Adaptive traffic light control using vision-based deep learning for vehicle density estimation

Weerasak Karoon
Graduate School of Applied Statistics, National Institute of Development Administration
weerasak.karo@stu.nida.ac.th

Peeranut Chuasuai
Graduate School of Applied Statistics, National Institute of Development Administration
peeranut.chu@stu.nida.ac.th

Pearploy Thipprasert
Graduate School of Applied Statistics, National Institute of Development Administration
pearploy.thi@stu.nida.ac.th

Nachasa Khongchu
Graduate School of Applied Statistics, National Institute of Development Administration
supharak.kho@stu.nida.ac.th

Piyaboon Kunakornjittirak
Graduate School of Applied Statistics, National Institute of Development Administration
piyaboon.kun@stu.nida.ac.th

Thitirat SIRIBORVORNRATANAKUL*
Graduate School of Applied Statistics, National Institute of Development Administration
thitirat@as.nida.ac.th

ABSTRACT

In today's urban landscape, traffic congestion poses a significant and far-reaching problem impacting cities, economic development, and individual well-being. The root of this challenge lies in the ineffective traffic light management system. In this study, we integrated intersection videos captured by cameras into our system, aiming to improve solutions by evaluating two vehicle detection methods: background subtraction (MOG) and YOLOv3. YOLOv3 outperformed MOG in accuracy, leading to its adoption. We employed the DeepSORT algorithm for vehicle tracking and counting, crucial for determining green light duration. Using Arduino, we controlled the green light based on these calculations. Our experiments confirmed YOLOv3's superiority in vehicle detection, while our prototype system demonstrated proficiency in detection, counting, and green light duration calculation. However, room for improvement remains in vehicle type classification.

CCS CONCEPTS

• **Computing methodologies**; • **Artificial intelligence**; • **Computer vision**;

KEYWORDS

Computer vision, deep learning, adaptive traffic lights, YOLOv3, background subtraction

ACM Reference Format:
Weerasak Karoon, Peeranut Chuasuai, Pearploy Thipprasert, Nachasa Khongchu, Piyaboon Kunakornjittirak, and Thitirat SIRIBORVORN-RATANAKUL*. 2024. Adaptive traffic light control using vision-based deep learning for vehicle density estimation. In *2024 6th Asia Pacific Information Technology Conference (APIT 2024), January 29–31, 2024, Bangkok, Thailand*. ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3651623.3651629

APIT 2024, January 29–31, 2024, Bangkok, Thailand
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-1621-8/24/01
https://doi.org/10.1145/3651623.3651629

1 INTRODUCTION

Traffic congestion gives rise to a variety of problems, including air pollution emitted from vehicle exhaust pipes, noise pollution, physical health issues, and prolonged waiting times, which also contribute to driver stress. Often, most drivers on heavily congested roads find themselves stuck at red lights, waiting for the green light to appear. This issue is of great importance to everyone because it affects our health and the efficient use of our time. Therefore, it requires a solution. The primary cause of this problem is the inadequate control of traffic lights, which is in need of improvement.

Recently, an intelligent traffic light system has been introduced to address this problem. This system can adjust the duration of a traffic signal based on the density of vehicles on the road. The key components of these solutions involve counting and categorizing the types of vehicles on the road, followed by density calculation and the adjustment of signal timing to reduce waiting times. To determine these critical aspects, this paper focuses on adaptive time control for traffic lights, where the duration of green lights is calculated based on the number of vehicles and the distance of traffic congestion. This approach is used to tailor traffic light management to the number of cars waiting, utilizing two methods: classical image processing techniques and deep learning techniques. The challenge addressed in the literature review (Section 2) is the detection of parked cars on the side of the road, as the calculation should consider only the vehicles that are actively in motion. Parked vehicles should not be included in the count. Ultimately, the vehicle count will be used to allocate time for traffic light control.

2 RELATED WORKS

2.1 Classical Image Processing Techniques

Several papers have applied classical image processing techniques for vehicle detection. For example, in 2020, [5] presented a method for real-time traffic-based vehicle detection using background subtraction with truncation thresholds, erosion for noise reduction, dilation for shape enhancement, and an adaptive Gaussian mixture model (MOG2) for background modeling. They conducted experiments under various weather conditions, including morning, daytime, and afternoon, achieving an average accuracy of 96.01

percent. The car counting accuracy is commendable, and the technique has been successfully tested in a real-world environment, demonstrating reliability. First published (online) in 2020, [2] conducted a case study to address traffic congestion on a bridge using two cameras with different frame rates. They applied an optical flow-based approach for vehicle detection, testing it under various weather and traffic conditions. The approach yielded an accuracy of over 90%, and its performance remained unaffected by environmental conditions. Another work in 2020 [3] utilized a background subtraction algorithm to detect vehicles by subtracting the road image without vehicles, revealing the foreground vehicles. Vehicles were counted as they entered or crossed the frame, with counting lines distinguishing vehicles in various positions. The perimeter of the bounding box was used for vehicle classification. Vehicles with a perimeter of less than 300 were classified as bikes, those with a perimeter of less than 500 as cars, and those with a perimeter exceeding 500 as trucks or buses. The accuracy rates for object detection and tracking were 97.1% and 98.4%, respectively.

In 2021, classical image processing methods continue to be employed. [4] proposed an adaptive traffic light control system that captures road photos and converts them to grayscale. They applied five edge detection methods (Log, Sobel, Roberts, Active contour, and Canny) to extract edges. The detection results were used to calculate areas covered by vehicles, which, in turn, determined traffic density by comparing the area in the captured image to the empty areas in reference photos. This ratio was converted to a percentage and used to adjust traffic light durations. For instance, increasing the green light duration for higher percentages. The experiment revealed that Canny provided the best accuracy among the five detectors. Another work [8] captured images using a CCTV camera to identify traffic density for traffic signal control. Image processing methods such as RGB to Gray, resizing, and Canny edge detection were applied. They used the SURF algorithm [6] to calculate the matching percentage between captured images and reference images. A higher matching percentage indicated lower traffic density, leading to the allocation of time for controlling each signal based on the matching percentage. For instance, increasing the green light duration for lower matching percentages. The results demonstrated that the applied SURF algorithm improved detection accuracy.

In summary, classical image processing methods, including background subtraction and optical flow, continue to be utilized due to their implementation advantages and acceptable performance.

2.2 Deep Learning Techniques

In the past decade, deep neural networks (a.k.a., deep learning) have gained prominence as a method for vehicle detection. The most widely embraced approach is YOLO, known for its commendable performance in terms of both accuracy and speed, particularly in real-time applications. Consequently, it has found applications in numerous research papers. For example, [10] introduced a smart traffic control and management system using OpenCV and YOLOv3. This system is designed to detect vehicle movement, identify, track, and count vehicles in real-time by analyzing live video streams from cameras. Upon detecting, classifying, and counting the vehicles in a specific lane, it adjusts traffic lights according to a predefined threshold value, thereby facilitating the safe passage of more vehicles while minimizing waiting times. The use of pre-trained model weights on the COCO dataset ensures rapid inference speed. This system excels in real-time detection of various vehicle types, even in obstructed and high-density traffic scenarios. However, its smooth operation necessitates high-spec hardware to achieve an output exceeding 10 frames per second.

[1] proposed YOLOv3 for object detection and DeepSORT for tracking multiple objects. Their primary objective was to automate non-invasive and cost-effective volume surveys for the Brazilian National Department of Transport Infrastructure. They achieved a precision rate exceeding 90% in global vehicle counting using real-world videos recorded on Brazilian roads. Additionally, they demonstrated that their approach outperformed previously proposed tools with an impressive 99.15% precision in public datasets. Another work of [12] compared the results of vehicle density detection from two methods: 1) using classical techniques, a combination of background subtraction and blob detection, and 2) using YOLOv3. Their tests were conducted with four different cameras. YOLOv3 outperformed background subtraction, delivering accuracy over 90%, recall over 87%, and precision over 81%. However, blob detection was found to be valuable for determining the speed and direction of vehicles, a critical aspect of their challenge.

[7] conducted research based on deep learning models to explore various object identification strategies for recognizing vehicle types. Their evaluation encompassed processing speed, accuracy, and F1-score due to non-uniform data availability. They considered Faster-RCNN, YOLO, and SSD, all capable of real-time processing with high accuracy. Among these models, YOLOv4 exhibited the best performance with a 93% accuracy rate in car model recognition. Faster-RCNN proved the fastest among RCNN models but had limitations in terms of frames per second (FPS) due to CNN utilization, resulting in slow speeds. SSD, while faster, had accuracy limitations and occasional misses. YOLO, though slower than SSD, consistently detected vehicles without misses in each frame.

In conclusion, the control of traffic light durations hinges on two primary variables: the number of vehicles on the road and the type of vehicles. Researchers employ two main approaches—classical image processing techniques, known for their simplicity, and deep learning techniques, appreciated for their accuracy and speed. Our study pertains to real-time scenarios that require swift execution, making YOLO a compelling choice. Nevertheless, in the real world, several undisclosed factors must be considered, such as idle parked vehicles at intersections. The system should be designed to account for these vehicles, allocating appropriate traffic signal durations. Consequently, our work employs both methods to create a more efficient and precise system, with the aim of addressing this challenge.

3 PROPOSED METHODS

3.1 Dataset

This research utilizes two types of datasets. First, we collected our own dataset, comprising videos recorded at the intersection of Siriraj Hospital on May 29, 2022. All four videos were recorded to test the performance of our model. Second, we also used the UA-DETRAC dataset [9], consisting of four videos, to assess YOLO's vehicle counting performance in comparison to background subtraction and deep learning techniques.

3.2 Model Selection

In the proposed method, we initially compare two approaches: classical image processing techniques (Background Subtraction) and deep learning techniques (YOLO), as shown in Figure 1. Starting with background subtraction, we use four traffic videos captured on the road with vehicles as input. Subsequently, the Background Subtractor (MOG2) class from OpenCV, a Gaussian Mixture-based Background Segmentation Algorithm [5], begins creating a background model, in this case, representing the road. As frames are continuously fed into the system, the background model is updated accordingly. Additionally, the threshold is adjusted, and the frame undergoes erosion and dilation operations to determine whether a pixel belongs to the background model. This value represents a distance threshold between the pixel and the background model, indicating the pixels that are well described by the background model. Subsequently, the background is subtracted, and bounding boxes are drawn around the contours' areas to detect the vehicles. This comparison is performed on all four videos.

The second method involves YOLO. In this study, YOLOv3-608 pre-trained weights on the COCO dataset are used for vehicle detection. According to the information provided by the authors of YOLOv3-608, YOLOv3-608 exhibits good performance with an mAP of 57.9. Although it operates at a speed of 20 FPS, which is slower than some other versions, our focus is on accuracy rather than speed. This emphasis on accuracy is because our goal is to detect and count stationary vehicles at intersections rather than moving ones. We also input the same four videos into our YOLO model, allowing us to compare the vehicle detection performance of both methods and select the superior approach for implementing our system.

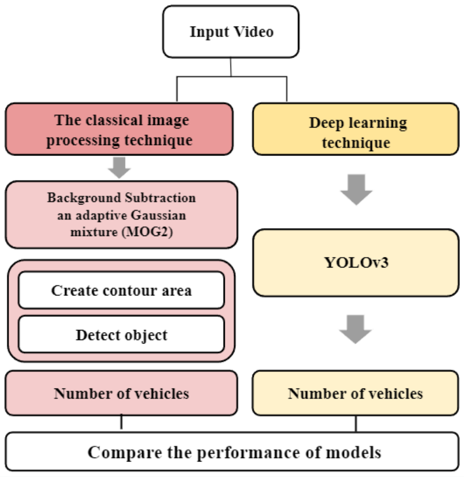
Input Video
The classical image processing technique
Deep learning technique
Background Subtraction an adaptive Gaussian mixture (MOG2)
YOLOv3
Create contour area
Detect object
Number of vehicles
Number of vehicles
Compare the performance of models

Figure 1: Model selection method

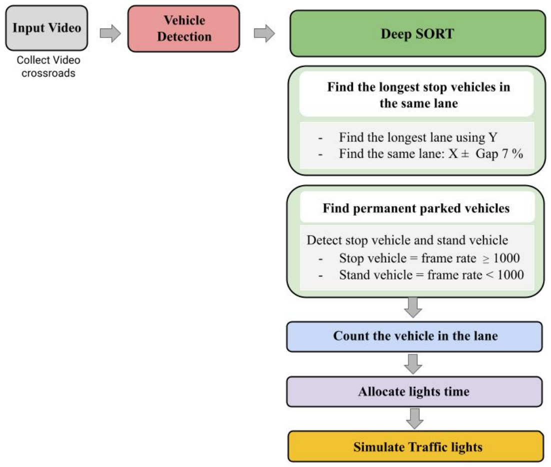
Input Video
Collect Video crossroads
Vehicle Detection
Deep SORT
Find the longest stop vehicles in the same lane
- Find the longest lane using Y
- Find the same lane: X ± Gap 7 %
Find permanent parked vehicles
Detect stop vehicle and stand vehicle
- Stop vehicle = frame rate ≥ 1000
- Stand vehicle = frame rate < 1000
Count the vehicle in the lane
Allocate lights time
Simulate Traffic lights

Figure 2: Traffic light control method

3.3 Traffic Light Control

After comparing the performance of the two methods, it became evident that YOLOv3 outperforms background subtraction, as we will discuss in more detail in Section 4. Consequently, we have chosen YOLOv3 for the implementation of our system, specifically for vehicle detection. Our primary approach to controlling traffic lights revolves around determining the number of vehicles that come to a stop in the same lane. Vehicles that remain stationary for longer periods in the same lane will necessitate a longer duration for a green light. However, it is crucial that these vehicles are not permanently stopped, as this would lead to an inaccurate allocation of green light duration. The process outlined in Figure 2 provides a visual representation of our system.

Find the longest stop vehicles in the same lane: After the vehicles were detected, we applied the DeepSORT algorithm [14] to track them and determine the centroid (x,y) for all tracked vehicles in each frame. Subsequently, we sought to find the maximum y-coordinate, referred to as Y (min), to identify the longest lane. To calculate the appropriate green light duration and determine if the cars were positioned in the same lane, in our experiments, we compared the actual x-coordinate to X ± Gap (7%) from the maximum Y coordinate. If the other x-coordinate fell within the ±7% range, we classified the cars as being in the same lane.

Find permanently parked vehicles: We made a clear distinction between stopped and standing vehicles in this paper. A stopped vehicle is defined as a car that is parked, while a standing vehicle is a car that is about to move. This separation was achieved using the following criteria: a stopped vehicle is one that appears in frames with a rate of 1000 or more frames, whereas a standing vehicle appears in frames with a rate of less than 1000 frames. Additionally, we rechecked whether the centroid of the vehicle changed every 10 frames. Permanent stop vehicles can also be determined by their ID from the DeepSORT tracking results. If we find a vehicle with the same ID in the previous frame, it indicates that the vehicle is permanently stopped and not merely waiting for the green light.

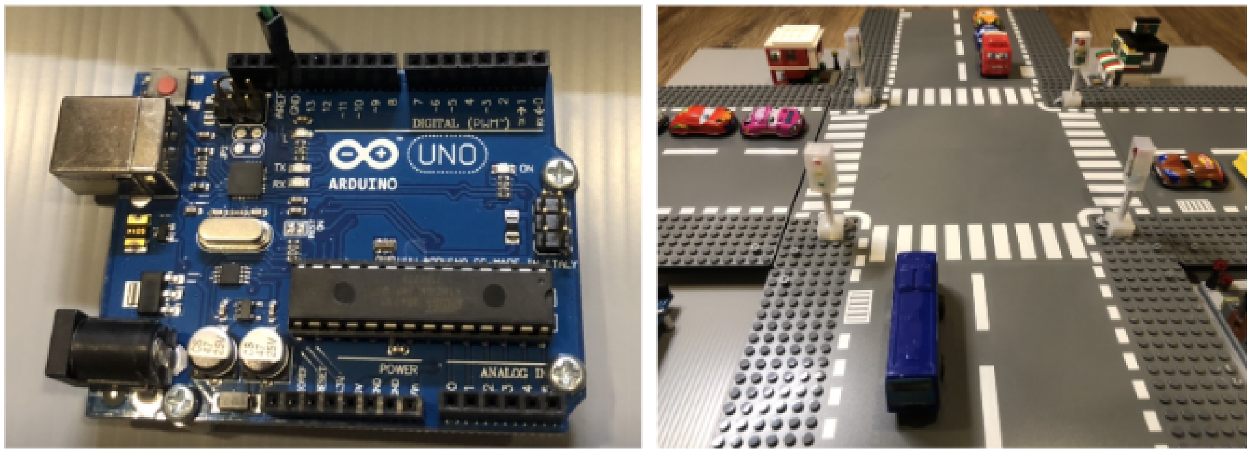

Figure 3: Simulate a traffic light using a Lego model and Arduino

Count the vehicles in the lane: After identifying the lane with the most adjacent cars, excluding parked cars, we proceed to count the total number of cars in that lane based on the X value of their centroids, with variations of no more than 7% from the centroid of the furthest car.

Allocate light time: The duration of the green light is determined by the number of vehicles of each type. In a study from 2012 [13], the average acceleration for each vehicle was obtained: car = 4.65 km/(h*s) (1.293 m/s^2), heavy vehicle = 4.96 km/(h*s) (1.379 m/s^2). In another study from 2018 [11], the average length of each vehicle was determined: car = 3.72 m, bus = 10.1 m, and truck = 7.5 m. These values are utilized to calculate the duration of the green light for each vehicle as written in Equations 1-4. Using these equations, we can calculate the green light durations as follows: car = 2.55 sec, bus = 3.92 sec, and truck = 3.41 sec. For instance, consider the duration of the green light for 3 cars, 2 buses, and 1 truck: t_G = (3 x 2.55) + (2 x 3.92) + 3.41 = 18.9 sec.

$$s = ut + \frac{1}{2}at^2 \tag{1}$$

$$t = \sqrt{\frac{2s}{a}} \tag{2}$$

t = time for moving out from traffic of each vehicle
s = vehicles length (m)
u = speed of the vehicle from a standstill, always zero (m/s)
a = acceleration of each vehicle (m/s^2)

$$t_i = \sqrt{\frac{2\,(s_i + gap)}{a}} \tag{3}$$

$$t_G = \sum (n_i \times t_i) \tag{4}$$

t_i: Duration for Green light of each vehicle type i (sec.)
t_G: Total duration for Greenlight (sec.)
n_i: Number of each vehicle type i gap = 0.5: Constant distance from front vehicle (m)

Simulate traffic lights: After calculating the green light durations (in seconds), the system will operate as a transmitter, sending the results to the Arduino using the asynchronous UART (Universal Asynchronous Receiver-Transmitter) serial data transmission format and the Lego model. It will display results in terms of on-off, green light, red light, and the time calculated based on the number of counted cars (see Figure 3).

4 RESULTS AND DISCUSSION

The experiments were conducted on a computer with 16 GB of DDR4 memory, an AMD Ryzen 7 5800H CPU (running at 3.20 GHz, with the ability to boost up to 4.40 GHz, and equipped with 16 MB of L3 Cache), as well as an NVIDIA GeForce RTX 3060 GPU with 6GB of GDDR6 memory. The code was developed using the Python programming language. The YOLO (You Only Look Once) method from the OpenCV library utilized YOLOv3-608 weights pre-trained on the COCO dataset.

4.1 Model Selection

The vehicle count results for all four videos from both models, background subtraction and YOLOv3, consistently demonstrated that YOLOv3 exhibited superior vehicle detection performance in all videos and was capable of classifying the types of vehicles (as shown in Tables 1 and 2). As example results shown in Figures 4 and 5, we selected YOLOv3 as our vehicle detector to count the number of cars and determine the subsequent green light duration.

4.2 Traffic Light Control

When utilizing YOLOv3 to detect and count the number of cars in Video 4 at the Siriraj Hospital intersection, we successfully detected the vehicles and calculated the green light duration. However, it's worth noting that the accuracy in counting the number of cars in the video is relatively low. Our system cannot check the number of vehicles in every frame due to the rapid data processing time and potential issues with duplicate car counts arising from repeated vehicle detection. This challenge may need to be addressed by implementing more efficient object-tracking methods and reducing the re-tracking of the same object.

Table 1: Performance of the background subtraction model

Input video	Counted manually	Counted by model	Model accuracy
Video 1	7	3	0.429
Video 2	12	1	0.083
Video 3	12	1	0.083
Video 4	19	2	0.105

Table 2: Performance of the YOLOv3 model

Input video	Car type	Counted manually	Counted by model		Model accuracy
			True positive	False positive	
Video 1	Car	13	10	0	0.769
	Motorbike	9	2	0	0.222
	Truck	-	-	-	-
	Bus	0	0	1	0.000
Video 2	Car	7	6	0	0.857
	Motorbike	-	-	-	-
	Truck	0	0	2	0.000
	Bus	0	0	1	0.000
Video 3	Car	8	7	0	0.875
	Motorbike	4	2	0	0.500
	Truck	0	0	1	0.000
	Bus	-	-	-	-
Video 4	Car	9	6	0	0.667
	Motorbike	1	0	1	0.000
	Truck	-	-	-	-
	Bus	0	0	3	0.000

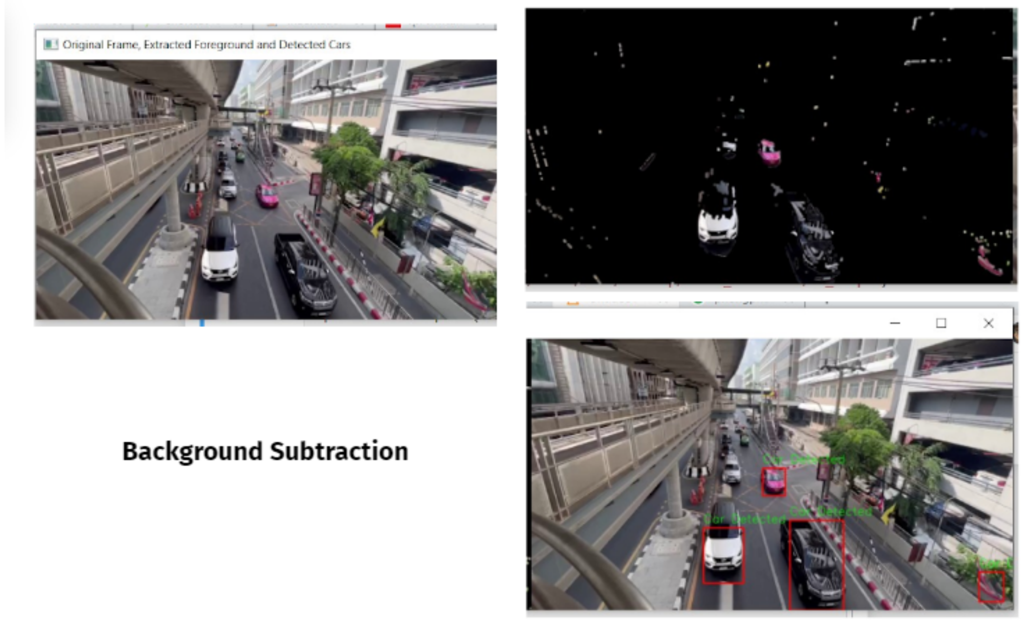

Figure 4: An example result of the background subtraction model

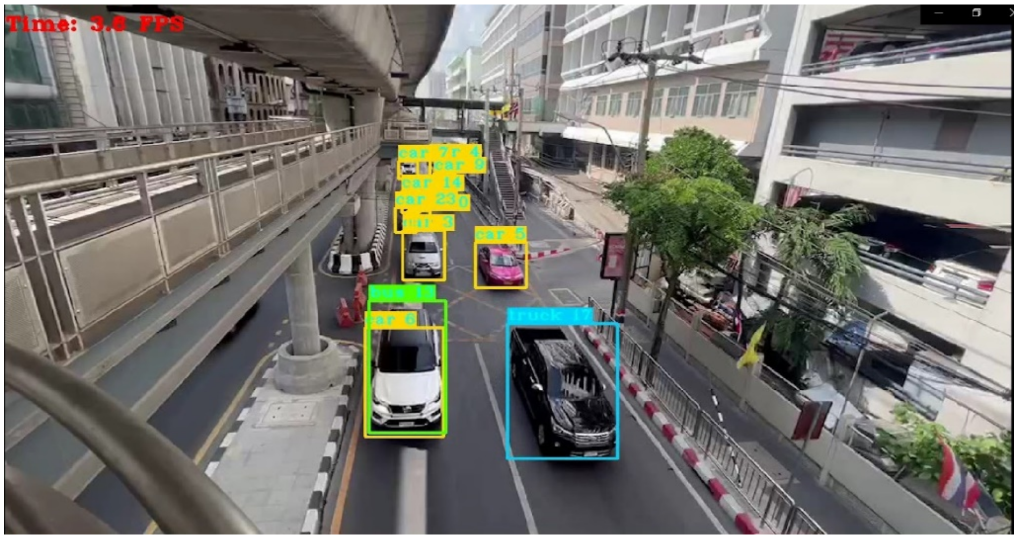

Figure 5: An example result of the YOLOv3 model

5 CONCLUSION

From this study, it is evident that the YOLOv3 method outperforms the background subtraction method in terms of vehicle detection accuracy. We have successfully implemented adaptive green light duration control based on YOLOv3. Our system is capable of detecting and counting vehicles and calculating green light durations at intersections as intended. However, there is room for improvement in vehicle type classification, and further development is needed in the areas of vehicle detection and re-counting. Advanced tracking object algorithms can be employed to address these issues by monitoring changes in the vehicles' positions to avoid duplicate counts.

Furthermore, in addition to enhancing the efficiency of vehicle tracking, there are various ways to improve traffic light control. These include expanding the analysis of vehicles at intersections where both cars stop at red lights and vehicles pass through green lights in different lanes to filter out cars that didn't stop at red lights. Additionally, there is room for improvement in the detection method for tracking vehicles on non-straight roads or at intersections that are not four-way junctions, such as roundabouts or underpasses. Customizing the vehicle type dataset to match specific use cases, such as TukTuk in Thailand and Tram in the UK, can also enhance the system's adaptability for a wide range of scenarios. These suggestions aim to create the most adaptive system for general use cases.

REFERENCES

[1] Adson M. Santos, Carmelo J. A. Bastos-Filho, Alexandre M. A. Maciel, and Estanislau Lima. 2020. Counting Vehicle with High-Precision in Brazilian Roads Using YOLOv3 and Deep SORT. In Proceedings of the SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI). IEEE, Porto de Galinhas, Brazil.

[2] Ameni Chetouane, Sabra Mabrouk, Imen Jemili, and Mohamed Mosbah. 2022. Vision-based vehicle detection for road traffic congestion classification. Concurrency and Computation: Practice and Experience, 34, 7 (March 2022).

[3] Apeksha P. Kulkarni and Vishwanath P. Baligar. 2020. Real Time Vehicle Detection, Tracking and Counting Using Raspberry-Pi. In Proceedings of International Conference on Innovative Mechanisms for Industry Applications (ICIMIA). IEEE, Bangalore, India.

[4] Belinda Chong Chiew Meng, Nor Salwa Damanhuri, and Nor Azlan Othman. 2021. IOP Conference Series: Materials Science and Engineering, 1088.

[5] De Rosal Ignatius Moses Setiadi, Rizki Ramadhan Fratama, and Nurul Diyah Ayu Partiningsih. 2020. Improved Accuracy of Vehicle Counter for Real-Time Traffic Monitoring System. Transport and Telecommunication, 21, 2, 125-133.

[6] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. 2006. SURF: Speeded Up Robust Features. In Proceedings of European Conference on Computer Vision (ECCV), 404-417. Springer, Graz, Austria.

[7] Jeong-ah Kim, Ju-Yeong Sung, and Se-ho Park. 2020. Comparison of Faster-RCNN, YOLO, and SSD for Real-Time Vehicle Type Recognition. In Proceedings of International Conference on Consumer Electronics - Asia (ICCE-Asia). IEEE, Seoul, South Korea.

[8] Lakshmanan M, NVN Jyotika, NivethaP, and Preethi. A.I. 2021. Traffic Light Controller using Image Processing. Turkish Journal of Computer and Mathematics Education (TURCOMAT), 12, 2.

[9] Longyin Wen, Dawei Du, Zhaowei Cai, Zhen Lei, Ming-Ching Chang, Honggang Qi, Jongwoo Lim, Ming-Hsuan Yang, and Siwei Lyu. 2020. UA-DETRAC: A new benchmark and protocol for multi-object detection and tracking. Computer Vision and Image Understanding, 193.

[10] Manish Kumar Singh, Krishna Deep Mishra, and Subrata Sahana. 2021. An intelligent realtime traffic control based on vehicle density. International Journal of Engineering Technology and Management Sciences, 5, 3 (May 2021).

[11] Mithun Mohan and Satish Chandra. 2018. Occupancy time-based passenger car equivalents at unsignalized intersections in India. Current Science, 114, 6, 1346-1352 (March 2018).

[12] C R Rashmi and C P Shantala. 2020. Vehicle Density Analysis and Classification using YOLOv3 for Smart Cities. In Proceedings of International Conference on Electronics, Communication and Aerospace Technology (ICECA). IEEE, Coimbatore, India.

[13] Satish Chandra and Shalinee Shukla. 2012. Overtaking Behavior on Divided Highways Under Mixed Traffic Conditions. Procedia - Social and Behavioral Sciences, 43, 313-322.

[14] Nicolai Wojke, Alex Bewley, and Dietrich Paulus. 2017. Simple online and realtime tracking with a deep association metric. In Proceedings of International Conference on Image Processing (ICIP). IEEE, Beijing, China.