ECE 445

SENIOR DESIGN LABORATORY

FINAL REPORT DRAFT

Continuous Vehicles Capture

Team #21

JIAWEI ZHANG
(jiaweiz8@illinois.edu)
YINING GUO
(yiningg5@illinois.edu)
BINYANG SHEN
(binyang4@illinois.edu)
ZIJIN LI
(zijinli3@illinois.edu)

TA: Pujing Lin

May 18, 2025

Abstract

This project demonstrates the design and implementation of a real-time traffic monitoring and analysis system built on the Raspberry Pi platform. The system integrates computer vision algorithms with the Raspberry Pi's processing capabilities to detect, classify, and track vehicles in traffic flows. Our solution utilizes the OpenCV image processing library and employs a lightweight neural network model optimized for the computational constraints of the Raspberry Pi. This embedded solution provides a cost-effective alternative to traditional traffic monitoring systems while offering comparable functionality. The system's modular design allows for easy deployment in urban environments without requiring extensive infrastructure modifications. The system demonstrates reliability in continuous operation and shows potential applications in smart city traffic management, urban planning, and intelligent transportation systems.

Contents

1 Introduction

1.1 Background

With rapid urbanization intensifying traffic congestion globally, traditional monitoring approaches such as manual observation and fixed sensors increasingly struggle to deliver real-time, accurate road condition data. These methods often suffer from high deployment costs, limited spatial coverage, and delayed response to dynamic traffic changes. Municipal authorities consequently face critical challenges in optimizing traffic flow and mitigating congestion. This gap underscores the urgent demand for an adaptive, portable monitoring solution capable of providing instantaneous vehicle density analytics and congestion assessments across diverse road networks.

1.2 Objective

This project aims to develop a Raspberry Pi-powered portable traffic monitoring system that integrates computer vision and edge computing technologies. The system will employ camera modules for continuous video streaming, coupled with YOLO-based object detection algorithms to achieve real-time vehicle identification, classification (e.g., cars, trucks), and multi-lane counting. By processing traffic flow patterns through density heatmaps and velocity vectors, it will dynamically generate congestion indices while operating under 5W power constraints for outdoor sustainability. Key deliverables include an interactive dashboard for data visualization, configurable SMS/email alert thresholds for traffic management units, and a modular architecture supporting future integration with smart city infrastructures. The solution prioritizes deployment flexibility, achieving $\geq$90% detection accuracy in varying lighting conditions, while maintaining a unit cost below $200 for scalable urban implementation.

1.3 Subsystem Overview

2 Design

2.1 Subsystem Diagrams and Schematics

2.1.1 Mechanical subsystem Design

he mechanical design began with the goal of creating a compact, modular and field-deployable traffic monitoring platform. We adopted an iterative development cycle that starts with breadboard-based prototyping on a red plastic baseboard. This early platform supported the rapid integration of the Raspberry Pi, camera module, and peripheral components, facilitating pin mapping, wiring layout, and early-stage function verification.

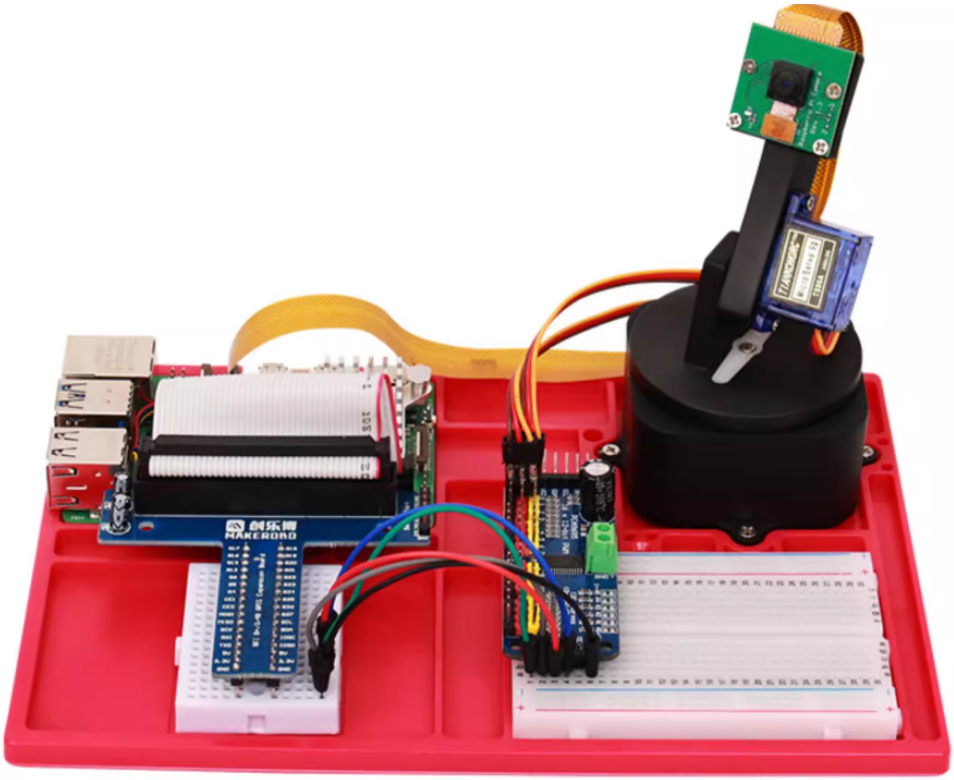

Figure 1: Ideal Platform

After electrical prototyping was completed, the focus shifted to enclosing the system in a field-ready housing. The design requirements included water resistance, ease of assembly, stability of the internal components, and structural support for the camera.

We transitioned to CAD modeling using Fusion 360, where modular parts were created: a base shell for component mounting, a lid with clearance for wires and airflow, and a camera mount system with adjustable geometry. The prototypes were printed in PLA+ for strength and weather resistance.

Snap-fit camera shell: a redesigned 3D printed enclosure that houses the camera module.

Unlike previous iterations that required screws or adhesive, this version employs a press-fit tab locking system for faster assembly and better modularity.

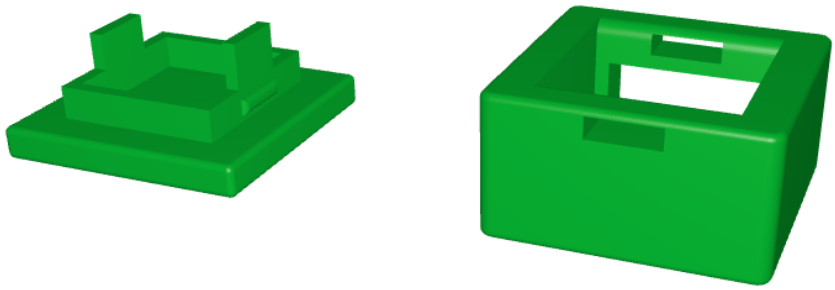

Figure 2: Camera Shell

Passive cooling and ventilation optimization: To ensure thermal stability during continuous outdoor operation, we incorporate passive cooling strategies into the enclosure design. The base and lid feature airflow channels aligned along the sidewalls, which promote natural convection without allowing direct ingress of rainwater. A dedicated opening beneath the Raspberry Pi heat sink improves vertical airflow, and the surrounding clearance ensures that radiated heat is not trapped inside the housing. This feature significantly reduces thermal build-up under high ambient conditions, especially when the camera and wireless modules are operating simultaneously. Internal temperature monitoring during test cycles confirmed that the addition of this ventilation path reduced the internal temperature of the enclosure compared to fully sealed prototypes.

The mechanical design of this project consists of four main parts: the base (part 1), the camera face (part 2), the camera bracket (part 3) and the cover (part 4). These parts were assembled in a modular fashion to form a complete device housing system as shown in the combination diagram.

3D modeling and printing technology facilitates the assembly of the overall structure, ensuring effective protection and adaptability during installation.

The bottom of the enclosure is reserved for USB, power and signal ports and features a concave wall structure for enhanced dust protection and cabling. The top cover structure has a column mounting position for fixing the camera bracket and is designed with a slide slot or mating port for quick removal and installation, improving the maintainability of the device. The camera bracket works in conjunction with the panel to achieve a stable camera position, and a standard window in the panel ensures that the image area is not obscured.

Figure 3: Cooling Shell

These components are assembled into a compact, airtight fixed surveillance unit that meets the multiple requirements of equipment protection, clear image and signal access in outdoor operations.

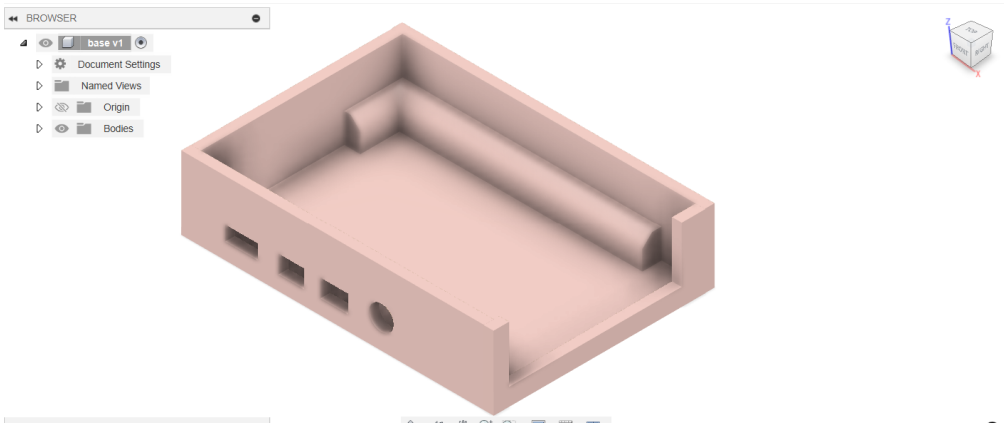

Figure 4: Part 1

2.1.2 Circuit Diagram

2.1.3 Visualization Diagram

This web interface showed in Fig9 presents a comprehensive vehicle detection and tracking system dashboard developed totally by Jiawei Zhang. The layout follows a clean, card-based design organized into distinct functional sections. At the top, a header dis-

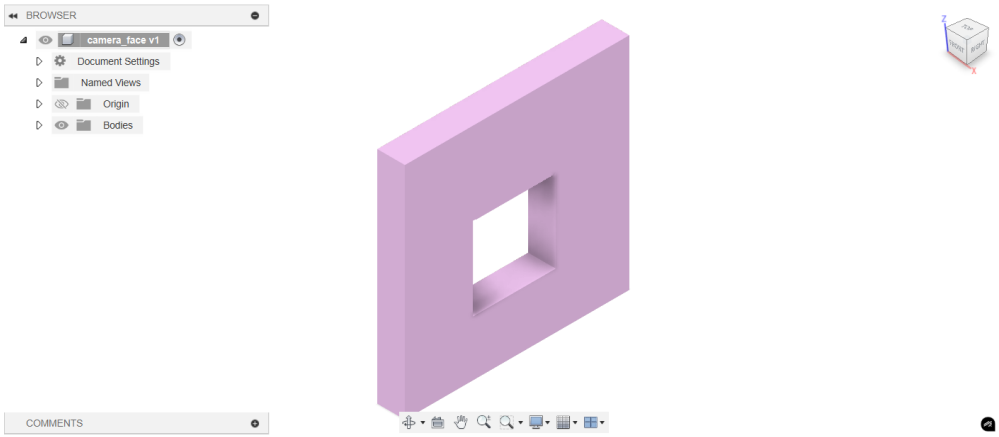

Figure 5: Part 2

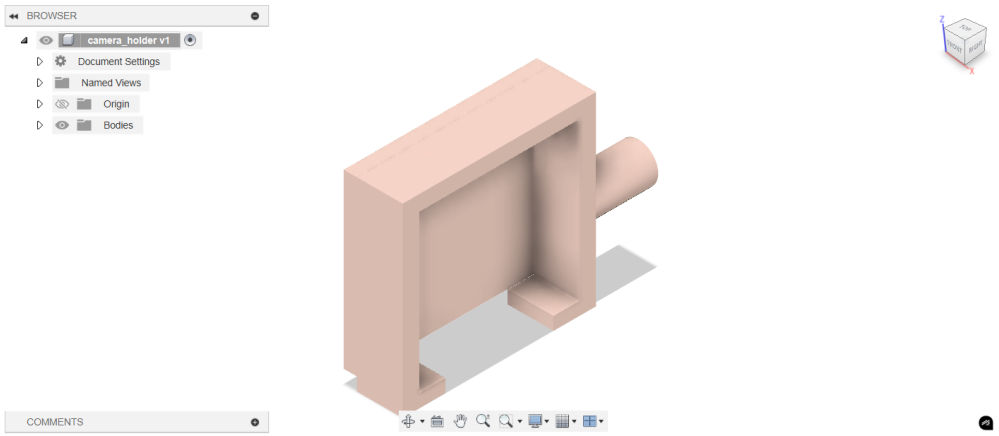

Figure 6: Part 3

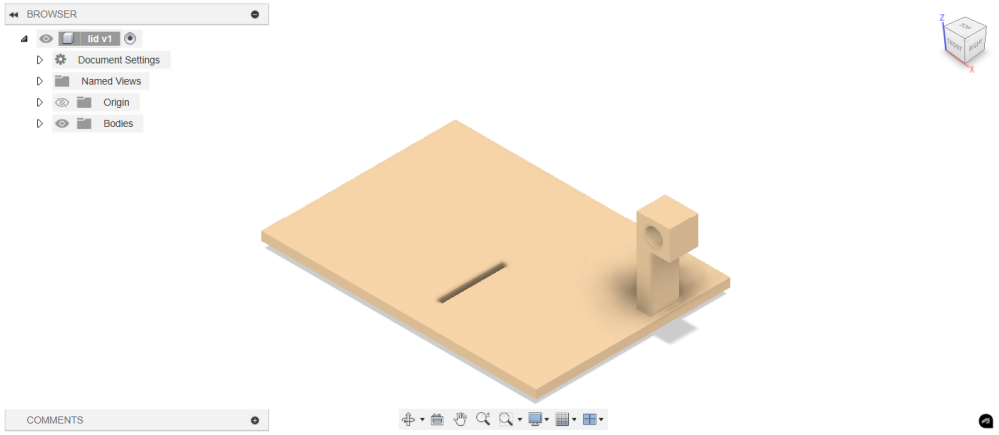

Figure 7: Part 4

plays the system title and connection status, followed by a customizable alarm configuration panel with threshold sliders and vehicle weight settings. The dashboard's central

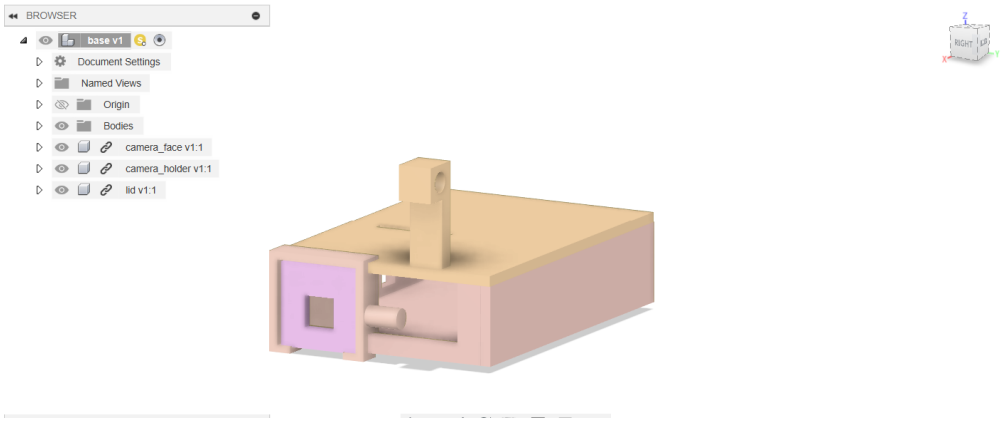

Figure 8: General overview

area features statistical cards showing current and cumulative counts for cars, trucks, and buses, with color-coded icons for easy identification. Below this, the vehicle history section displays both active and inactive vehicles as individual cards that progressively fade based on detection recency. Two interactive charts visualize the detection data: a timeline graph showing vehicle counts over time and a pie chart displaying the distribution of vehicle types. At the bottom, a detailed log table records all detection events and system messages. The interface incorporates real-time updates through Socket.IO connections, with thoughtful visual indicators like an alarm system that activates during high-density traffic situations. The entire dashboard is responsive and includes interactive elements for data filtering, refresh controls, and system configuration.

2.2 Subsystem Descriptions and Details

2.2.1 Mechanical Subsystem

2.2.2 Image Processing Subsystem

The Detection Module serves as the visual perception system of our vehicle tracking solution. At its core, this subsystem employs advanced deep learning techniques through the YOLOv5 neural network to identify various vehicle types within video frames. Unlike traditional computer vision approaches that rely on hand-crafted features, our system leverages the power of convolutional neural networks to recognize cars, buses, and trucks with remarkable robustness across varying lighting conditions and partial occlusions. The module operates on an intelligent keyframe approach, analyzing selected frames at regular intervals rather than processing every single frame. This strategic design choice maintains high detection accuracy while significantly reducing computational demands, allowing the system to function effectively even on hardware with limited processing power. When a vehicle is detected, the module generates precise bounding box coordinates and assigns class labels, creating the foundation for all subsequent tracking and analysis operations within the broader system.

The Tracking Module represents the intelligence center of our vehicle monitoring system.

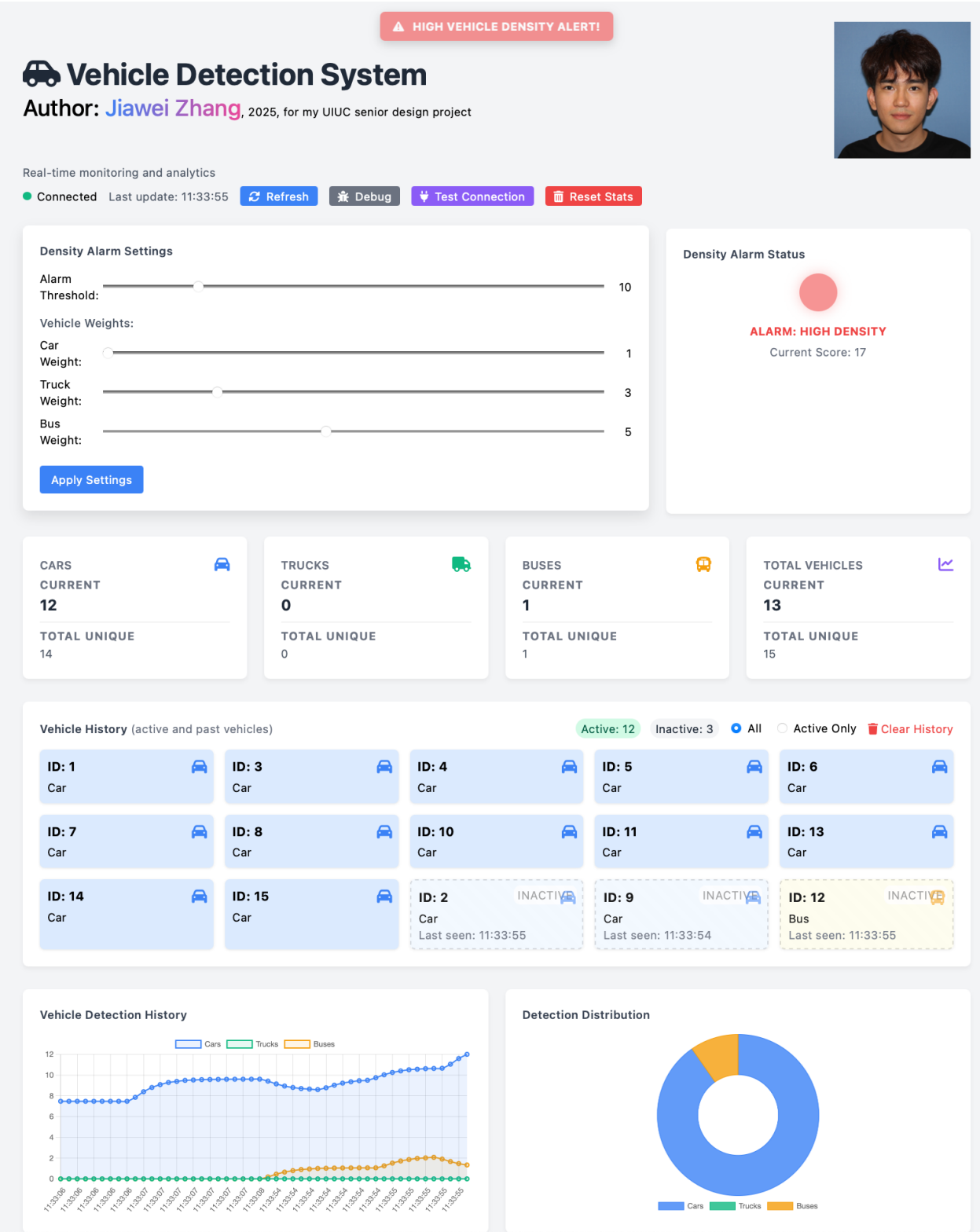

Figure 9: Web data dashboard

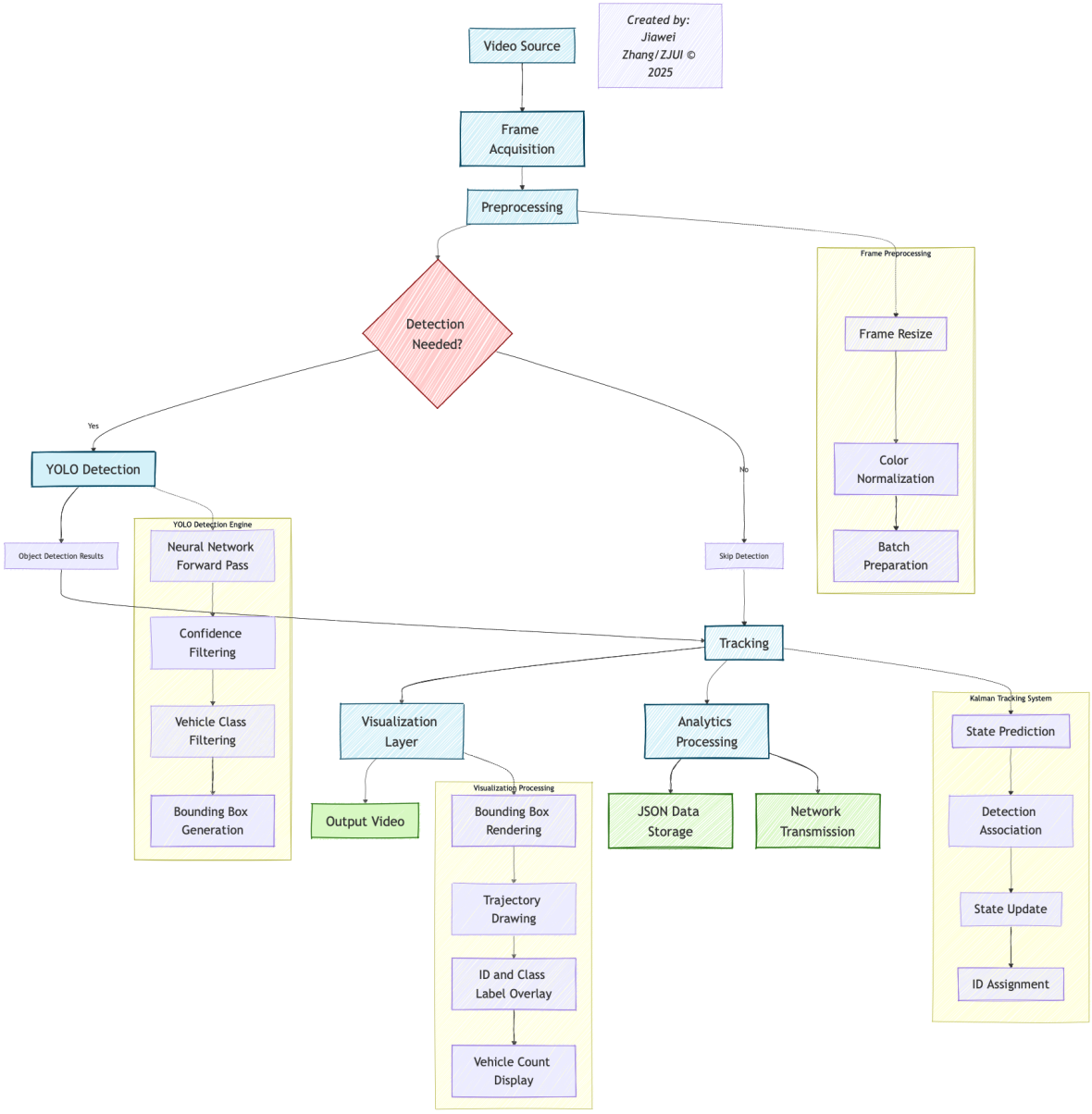

Figure 10: Enter Caption

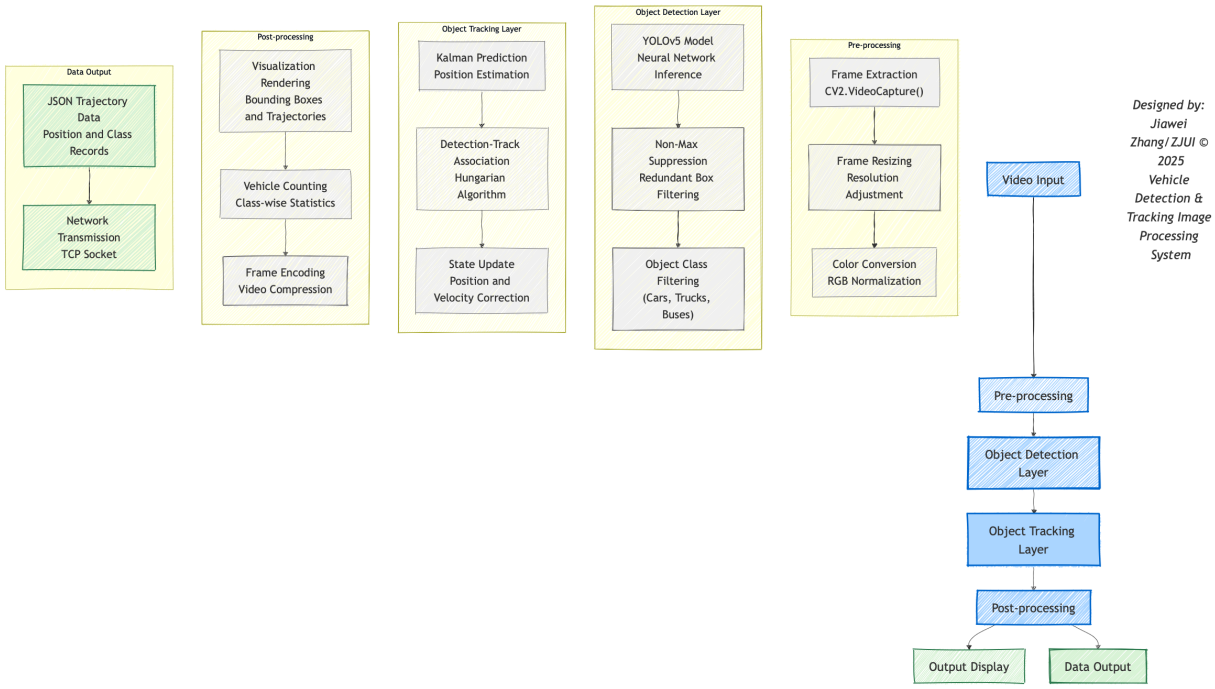

Figure 11: Enter Caption

This subsystem takes the initial detections and transforms them into coherent vehicle trajectories across time, essentially solving the complex problem of maintaining vehicle identity between frames. At its heart lies a sophisticated Kalman filtering framework that creates a mathematical model of each vehicle's motion, allowing the system to predict where vehicles will appear in subsequent frames even when detection temporarily fails. The module employs a delicate balance of prediction and measurement update steps, continuously refining its understanding of vehicle position and velocity. When new detections arrive, the tracker uses an elegant Hungarian assignment algorithm to match these observations with existing tracked vehicles in the most optimal way possible. This approach allows the system to handle challenging scenarios like brief occlusions when one vehicle passes behind another, or momentary detection failures in suboptimal lighting conditions. The combination of robust mathematical modeling and efficient assignment algorithms enables the tracking module to maintain reliable vehicle identity even in dense traffic scenarios with numerous similar-looking vehicles.

The Data Management Module functions as the system's memory and communication center. This subsystem handles the crucial tasks of organizing, storing, and transmitting the rich stream of vehicle tracking data generated during operation. The module implements a structured JSON-based recording system that methodically captures each vehicle's position, size, class, and identity across frames, creating a comprehensive digital record that can be analyzed for patterns and insights. For networked deployments, the module includes a sophisticated transmission component that efficiently shares tracking data with remote systems through TCP socket connections. This capability enables

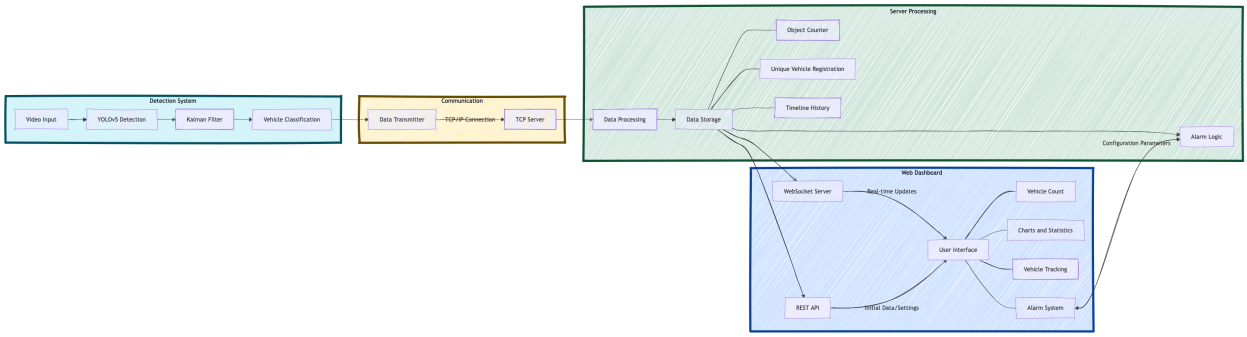

Figure 12: Enter Caption

distributed applications where processing happens on one device while visualization or higher-level analytics occur elsewhere. The data management approach prioritizes both completeness and efficiency, ensuring that all relevant tracking information is preserved while avoiding unnecessary data duplication or transmission. This balanced design makes the module suitable for both offline analysis scenarios where comprehensive records are paramount, and real-time monitoring applications where transmission efficiency becomes critical.

During the progress of our project work, we discovered a very troublesome issue. When we considered and added the vehicle speed calculation, we found that due to the limited computing resources of the Raspberry PI hardware and the limited number of video frames captured by the camera, even if we used the most advanced calculation method, it was still very difficult for us to calculate the vehicle speed very effectively. This is largely limited by the fact that our system has only one monocular camera and no other more sensors that can obtain vehicle speed information. After multiple verification tests, we considered giving up the function of vehicle speed detection. There are two considerations for us to abandon this function: First, this is not the main goal of our entire project. Second, when the vehicle speed detection module is added, it will greatly increase the computational load of each frame captured by the Raspberry party, which leads to the overall system working very unsmoothly. Due to the huge computational load required for vehicle speed detection, even if the camera can capture enough frames, Theoretically, it is possible to achieve vehicle detection, type identification, and traffic flow statistics. However, in reality, due to the problem of the program's calculation speed, the final video frames that can be effectively processed are very limited, which greatly affects the basic operation of the entire system. Therefore, we consider removing the unimportant function of the vehicle speed detection system.

2.2.3 Data Visualization Subsystem

The vehicle detection system provides a comprehensive visualization interface that transforms raw detection data into an intuitive, real-time monitoring dashboard. The system receives data from a detection module that identifies vehicles in video footage, processes

this information through a server component, and presents it in a visually engaging web interface. This creates a complete pipeline from physical vehicle detection to interactive data presentation.

At the heart of the system is a robust backend server that receives detection data through TCP socket connections. The server normalizes and processes incoming data streams, maintaining both current state and historical records of detected vehicles. It implements an intelligent alarm system that evaluates vehicle density based on configurable weights for different vehicle types, enabling customized monitoring thresholds. The server also distinguishes between unique vehicles to maintain accurate cumulative statistics over time.

The visualization dashboard presents vehicle detection data through an elegant, card-based interface designed for immediate comprehension. Statistical counters display both current vehicle counts and cumulative unique vehicles by type. The interface features two primary charts - a timeline history showing vehicle counts over time and a pie chart illustrating the distribution of detected vehicle types. Color-coding distinguishes between cars, trucks, and buses throughout the interface, creating a consistent visual language.

A particularly innovative aspect of the system is the vehicle history display, which shows both active and inactive vehicles through a progressive fading mechanism. Recently detected vehicles appear prominently, while vehicles that have left the scene gradually fade away but remain accessible in the history. This approach maintains situational awareness of current vehicles while preserving historical context. Users can filter to show only active vehicles or view the complete detection history.

The alarm system provides intelligent monitoring of vehicle density with a customizable configuration interface. Users can adjust the alarm threshold and set different weights for each vehicle type, allowing the system to prioritize larger vehicles or specific classes in density calculations. The alarm manifests visually through an indicator light, status messages, and a prominent banner notification when activated, ensuring users are immediately aware of high-density situations.

Real-time communication ensures the dashboard stays continuously updated as new vehicles are detected. The system uses WebSocket technology to push updates to the client as they occur, while also providing REST API endpoints for configuration changes and data queries. A heartbeat mechanism maintains connection status awareness, with automatic reconnection attempts if the connection is interrupted. This bidirectional communication enables both live updates and user-initiated controls like refreshing data or adjusting alarm parameters.

3 Costs

Description	Quantity	Cost ($)
Raspberry Pi 5B (5GB)	1	55
Power Supply	1	10
Camera	3	55
3D Printed Housing	2	40
Total Parts Cost		160

Grand Total

Category	Total Cost ($)
Labor	1000
Parts	160
Grand Total	1160

4 Project Schedule

Date	Jiawei Zhang	Binyang Shen	Yining Guo	Zijin Li
2/24/25	Project initiation and team building	Project initiation and team building	Project initiation and team building	Project initiation and team building
3/3/25	Research on Yolo series models	Research for different models of camera housings	Learn the knowledge about Yolov8 and OpenCV	Primarily study about Unity and D3.js
3/10/25	Implementation of Yolo target detection model deployment and inference	Select camera model and determine housing size	Search for suitable datasets	Design the visualization framework
3/17/25	Completed the vehicle detection of one-minute traffic video by using Yolov8 model	Preliminary design of camera housing model with CAD	Detect the vehicle detection performance with the datasets and expand our datasets	Design the visualization logic structure
3/24/25	Added detection boxes to the inference script to identify vehicles in fixed areas	Evaluation of the strength and hardness of 3D printed models	Observe the differences between images preprocessed using OpenCV and those without such processing. Evaluate the results and determine, based on the specific conditions, whether to incorporate OpenCV-based preprocessing.	Collaborate with teammates to verify the datatype and recognition feature

3/31/25	Implemented the first version of the speed calculation algorithm	Design additional equipment such as brackets to suit the actual situation	Integrate Speed Measurement with Detection Outcomes	Create the necessary graph structure and pattern
4/7/25	Optimize and improve the speed calculation algorithm	Design improvements to the first version of the model	Enhance Traffic Density Calculation Using Optimized Speed Data	Pretest the visualization part with manually importing data
4/14/25	Design and try the traffic density index and calculation algorithm	Performs second shell 3D printing	Improve classification and counting functions, and achieve higher accuracy	Assemble all visualization unit
4/21/25	Optimize traffic density index and algorithm; Establish the logical module and code implementation between the data receiving end and the vehicle identification module for data transmission	Assembly test with other parts together	Verify the functions of speed detection and density analysis	Test visualization part with real-time data
4/28/25	Build the final overall system with co-responsible partners; Begin the design of the visual web page	Final rectification based on test results	Build the final overall system with co-responsible partners	Debug and successfully output summary based on different monitoring time scale

5/12/25	Collaborative data visualization, joint debugging system; Design various sections of the visual web page, including real-time curve graphs, real-time kanban boards, and real-time vehicle type statistics	Measurement of the size data after completion of final micro-modifications	Think more about the real-life application and finally adjust the system	Improve the user-interface and improve visualization of targeting data
5/19/25	Write Final Report; Prepare Final Demo and Presentation	Prepare Final Presentation	Prepare Final Presentation	Prepare Final Presentation

5 Requirements and Verification

5.1 Completeness of requirements

5.2 Verification procedures and quantitative results

5.2.1 Mechanical Unit Requirements

Requirement	Verification Method	Pass Criteria
.	.	.
.	.	.

Table 2: Mechanical Unit Verification Protocol

5.2.2 Image Processing Requirements

Requirement	Verification Method	Pass Criteria
The vehicle Detection shall achieve a minimum accuracy of 85% accuracy under daylight conditions maintain at least 70% accuracy in low-light scenarios. Detection frames can be added to the detected vehicles.	Using a test set that includes vehicles of multiple categories, calculate the precision and recall rates of the classification results based on the test set. Also, using test videos, compare the system's detection results with the actual situations in the videos, and calculate the detection accuracy.	The accuracy rate of vehicle detection under daylight conditions is $\geq 85\%$ and $\geq 70\%$ under low-light scenarios.
Vehicle classification shall distinguish between car, bus and truck with 80% accuracy.	Evaluate classification performance on more than 20 test videos containing enough cars, buses and trucks.	The accuracy rate exceeds 80%.
It can accurately track the corresponding cars and assign a corresponding id to each car. Within the frame where the car is detected, the car id is fixed and will not change..	We need to record the id changes of each vehicle on the test video set. The id remains unchanged as a positive sample, while the id changes as a negative sample.	The proportion of positive samples exceeds 80%.

It can accurately record the traffic density within a certain period of time, that is, the number of vehicles detected on the road.	The number of vehicles in each frame is manually counted and then compared with the results detected by the program. If an error occurs in each frame, it is recorded as a failure; if there is no error, it is recorded as a success.	The proportion of positive samples exceeds 80%..

Table 3: Image Processing Verification Protocol

5.2.3 Traffic Analysis Requirements

Requirement	**Verification Method**	**Pass Criteria**
The system maintains a stable network connection. The received data should be able to be displayed in real time on the web page.	The data transmission is constantly interrupted in the middle. The data receiving section is checked to see if it can still work without reporting errors when the network is unstable, the data transmission is discontinuous or interrupted.	The web board can run continuously for more than 10 minutes without crashing.
Inspectors can interact with buttons on the web page, including clear historical data, adjusting the size of thresholds and the allocation of weights for each vehicle, with information records and good interaction support.	Test each button on the web page, including those for clearing historical data, adjusting thresholds, and changing weight allocations. Check if the interactions are responsive and if the changes are correctly recorded and reflected.	All buttons function correctly, and the system records and applies the changes made by the inspectors without any errors.

In the web dashboard, the currently detected vehicles can be reflected in real time, and the categories and data of the vehicles can be statistically analyzed, including the ability to distinguish between the currently detected vehicles and retain the previous ones.	Introduce multiple vehicles into the system and observe if their categories and data are immediately displayed on the web dashboard. Check if the system can distinguish between current and previous detections.	The web dashboard accurately reflects real-time vehicle data and provides correct statistical analysis, including distinguishing current and past detections.
When the traffic density exceeds the set threshold, it should be able to give an alarm in real time.	Gradually increase the number of vehicles until the traffic density exceeds the preset threshold. Observe if an alarm is triggered in real time on the system.	The system triggers an alarm promptly when the traffic density exceeds the threshold.

Table 4: Traffic Analysis Verification Protocol

6 Conclusion

6.1 Accomplishments

We have successfully developed a continuous vehicle capture system based on Raspberry Pi, characterized by low power consumption, high portability, and outstanding adaptability. With the rapid advancement of society, the increasing frequency and range of human travel, and the growing diversity of transportation modes, relevant authorities require an integrated traffic planning and monitoring network. The Raspberry Pi-based vehicle capture system we have built can serve as a template for network access points, enabling rapid expansion and flexible adaptation to ever-changing demands, thereby facilitating the establishment of a comprehensive traffic surveillance and planning network at a significantly reduced cost.

6.2 Uncertainties

Currently, the capture accuracy of the system we have developed under varying lighting conditions cannot be effectively predicted or guaranteed, as factors such as weather, time, and location can all influence lighting conditions, thereby affecting the system's performance. This represents the primary source of uncertainty for our system. Additionally, during prolonged continuous code debugging, we unexpectedly discovered fluctuations in hardware-level processing speed and connection stability, which may also become potential sources of failure and affect the system's reliability in future applications.

6.3 Future Work

Although the computational capabilities of the Raspberry Pi as a processing module fall significantly short compared to computing clusters or even typical personal computers, we still aim to fully leverage its processing power, memory, and network speed. In doing so, we plan to establish a systematic and stable optimization scheme, serving as a foundation for future hardware upgrades. Additionally, we will continue experimenting with and evaluating the performance of different visual recognition models on the Raspberry Pi, striving to identify a more suitable and efficient model to enhance system performance. Furthermore, we will persist in adapting external hardware components, such as implementing a reliable and continuous power supply, and designing a more refined and scientifically constructed casing to improve aesthetics and further enhance the system's adaptability to extreme environmental conditions.

6.4 Ethical Considerations

This Raspberry Pi-based vehicle recognition system introduces several ethical considerations that must be carefully evaluated. First, privacy is paramount, as continuous video monitoring could inadvertently capture sensitive personal data. It is essential to implement strict access controls and anonymization measures to safeguard individual privacy. Second, transparency is crucial; stakeholders should be informed clearly about how data

is collected, processed, and stored. Third, data security must be rigorously maintained to protect against unauthorized access or misuse. Additionally, biases in visual recognition models could lead to unfair treatment or inaccuracies in detection, requiring comprehensive model validation to ensure fairness and accuracy. Lastly, the environmental impact of continuous operation and hardware disposal should be considered, prompting sustainable practices for system maintenance and hardware recycling.

References

[1] IEEE. ""IEEE Code of Ethics"." (2016), [Online]. Available: https://www.ieee.org/about/corporate/governance/p7-8.html (visited on 02/08/2020).

[2] ACM. ""ACM Code of Ethics and Professional Conduct"." (2018), [Online]. Available: https://www.acm.org/code-of-ethics (visited on 07/09/2023).

[3] M. Anandhalli and V. P. Baligar, "A novel approach in real-time vehicle detection and tracking using raspberry pi," *Alexandria Engineering Journal*, vol. 57, no. 3, pp. 1597–1607, 2018, ISSN: 1110-0168. DOI: 10.1016/j.aej.2017.06.008. [Online]. Available: https://doi.org/10.1016/j.aej.2017.06.008.

[4] Ultralytics. "YOLOv5 Documentation." (2023), [Online]. Available: https://docs.ultralytics.com (visited on 03/10/2025).

[5] O. Team. "OpenCV: Image Processing (Imgproc Module)." (2024), [Online]. Available: https://docs.opencv.org (visited on 03/22/2025).