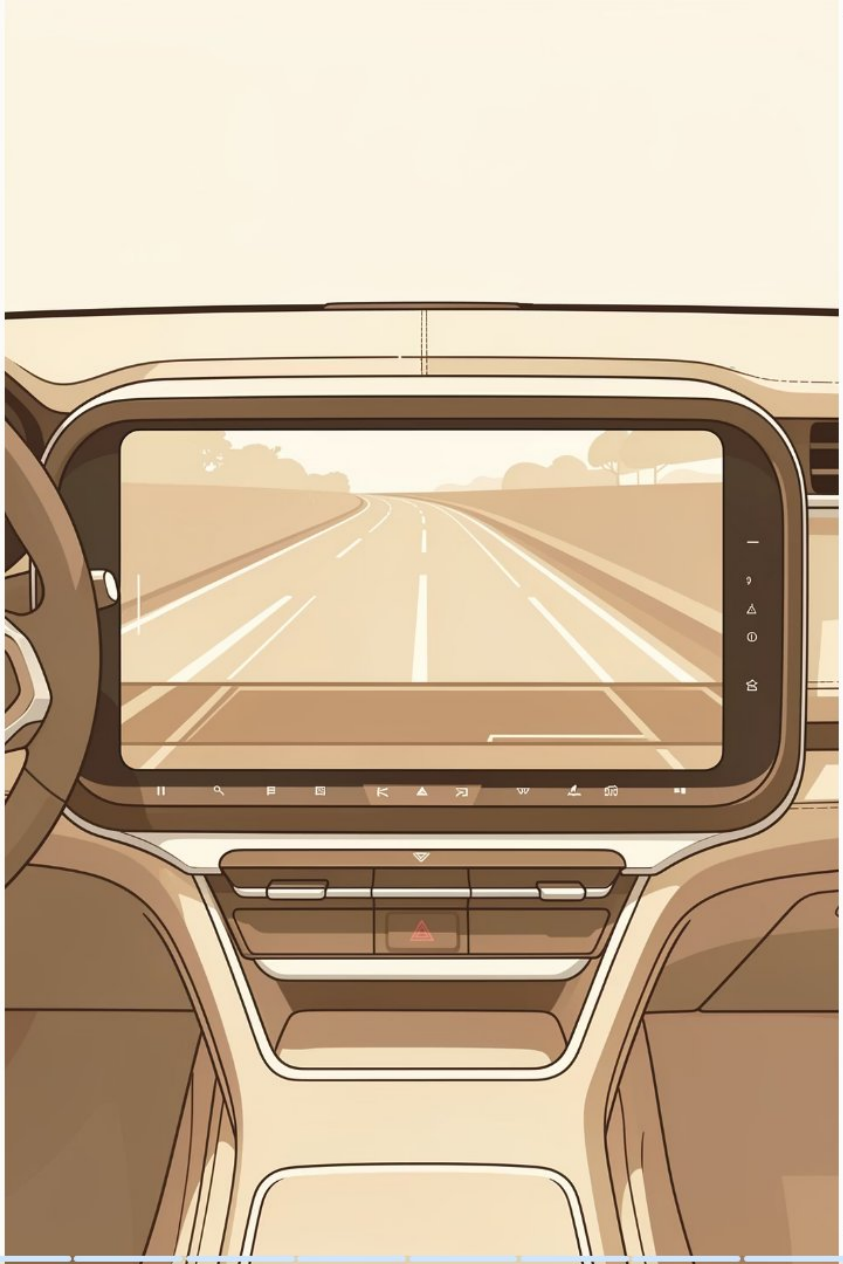
Real-Time Anomaly Detection from Dashcam footage on Raspberry-pi

Real-Time Video-Based Anomaly Detection Using Computer Vision on Raspberry Pi



TEAM MEMBERS

SASMITHA S P

DHIVYA G B

ABIRAMI S A

The Problem

Passive Recording

Traditional dashcams only record video without analyzing or interpreting events in real time.

After-the-Fact Detection

Accident detection usually occurs after the event, limiting preventive action.

Cost Barrier

Advanced ADAS systems are expensive and require hardware-intensive setups.

Project Overview

System Capabilities

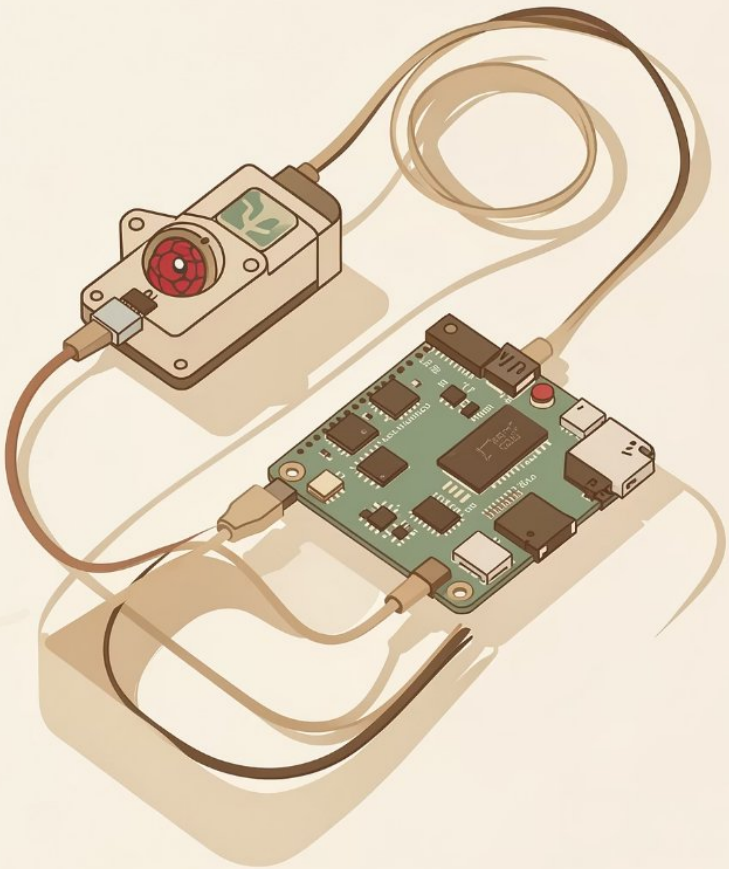
This project implements a real-time computer vision system on Raspberry Pi that captures live dashcam video and processes frames continuously using Python and OpenCV.

A lightweight CNN-based object detection model is used to analyze road scenes, track object behavior across frames, and identify abnormal events such as sudden object appearance or irregular motion patterns.

Key Innovation

Hybrid anomaly detection logic analyzes both motion patterns and object presence, generating real-time visual alerts when unsafe situations are detected.

Instead of simple video recording, the system actively interprets visual data in real time and generates alerts when unusual or unsafe situations are detected, making it suitable for low-cost embedded road safety monitoring.



Software Stack

Python

Serves as the core programming language, orchestrating all system components and logic.

OpenCV

Handles real-time video capture, frame processing, and optical flow analysis for motion detection.

TensorFlow Lite

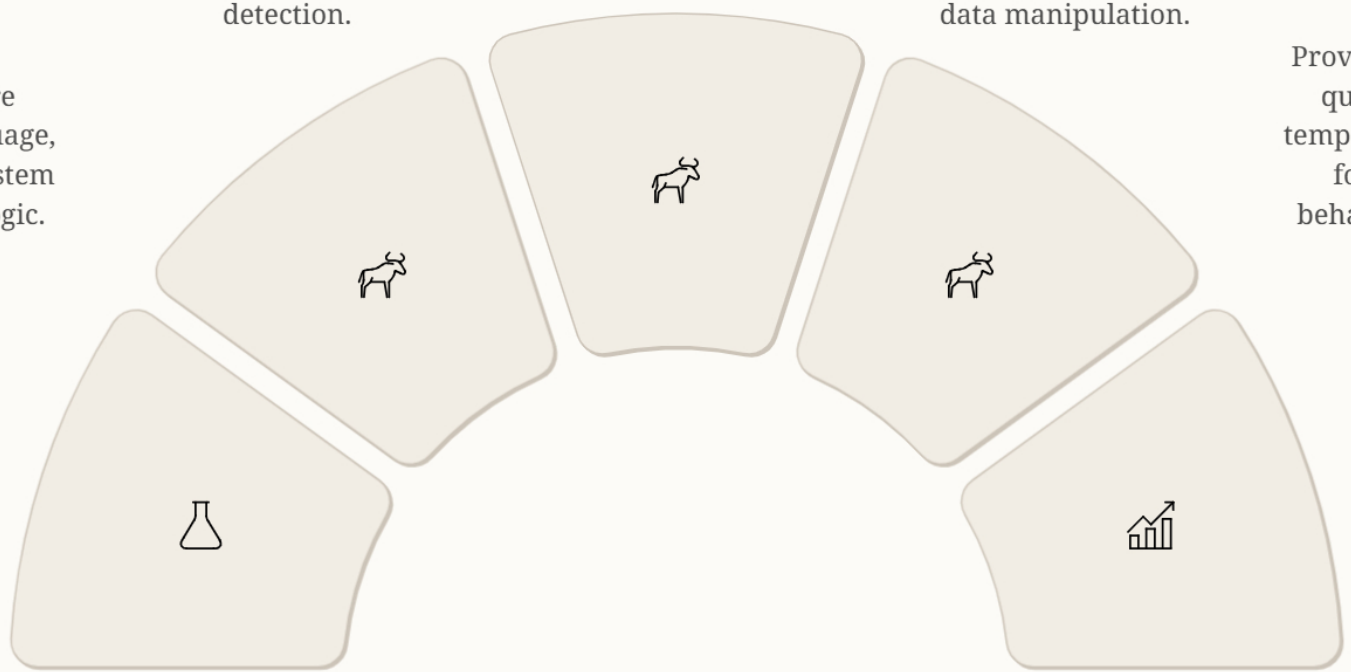
Enables efficient, on-device inference of the CNN model for object detection on the Raspberry Pi.

NumPy

Optimizes numerical operations and manages array-based data structures for efficient data manipulation.

Deque

Provides a double-ended queue for managing temporal memory, crucial for tracking object behavior across frames.



System Architecture

Camera Input

Dashcam continuously streams video frames

Frame Processing

OpenCV captures and preprocesses frames in Python

Motion Analysis

Optical flow detects motion patterns between frames

Object Detection

CNN identifies objects using TensorFlow Lite

Anomaly Alert

If an Object reaches an unsafe distance threshold, Hybrid logic triggers visual warnings on screen

Python Libraries Imported

Python

```
import cv2

import numpy as np

import tensorflow as tf

import time

from collections import deque
```

Hardware Platform

Raspberry Pi Specifications

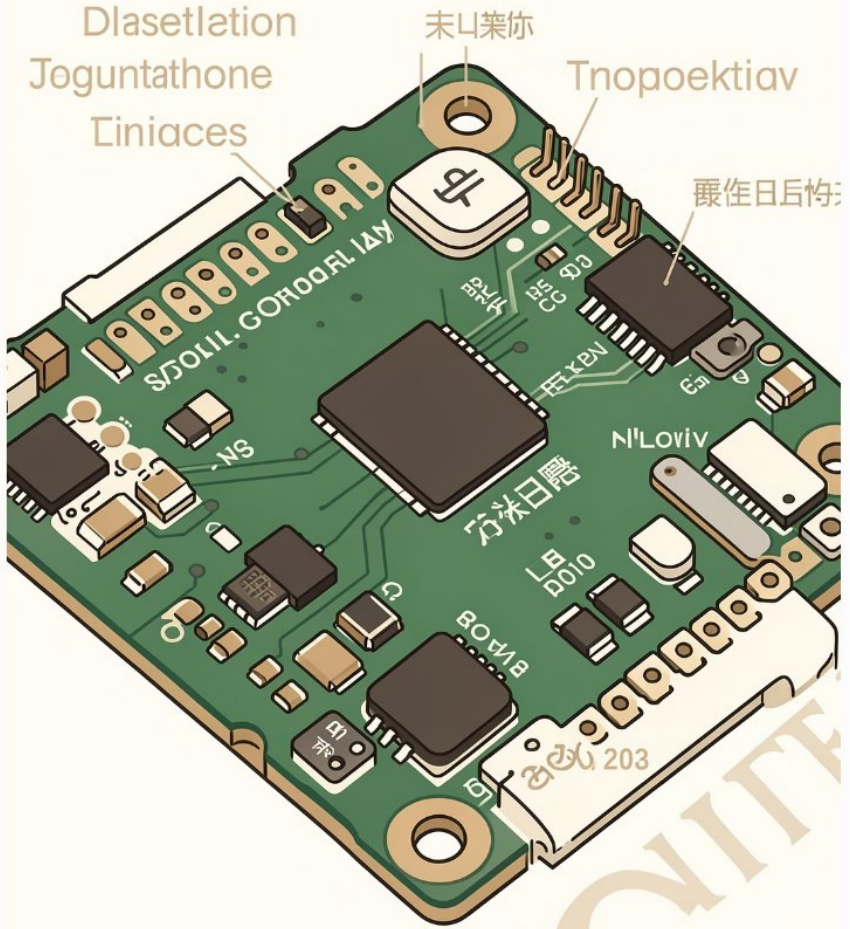
- Broadcom ARM Cortex-A series processor
- ARMv8 64-bit architecture
- ~1.5 GHz quad-core CPU
- 2GB/4GB RAM depending on model

Why Raspberry Pi?

- Supports Python and OpenCV
- Compatible with TensorFlow Lite
- Suitable for real-time edge AI applications
- Low power and cost-effective

Camera Module

Dashcam camera connected to Raspberry-pi provides continuous live video input



Edge AI Implementation

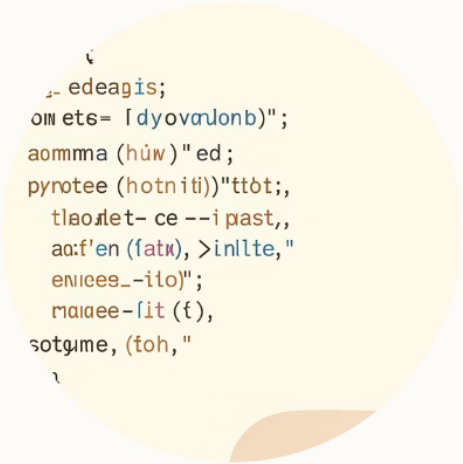
- Real-time video frames captured from dashcam
- Frame processing and analysis performed on Raspberry Pi
- Computer vision implemented using Python and OpenCV
- Distance estimation based on object size and motion across frames
- Anomaly detected when object crosses predefined distance threshold
- Visual warning displayed on screen and audio alert generated

Core Detection Techniques

Optical Flow (Farneback)

Detects sudden or approaching motion between frames

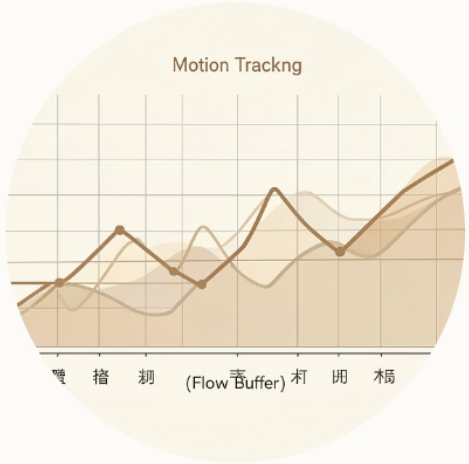
flow = cv2.calcOpticalFlowFarneback(prev_gray, gray, ...) mag, _ = cv2.cartToPolar(flow[...,0], flow[...,1]) flow_mean = np.mean(mag)



Object Detection (TensorFlow Lite)

Lightweight convolutional neural network identifies objects and provides spatial context.

interpreter = tf.lite.Interpreter(model_path="model.tflite") interpreter.invoke()



Temporal Motion Tracking

Maintains motion history to avoid false alarms

flow_buffer = deque(maxlen=40) motion = abs(flow_mean - np.mean(flow_buffer))

Configuration Parameters

Defines system sensitivity and behavior

FLOW_THRESHOLD = 2.0

Controls sensitivity to optical flow motion detection

MOTION_THRESHOLD = 1.0

Sets the baseline for motion deviation detection

CONFIRM_FRAMES = 3

Number of consecutive frames required to confirm an anomaly

Temporal Memory Buffer

Uses Sliding window memory

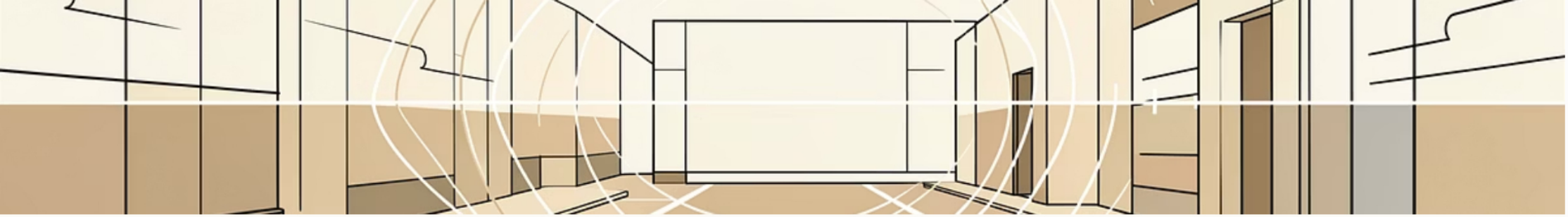
Stores motion History

Helps confirm persistent anomalies

python

motion_buffer=deque(maxlen=40)

flow_buffer=deque(maxlen=40)



Frame Preprocessing

Each frame converted to grayscale

required for optical flow computation

python

gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

Motion Magnitude Computation

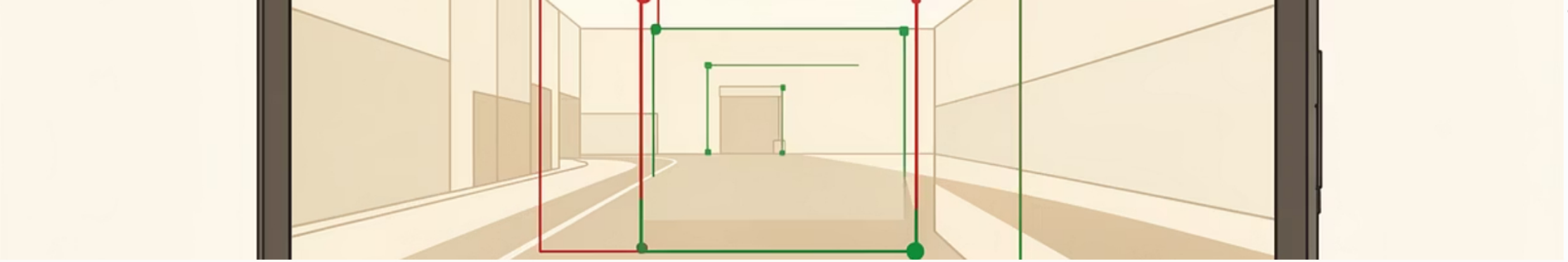
Flow vectors converted to magnitude

Mean magnitude represents scene motion

Python

mag

flow_mean = np.mean(mag)



Visual Feedback

Bounding boxes drawn on detected objects

color changes during anomaly

On-screen alert displayed

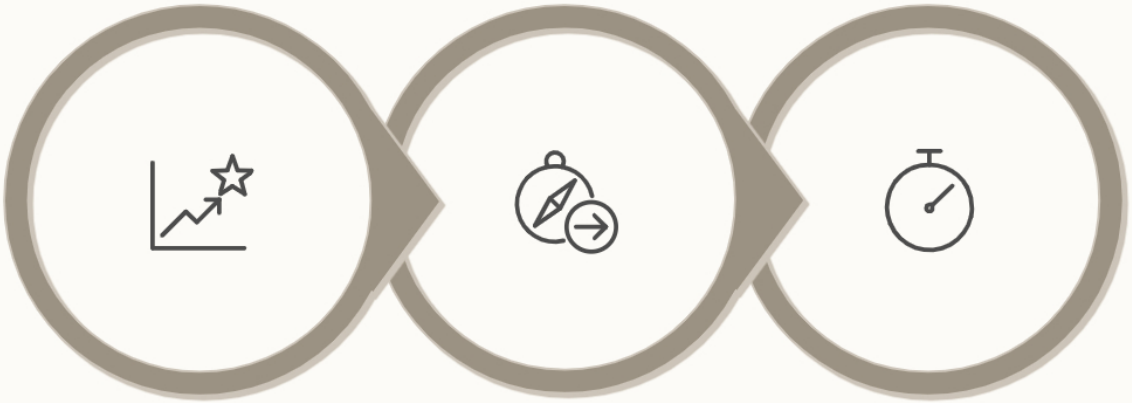
FPS Calculation

Real-time Performance monitoring

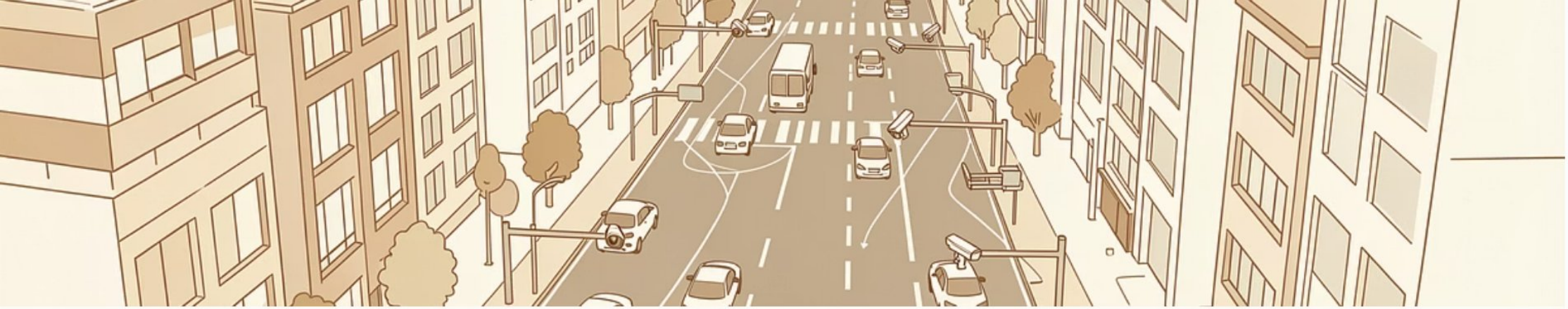
Python

fps = 1.0 / (time.time() - start)

Hybrid Anomaly Logic



An anomaly is detected when significant optical flow motion is observed, valid objects are detected by the AI model, and the condition persists across consecutive frames. This hybrid decision logic improves reliability and minimizes false positives.



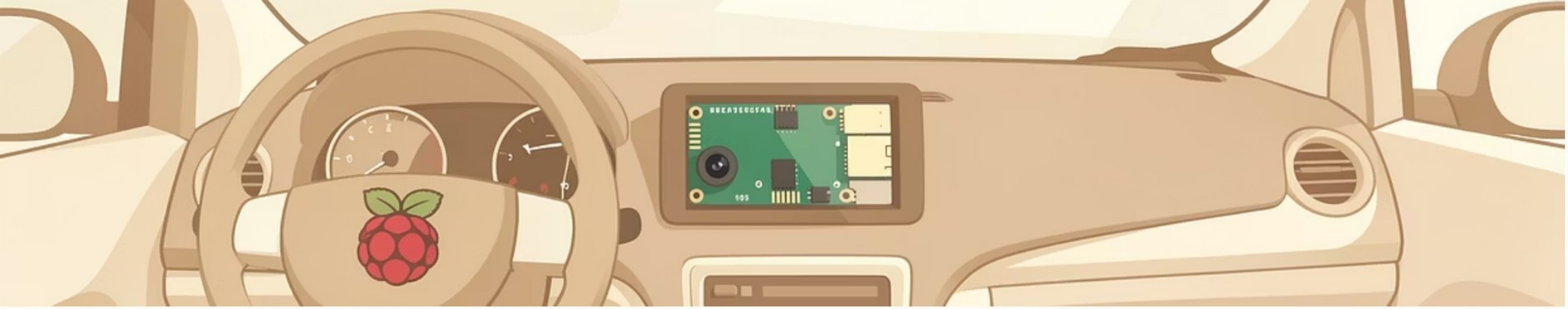
Results & Applications

Performance Achieved

- Real-time anomaly detection
- Stable live video processing
- Works under varying conditions
- Embedded deployment ready

Application Areas

- Smart dashcam systems
- Accident monitoring
- Traffic surveillance
- Intelligent transportation



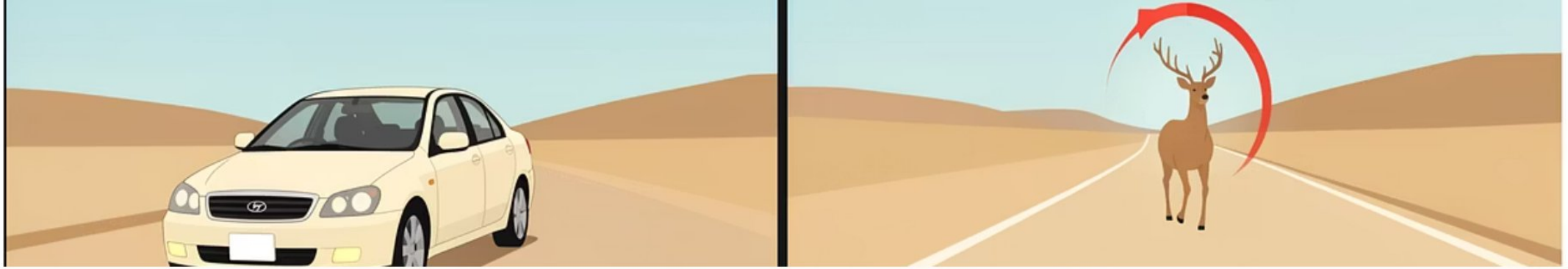
Real World Deployment Scenario

Raspberry Pi mounted in vehicle

Camera facing forward

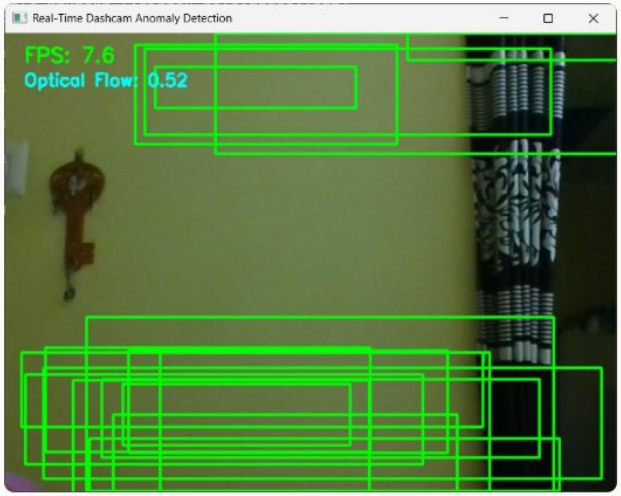
Systems runs automatically at startup

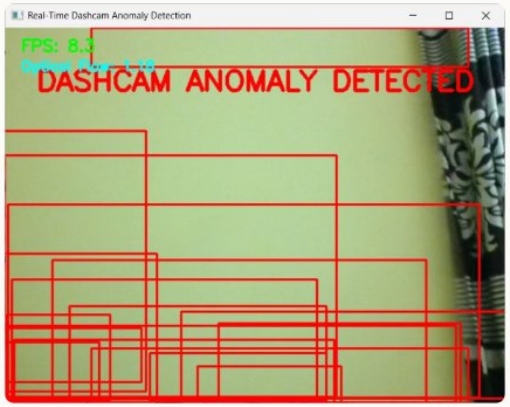
Alerts shown on connected display



Performance Snapshot

Real-Time Dashcam Anomaly Detection Dashcam anomaly detected





Conclusion & Future Scope

Project Success

This project demonstrates that real-time video-based anomaly detection can be implemented on low-cost edge devices such as Raspberry Pi, enabling practical deployment for road safety monitoring.

Future Enhancements

- Distance-based collision detection
- Audio warning alerts
- Object classification
- Sensor fusion with IMU and radar
- Hardware accelerator integration

Thank You!

For your attention and support in our project on real-time anomaly detection.