WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

Development of an Intelligent Traffic Control using Optimal SSD Algorithm for Two Cross Intersections

Bashir Muhammad [a]*, E.C. Anene [a], A.L. Amoo [b]

*[a]*Department of Mechatronics and Systems Engineering, Abubakar Tafawa Balewa University, Bauchi, 740272, Nigeria*
[a] Department of Electrical and Electronics Engineering, Abubakar Tafawa Balewa University, Bauchi, 740272, Nigeria
[b] Department of Electrical and Electronics Engineering, Abubakar Tafawa Balewa University, Bauchi, 740272, Nigeria

A B S T R A C T

This research presents the development of an intelligent traffic control system by feature map extraction using visual geometry group (VGG-16) model and leveraging the Single Shot Multibox Detector (SSD) that uses a convolutional neural network (CNN) which is a deep learning algorithm optimized for real-time vehicle detection, classification, and localization at two cross intersections. The system aims to enhance urban traffic flow by accurately detecting, classifying, and localizing six classes of vehicles; cars, motorcycles, tricycles, bicycles, buses, and trucks using images captured from surveillance cameras. The Single Shot Multibox Detector (SSD) model was trained on Google Colaboratory for 100 epochs with a batch size of 32, achieving a detection accuracy of 84%, precision of 89%, recall of 88%, and an F1-score of 89%. Testing was conducted on a Raspberry Pi 3 Model B using a custom Python script, showcasing the model's robust performance in live environments. A MATLAB-based simulation (m-file) was developed to control traffic flow at a two-junction intersection with eight traffic lights. The simulation effectively demonstrated vehicle behaviour under dynamic traffic conditions, incorporating vehicle density-based signal switching to evaluate throughput, delay, queue length, intersection capacity, and green time utilization was evaluated. The "Platoon Concept" was also applied for improved vehicle flow. The study contributes significantly to the field of intelligent transportation systems by combining deep learning with traffic simulation, offering a scalable and adaptive solution for real-time traffic monitoring, control, and potential law enforcement integration. Recommendations for future work include expanding vehicle class datasets, incorporating emergency vehicle prioritization, and integrating advanced features such as license plate recognition and pedestrian detection to support smart city applications.

Keywords: Density Based Traffic Control, Image Processing, CNN, Raspberry Pi, MATLAB M-File.

1. Introduction

Historical records show that the first automated traffic control systems were developed by inventors Leonard Casciato and Josef Kates, and were implemented in Toronto, Ontario, Canada, in 1954 (James, 2018). Traffic lights, positioned at intersections, regulate vehicle movement using a system of coloured signals. Each colour conveys a specific instruction to drivers: red indicates "stop," green signals "go," and yellow warns to prepare to stop or proceed with caution. These signals manage both vehicular and pedestrian flow by prioritizing different traffic movements. As a result, they help ensure smoother traffic flow, enhance the capacity of intersections, and significantly reduce the frequency and severity of certain collisions, particularly right-angle crashes.

Efficient transportation and rapid travel infrastructure are vital to a country's economic growth. Errors and system failures can lead to long delays, wasted fuel, and financial losses. To address this, it is essential to implement a fast, reliable, and effective traffic control model. Urban traffic management is becoming an increasingly complex issue in many nations. As vehicle numbers continue to rise, transportation authorities must explore innovative solutions to meet these growing challenges. One promising approach to enhancing traffic flow and improving the overall performance of modern transport systems is the use of artificial intelligence and advanced control technologies Kawade et al. (2018).

Traffic congestion remains one of the most pressing challenges in metropolitan areas, prompting various efforts to alleviate it. A crucial first step in addressing congestion is understanding its characteristics, as this insight is key to determining the most effective solutions. Congestion impacts both passenger and freight transport affecting actual and perceived mobility and is often tied to historically high levels of accessibility and travel demand. Traffic jams contribute to environmental pollution and psychological stress, lead to wasted time and fuel, lower productivity, and impose significant costs on society (Rao & Rao, 2012).

Traffic congestion is a major issue in urban areas. As cities continue to expand, the number of vehicles is increasing at a rate that surpasses the average, leading to significant loss of travel time due to traffic jams. One key factor contributing to congestion is the widespread use of fixed-timing traffic lights (Atta et al., 2020).

Road traffic congestion has become a real-time concern in both developed and developing nations. Effectively analysing and addressing this issue requires the implementation of modern intelligent transportation systems, which have shown at least partial success in recent years. Transportation is a vital aspect of daily life, yet the rapid pace of population growth and ongoing migration from rural to urban areas are putting immense pressure on urban road infrastructure (Cirp et al., 2020).

In traffic control, electromechanical signal controllers often seen as basic, outdated clockwork systems operate on fixed time intervals. For example, a one-minute cycle may be insufficient to manage the movement of vehicles and pedestrians efficiently and safely. Since traditional traffic lights change at pre-set times regardless of actual traffic conditions, they often remain green even when no vehicles are present, resulting in an open-loop system. This inefficiency leads to unnecessary delays and increases the risk of conflicts between vehicles and pedestrians. To overcome these limitations, there is a need for intelligent, advanced automated control systems that can dynamically adjust signals to minimize delays and improve traffic flow at intersections. Such systems create a closed-loop environment. In line with this goal, this research will employ a robust artificial intelligence-based traffic control approach, integrating digital image processing algorithms to detect vehicle density and optimize traffic signal timing accordingly.

2. Literature Review

Kawade et al. (2018) developed an intelligent traffic control system using Open Source Computer Vision (OpenCV) and image processing techniques. The system operates with an 8051 microcontroller, which efficiently converts digital data into binary format. Key technologies applied include blob detection and color thresholding, while the Viola-Jones algorithm was used for object detection. An additional feature allows the system to detect fires on the road and respond accordingly by taking action in the affected lane. The project was implemented using the Python programming language. To ensure cost-effectiveness, OpenCV was chosen as the primary software tool, reducing the overall project expenses. The system successfully monitored traffic at intersections using a single camera. However, implementation in rural areas was not feasible due to the low volume of vehicles and the absence of significant traffic concerns.

Saputri et al., (2024) proposed the design of an intelligent transportation system centered on traffic flow estimation. Their research explored the application of the You Only Live Once (YOLO) v7 algorithm to estimate traffic conditions by developing a vehicle counting and velocity tracking system. This system was built using a custom dataset comprising car images and CCTV footage from Malang's traffic, serving as a case study. The methodology involved vehicle detection and tracking, with a custom dataset of 1,000 car images sourced from Google Images; 800 images were used for training, 100 for validation, and 100 for testing. Their findings revealed that at an Intersection over Union (IoU) threshold of 0.65 (the default setting), the custom YOLO v7 model achieved an average precision of 61.3%, outperforming the pre-trained model's 51.4%. When the IoU was adjusted to 0.5, the custom model's precision increased to 78.6%, compared to 69.7% for the pre-trained version. Nonetheless, the limited size and quality of the dataset led to reduced accuracy and consistency in the custom model's performance.

The study by Adeyemi and Salami (2021) presents the design and implementation of a traffic lane congestion monitoring and control system using the YOLO (You Only Look Once) neural network. The system integrates three core modules: vehicle detection, signal switching, and simulation. Using a custom-trained YOLO model, the system identifies various vehicle classes from live camera feeds and dynamically adjusts traffic signal timings based on real-time traffic density. Implemented on Google Colab with image datasets labeled using Roboflow and simulated via Pygame, the system achieved 96% precision, 62% recall, and 86% overall accuracy. It demonstrated the ability to reduce unnecessary waiting time, congestion, and fuel consumption by adaptively assigning longer green signals to busier lanes. The study also highlights limitations due to small datasets and suggests the system’s potential for real-world application with further enhancements.

Mochamad Aditya Irawanto and Casi Setianingsih (2022) proposed a method for detecting a traffic density detection system utilizing image processing and a Pin Hole algorithm. The system employs a fixed camera to capture video footage, which then undergoes several pre-processing steps including grayscale conversion, Contrast Limited Adaptive Histogram Equalization (CLAanHE), edge detection using the Canny method, a low pass filter (Bilateral filter), and image segmentation. The Pin Hole algorithm is then applied to count vehicles within a designated Region of Interest (ROI) and classify traffic density into categories such as "Traffic Jam," "Crowded Smoothly," or "Fluent". Testing demonstrated an accuracy of 85% using a 64x64 grid division and effective detection within light intensity values ranging from 5430 to 41379 LUX. The study also involved converting pixel data to metric units to validate detection against actual road conditions and found that a 1:4 scale was achieved. The authors conclude that this system offers a viable alternative for assessing traffic congestion, although further modifications are needed for nighttime detection.

Candradewi *et al* (2021) worked on an intelligent traffic monitoring system for vehicle type classification using a multi-class Support Vector Machine (SVM). The system comprises three primary stages: vehicle detection, feature extraction, and classification. Vehicle detection is accomplished using a Mixture of Gaussians (MOG) method combined with HOG-SVM, while feature extraction utilizes geometric properties, Histogram of Oriented Gradient (HOG), and Local Binary Pattern (LBP) to characterize vehicles into eight distinct classes. The study highlights SVM's strength in handling detailed and statistical computations. Experimental results show a vehicle detection accuracy of 93.76% (using HOG parameters of cell size 4x4 and block size 2x2) and an overall mean recognition rate for vehicle classification of 91.31%, with a mean precision of 77.32% and mean recall of 75.66%. This computer vision-based approach aims to offer more comprehensive and current traffic data than traditional manual methods.

Awoyera et al. (2019) introduces an Intelligent Traffic Control System (ITCS) utilizing an Adaptive Neuro-Fuzzy Inference System (ANFIS) to manage traffic flow in developing cities. The system employs CCTV cameras for real-time traffic lane photography and an image processing unit to assess traffic volume, subsequently assigning lane priority based on this data. The ANFIS-based controller learns from historical traffic data to predict future conditions and optimize green light extensions, aiming to alleviate congestion and reduce delays. A comparative analysis showed that the ANFIS controller

outperformed traditional fixed-time, fuzzy logic, and neural network-based controllers in terms of average queue length, waiting time, and delay. The paper concludes that the ANFIS controller's adaptability to real-time traffic conditions results in better performance and accuracy, and suggests future development to encompass networked junctions and diverse traffic type detection.

3. Methodology

This covered both the materials and methods used in the development and implementation process. These materials include hardware and software.

3.1 Hardware

The hardware comprises of the digital imaging camera which was used for the snapshot of vehicles, and the raspberry pi which was used for detection, classification, and localization of the vehicles.

3.1.1 Digital Imaging Camera

The system utilized image and video-based digital cameras capable of capturing high-definition still photographs with an 8-megapixel (MP) dual camera setup on the rear comprising a primary 8MP camera and a secondary Quarter Video Graphics Array (QVGA) lens. Six of these cameras were mounted side by side adjacent to the traffic lights. Although this model was the most expensive, it proved to be among the most efficient in performance. These cameras supported automated image and video analysis for traffic management, carried out in three key stages:

a) Preprocessing

In this phase, the system filtered out irrelevant elements such as shadows, variable lighting, rain, and motion-induced blur. This reduction of noise was essential for enhancing image clarity and ensuring more accurate analysis.

b) Processing

Next, the raw or preprocessed images were analyzed to identify patterns and extract meaningful features. A structured framework was developed to systematically process the captured traffic control zone images. The vehicle detection algorithm employed bounding boxes of various aspect ratios to estimate vehicle density.

c) Post-processing

In the final step, the system detected, identified, and counted vehicles moving into traffic control zones. Based on the percentage of vehicles present in each lane within a set time frame, traffic light signals were activated accordingly. Figure 6 illustrates the Raspberry Pi with a blue Ethernet cable connecting it to the camera system.

3.1.2 Block Diagram

The block diagram for the vehicle detection, classification and localization is shown in figure 1. It is a diagram showing the major parts of the system and their functions. The power supply supplies power to the camera, Raspberry Pi, and the personal computer (PC).

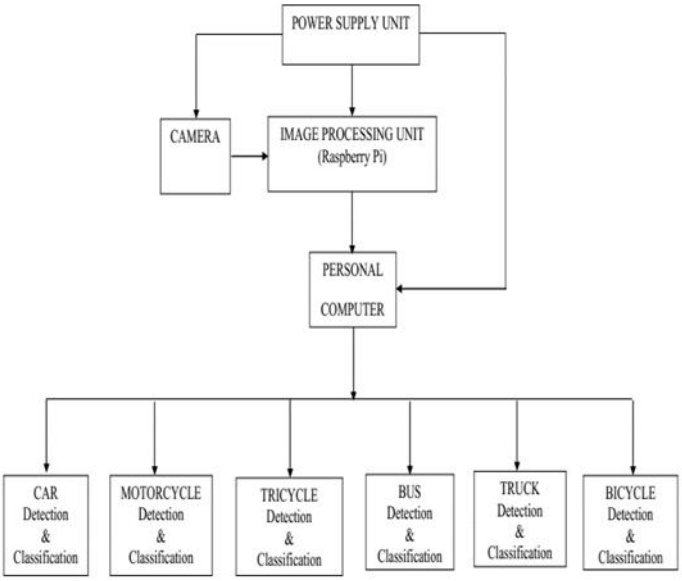

Figure 1: Block Diagram for Vehicle Detection, Classification and Localization

3.2 Training and Testing Datasets Using the SSD Algorithm

The Single Shot Multibox Detector (SSD) algorithm was employed for training and testing, utilizing the Pascal VOC (PASCAL Visual Object Classes Challenge) dataset as a benchmark for object detection, semantic segmentation, and classification tasks. In this study, vehicles within traffic control zones were identified by assigning bounding boxes to each one in the input images. Each bounding box was characterized using four parameters: the center coordinates (bx, by), width (bw), height (bh), and the vehicle class (e.g., car, bicycle, etc.). Additionally, the bounding box coordinates [x-min, y-min, x-max, y-max] were extracted based on the image's pixel resolution, with (x-min, y-min) denoting the top-left corner and (x-max, y-max) indicating the bottom-right corner of the bounding box.

3.3 Traffic Control Zone

The traffic control zone refers to the designated area within each lane where vehicles are monitored to determine when to allocate movement time, based on reaching a predefined density threshold. This threshold is detected using the camera system in conjunction with the Single Shot Multibox Detector (SSD) algorithm. As illustrated in figure 2 (b), the zone is defined as a rectangular region containing multiple vehicles, each enclosed within a bounding box that serves as its Region of Interest (ROI). This zone is located in the foreground of the intersection, positioned close to both the traffic lights and the cameras. The intelligent traffic control was implemented using a case study at the former two cross intersections of Wunti Market Bauchi State, Nigeria. This is shown in figure 2 (a).

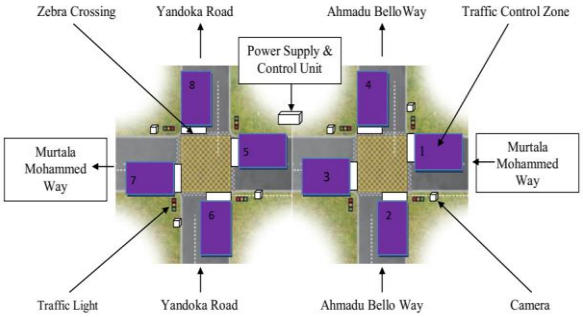

Figure 2 (a): Two Cross Intersections at Wunti Market Bauchi, Nigeria

Figure 2 (b): Traffic Control Zone at Yandoka Road Bauchi, Nigeria

3.4 Software

The software used were python programming language and matrix laboratory (MATLAB) R2023b. The python program was developed and run with the raspberry pi 3 model B via the computer and other supporting software like the Bonjour Print Services and the Virtual Network Computing (VNC) viewer. The Bulk Rename Utility (BRU) software was also used in renaming the dataset according the six classes of vehicles i.e. car, motorcycle, tricycle, bus, truck, and bicycle respectively.

3.5 Concept of Platoon Movement of Vehicles at the Corridor

The platoon-based traffic control strategy for two cross intersections, where a group of vehicles (often autonomous) travels in a coordinated manner, maintaining consistent spacing. A central system or the vehicles themselves manage the platoon to improve traffic flow using real-time data. To support this, vehicle detection ranges for the left eastbound (L_EB) and right westbound (R_WB) directions are doubled to prioritize corridor traffic and reduce overflow. Traffic lights are also synchronized: L_WB and R_WB turn green together in one cycle, while L_EB and R_EB do so in another.

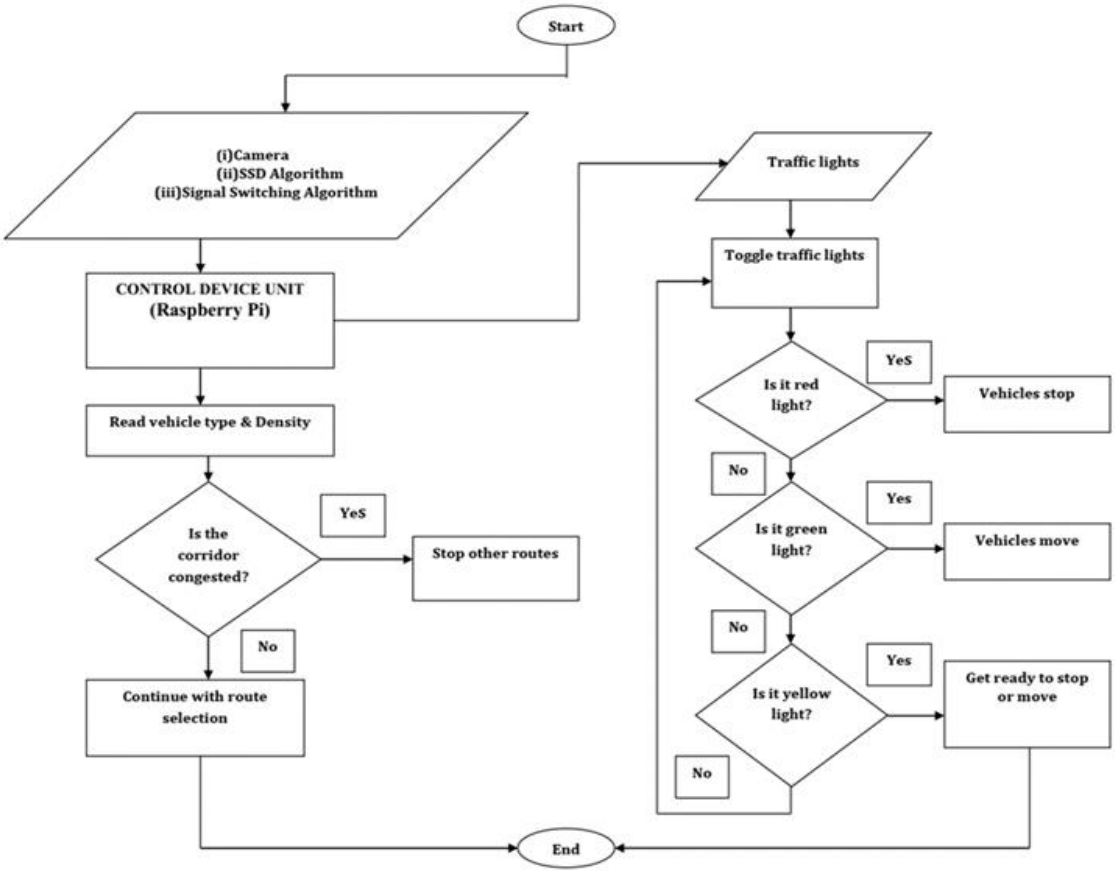

Figure 3: Traffic Flow Control Algorithm

4. Dataset Analysis used for Detection, Classification, and Localization

Vehicle detection identified the presence of vehicles in the image. Vehicle classification categorized those vehicles into different types (car, motorcycle, truck, bus, bicycle) based on their characteristics. Vehicle localization determined the precise position or location of these classified vehicles within the scene. The performance metrics used were Precision, Recall, F1-Score, and accuracy. The total number of dataset and the number containing the training, validation, and testing for each class (car, motorcycle, tricycle, bus, truck, bicycle) was recorded in table 1.

Table 1 - Training, Validation, and Testing Datasets for the Six Classes of Vehicles.

Object Detector	Class	Total Number of Dataset (100%)	No of Training Images (60%)	No of Validating Images (20%)	No of Test Images (20%)	Epoch
VGG-16 Based SSD Algorithm	Car	8144	4886	1629	1629	100
	Motorcycle	1402	842	280	280	100
	Tricycle	1145	687	229	229	100
	Bus	1085	651	217	217	100
	Truck	1054	632	211	211	100
	Bicycle	1092	656	218	218	100

5. Results

This presents the results from training and developing the model for vehicle detection, classification, and localization. It also outlines the model's testing process and the steps taken to achieve the desired outcomes, evaluated using performance metrics. Additionally, it includes the results obtained from the traffic control simulation conducted in MATLAB m-file.

5.1 Vehicle Detection and Performance Metrics

From table 2 it can be deduced that the overall accuracy is the average accuracy of the six classes of vehicles which is 0.84. Mean average precision of all the classes is 0.89 and average recall for all the classes is 0.88. The bar chat of F1-Score Accuracy by vehicle class is shown in figure 4. Mean Average Precision is a metric used to evaluate the performance of object detection models, calculated by averaging the average precision (AP) across all classes, where average precision (AP) is the area under the precision-recall curve for a specific class. For a dataset that contains k classes of objects, the mean average precision (mAP) of a detector at an intersection over union (IoU) threshold t is defined in equation 1. For this study the threshold set was mAP≥50%.

Table 2 - Performance Metrics for the Six Classes of Vehicles.

Class	Precision	Error in Precision	Recall	Error in Recall	F1-Score	Error in F1-Score	Accuracy
Tricycles	0.50	0.50	0.47	0.53	0.48	0.52	0.33
Trucks	0.97	0.03	0.95	0.05	0.96	0.04	0.93
Cars	0.97	0.03	0.96	0.04	0.96	0.04	0.93
Motorcycles	0.98	0.02	0.96	0.04	0.97	0.03	0.94
Buses	0.98	0.02	0.96	0.04	0.97	0.03	0.94
Bicycles	0.98	0.02	0.96	0.04	0.97	0.03	0.94

$$\text{mAP@}t = \frac{1}{k}\sum_{0}^{k-1}\text{AP@}t \qquad \cdots (1)$$

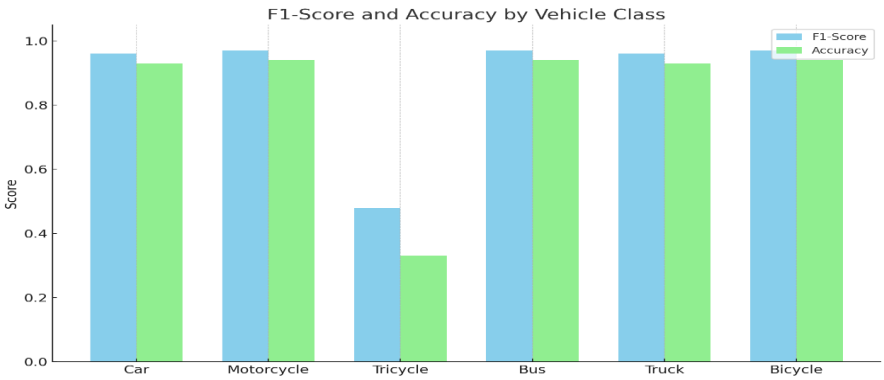

Figure 4: Bar Chart of F1-Score and Accuracy by Vehicle Class

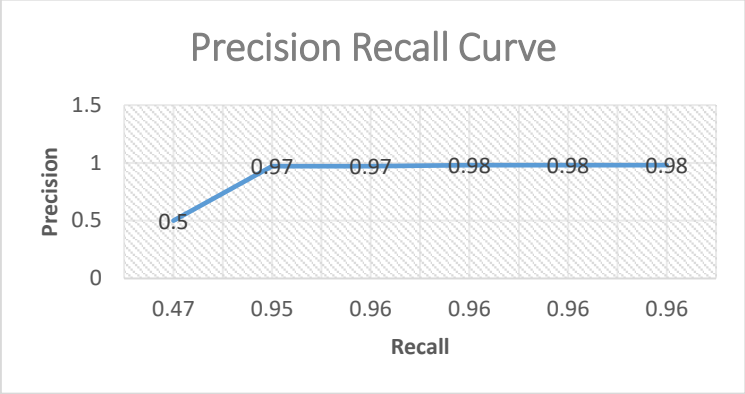

Figure 5: Precision Recall Curve

5.1.1 Confidence Score

The confidence score reflects how certain the model is that an object detected within a bounding box belongs to a specific class, typically ranging from 0 to 1 (or 0% to 100%). In this study, we considered predictions from our object detection models to be reliable when the confidence score was 0.5 (50%) or higher. A lower score indicates reduced certainty. Figure 6(a) displays vehicles captured in night time traffic, while Figure 6(b) shows the corresponding confidence scores for each vehicle class, derived from detection, classification, and localization processes via the Virtual Network Computing (VNC) Viewer. Instances where vehicles had confidence scores below 0.5 were primarily due to occlusion.

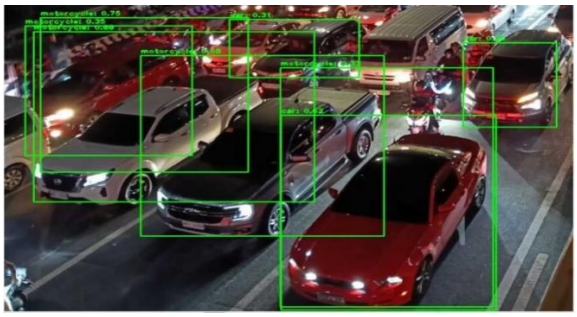

Figure 6 (a): Detection, Classification, and Localization of Vehicles at Night

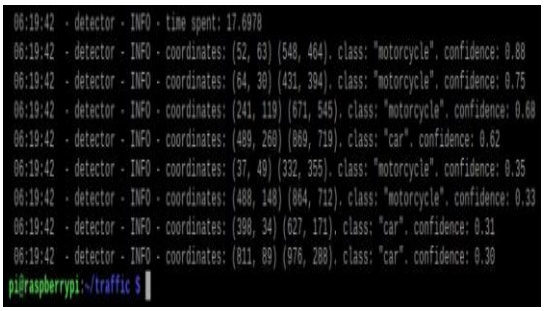

Figure 6 (b): Virtual Network Computing (VNC) Viewer

5.2 Simulation and Performance Metrics

The intelligent traffic control simulation in MATLAB began with defining the simulation environment, which involved modelling traffic flow comprising vehicles, roads, and traffic signals. The control algorithm was developed using a MATLAB m-file, through which the intelligent traffic control logic was implemented. The model was then executed by simulating traffic flow under the guidance of the developed control algorithm.

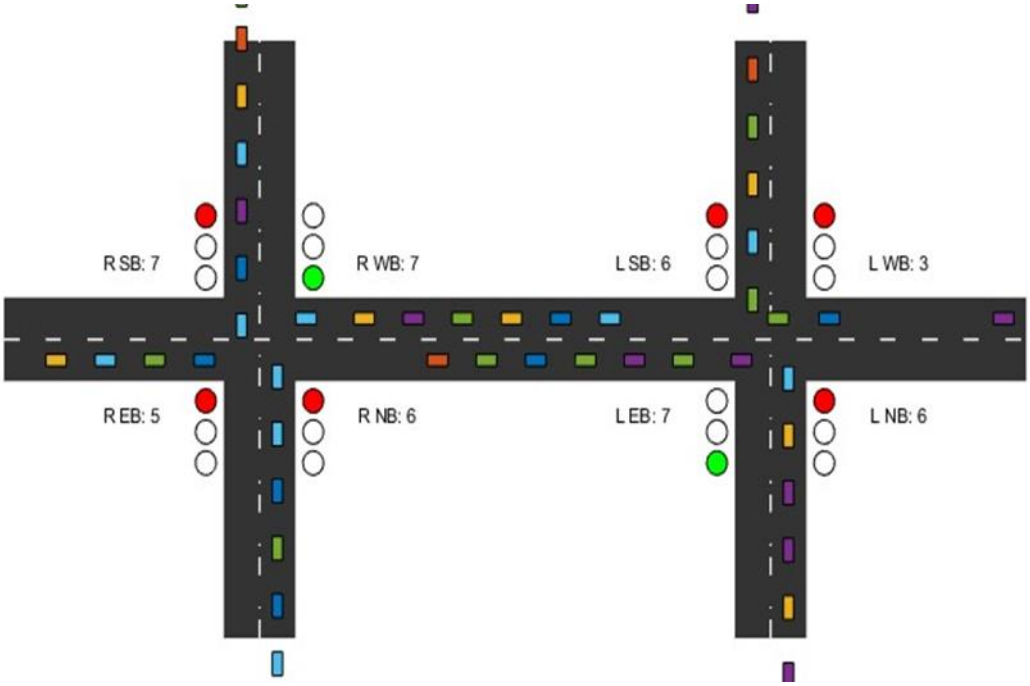

Figure 7: Movement of Vehicles Across the Traffic Lanes with Traffic Lights “Left Eastbound” and “Right Westbound” Green

5.2.1 Computational Throughput (Simulation Speed/Performance)

This refers to how fast the simulation runs on the computer, typically measured in frames per second (FPS) or simulation steps per unit of real time. pause(0.02);: This line at the end of the while loop attempts to regulate the simulation speed. A pause of 0.02 seconds aims for approximately 50 FPS (1 second / 0.02 seconds/frame = 50 frames/second).

5.2.2 Average Vehicle Delay

To calculate the average delay, we sum up all the delays and divide by the number of cars that have exited.

```
% At the end of the simulation (or in a periodic check)
if ~isempty(delays)
    avgDelay = mean(delays);
averageDelay = totalDelay / numCarsExited;
    fprintf('Average Vehicle Delay: %.2f frames\n', avgDelay);
else
    fprintf('No vehicles exited yet.\n');
end
```

5.2.3 Queue Length

Queue length is the number of cars stopped at each approach. The queue was set to 0.01.

```
if cars(i).v_current < 0.01 % Car is stopped
```

5.2.4 Travel Time

The calculate the average travel time, we accumulated the travel times and divided by the number of cars:

```
% After the simulation loop
if numCars > 0
```

```
    averageTravelTime = totalTravelTime / numCars;
    fprintf('Average travel time: %.2f frames\n', averageTravelTime);
else
    fprintf('No cars exited the simulation.\n');
end
```

5.2.5 Intersection Capacity

To calculate the intersection capacity, we needed to track the number of cars that pass through the intersection per unit time.

```
% --- 1. Car Counting for Density ---
    % Initialize counts for each of the 8 approaches
    counts.L_EB = 0; counts.L_WB = 0; counts.L_NB = 0; counts.L_SB = 0;
    counts.R_EB = 0; counts.R_WB = 0; counts.R_NB = 0; counts.R_SB = 0;
```

5.2.6 Green Time Utilization

To calculate the green time utilization, we need to track the time when the traffic light is green and the number of cars that pass through the intersection during that time.

```
% Frame counter for light updates
frameCount = 0;
updateInterval = 400; % Update lights every X frames (e.g., 300 frames * 0.02s/frame = 8s green/red time)
yellowDuration = 75;  % Frames for yellow light (e.g., 75 frames * 0.02s/frame = 1.5s)
```

6. Conclusion

This study developed an intelligent traffic control system using the Single Shot Multibox Detector (SSD) deep learning algorithm to detect, classify, and localize various vehicle types from surveillance images. Implemented and trained on Google Colab and tested on Raspberry Pi 3 Model B, the system achieved high accuracy with an 84% detection rate, 89% precision, and 88% recall. The MATLAB-based simulation modelled vehicle movement at two intersecting road junctions using eight traffic lights, with control based on vehicle density and a signal switching algorithm. The system effectively managed traffic flow, demonstrated through metrics such as queue length, delay, and intersection capacity, while incorporating the "Platoon Concept" for vehicle coordination. Future recommendations include expanding the dataset, improving multi-class detection, integrating features like emergency vehicle prioritization, license plate recognition, and traffic violation detection for enhanced real-time traffic management.

7. REFERENCES

Adeyemi, T. O., & Salami, T. H. (2021). Design of a Traffic Lane Congestion Monitoring and Control System using YOLO Neural Network Approach. *JOURNAL OF SCIENCE TECHNOLOGY AND EDUCATION 9(4), DECEMBER, 2021*, *9*(4), 205–215.

Atta, A., Abbas, S., Khan, M. A., Ahmed, G., & Farooq, U. (2020). An adaptive approach : Smart traffic congestion control system. *Journal of King Saud University - Computer and Information Sciences*, *32*(9), 1012–1019. https://doi.org/10.1016/j.jksuci.2018.10.011

Awoyera, O. O., Sacko, O., Darboe, O., & Cynthia, O. C. (2019). Anfis-Based Intelligent Traffic Control System (ITCS) for Developing Cities. *Journal of Traffic and Logistics Engineering*, *7*(1), 18–22. https://doi.org/10.18178/jtle.7.1.18-22

Candradewi, I., Harjoko, A., Alldino, B., & Sumbodo, A. (2021). Intelligent Traffic Monitoring Systems : Vehicle Type Classification Using Support Vector Machine. *International Journal Of Artificial Intelegence Research*, *5*(1), 78–90. https://doi.org/10.29099/ijair.v5i1.201

Cirp, P., Tartibu, L. K., Okwu, M. O., F, U., Stief, P., Siadat, A., Olayode, I. O., Tartibu, L. K., Okwu, M. O., & Uchechi, U. F. (2020). ScienceDirect ScienceDirect ScienceDirect Intelligent transportation systems , road Intelligent transportation systems , intersections and traffic congestion Johannesburg : intersections and traffic congestion in in Johannesburg : a a systematic review A . *Procedia CIRP*, *91*, 844–850. https://doi.org/10.1016/j.procir.2020.04.137

Kawade, D., Deshmukh, S., Gamare, S., & Sankhe, P. A. (2018). Smart Traffic Control Using Opencv. *IOSR Journal of Engineering*, *13*, 1–4.

Mochamad Aditya Irawanto, Casi Setianingsih, B. I. (2022). DETECTION OF TRAFFIC DENSITY WITH IMAGE PROCESSING USING PIN HOLE ALGORITHM. *IIUM Engineering Journal*, *23*(1), 244–257.

Rao, A. M., & Rao, K. R. (2012). MEASURING URBAN TRAFFIC CONGESTION – A REVIEW. *International Journal for Traffic and Transport Engineering*, *2*(4), 286–305.

Saputri, H. A., Avrillio, M., Christofer, L., Simanjaya, V., & Alam, I. N. (2024). ScienceDirect ScienceDirect Implementation of YOLO v7 algorithm in estimating traffic Implementation of YOLO flow v7 in algorithm Malang in estimating traffic flow in Malang. *Procedia Computer Science*, *245*(2022), 117–126. https://doi.org/10.1016/j.procs.2024.10.235