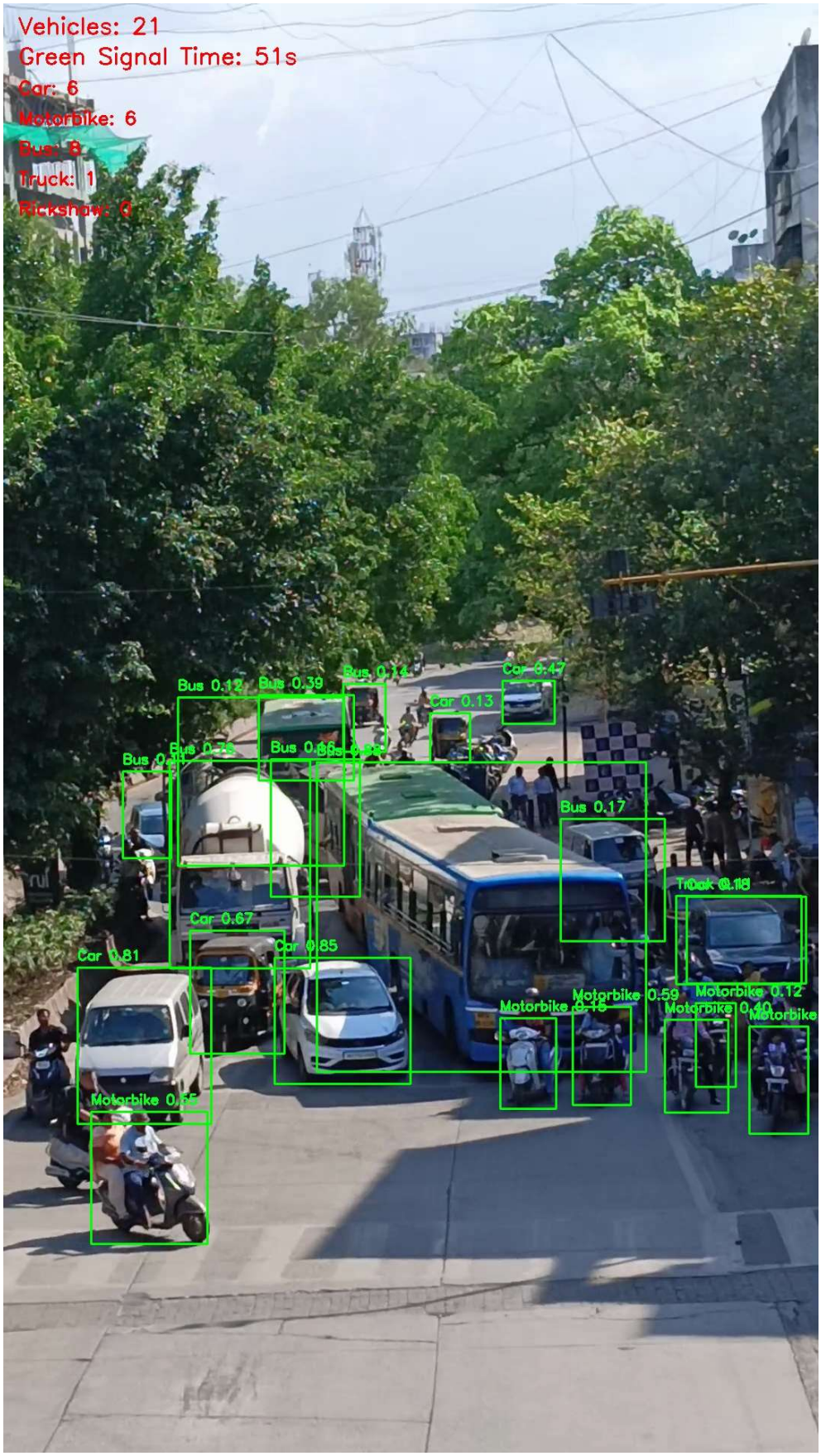
10. Release Resources and Output Summary

- **Release Video Capture and Windows:** Release `cap` and close any OpenCV windows.
- **Print Summary:** Output the total number of frames processed and the location where frames are saved.

- # Result:

We have taken a video from Karve Road and we have obtained the following results




Vehicles: 21
Green Signal Time: 51s
Car: 6
Motorbike: 6
Bus: 8
Truck: 1
Rickshaw: 0
Bus 0.12
Bus 0.39
Bus 0.14
Car 0.47
Car 0.13
Bus 0.17
Car 0.67
Car 0.85
Car 0.81
Motorbike 0.59
Motorbike 0.12
Motorbike 0.40
Motorbike

Conclusion

This project successfully demonstrates the feasibility of an intelligent traffic management system that leverages real-time image processing and ultrasonic distance measurement to optimize traffic signal timings. The system effectively detects and classifies vehicles using the YOLOv5 model, and calculates the required green signal time based on the number of detected vehicles.

Key Achievements:

- **Real-time Image Processing:** The system successfully processes video frames to detect and classify vehicles.
- **Object Detection:** The YOLOv5 model accurately identifies and classifies vehicles in the traffic scene.
- **Dynamic Signal Timing:** The system dynamically adjusts green signal time based on real-time traffic conditions.

Future Work:

- **Hardware Integration:** Integrate the system with real-world hardware components (camera, and traffic light controllers).
- **Robustness Testing:** Conduct extensive testing in various traffic scenarios to evaluate the system's performance under different conditions.
- **Advanced Traffic Control Strategies:** Explore advanced traffic control algorithms like adaptive signal control and vehicle coordination to further optimize traffic flow.
- **Machine Learning Optimization:** Fine-tune the YOLOv5 model and explore other object detection techniques for improved accuracy and efficiency.
- **User Interface:** Develop a user-friendly interface to monitor system performance, adjust parameters, and visualize traffic data.

By addressing these areas, the intelligent traffic management system can be further refined to significantly improve traffic efficiency and reduce congestion in urban areas.