[[1]](#footnote-1)

Smart Traffic Light System Using Computer Vision with Web-based control panel

Student name: Long Ngo Duc - BJ19BDS002

***Abstract -* Traffic lights have been around for more than a century and it has played an essential role to control the traffic flows between roads. Nowadays, the traffic load is increasing so rapidly that the traffic infrastructure cannot accommodate these changes, which leads to an increased potential for traffic jams. The current traffic signal system is quite a dummy in controlling the traffic as it just follows a sequence of time allotted and given the order to control traffic flow. Despite its long existences, there has not been any massive improvement to make it smarter and more suitable for an increasing number of traffic loads. Hence, we need a solution to tackle the problem, and design a system that can optimize the traffic light, balance the traffic loads between lanes and reduce waiting time and resources.**

# **INTRODUCTION**

Traffic jam has always been one of unsettled crisis that every country and city has to face. With the increasing number of people and cars, this issue has become worst. In the United State of America alone, almost $70 million per year lost due to traffic congestion in the business sector out of nearly $300 billion spent on gas. Some attempts have been made to tackle this problem. That includes building bridges to skip crossroad or widening the traffic lane and even constructing more roads to reduce traffic loads in other area. Although these approaches seem to be effective, they are very expensive. Smart Traffic signal can be another approach to tackle this issue even with the tight budget. The growth of deep learning and especially computer vision has allowed us to turn ideas that sound incredibly unrealistic into highly applicable ones. This project will take advantage of computer vision and focus on detecting the number of cars(vehicle) on each road and in a crossroad and control the signal accordingly by assigning the appropriate stopping time (red light time) and running time (green light time). This project can introduce a solution that potentially saves energy wasted at stopping time and minimize the waiting time, which therefore can reduce traffic congestions.

The project contains 5 steps. The first step is to design a framework which compensates for the lack of hardware and real-life data. Secondly, a car counting model will be build to measure the traffic loads in each lane. Third step of the project is to simulating real life situation and test to find the best algorithm. Then, simulated data will be used as traffic load and fed into the server. Lastly, server will process data, use the algorithm created to stream onto the front-end.

# **LITERATURE SURVEY**

In [1], an Intelligent Traffic Light Controlling (ITLC) algorithm is introduced. The real-time traffic flow is taken into consideration in this algorithm in order to allocate the time for the green light in each lane. It aims at optimizing the time of the traffic light to reduce wait time and hence, increase traffic fluency.

In [2], This paper introduced object detection which can be used later to detect car (vehicle) and tracking objects in multiple video frames. A real-time computer vision solution on the traffic analysis system is shown, which use traffic video sequence to analyze vehicles in traffic. Vehicle speed estimation, traffic flow direction estimation, traffic density estimation, and car color

determination are the most essential fundamental of traffic analysis using computer vision. Optical Flow Model and Kalman Filtering methods are used to determine the traffic density, vehicle speed, and vehicle color. Experimental analysis for color estimation shows the high accuracy of roughly 86%. The result of this work culminates in object tracking, object detection,

vehicle speed, vehicle color, traffic density, and traffic flow estimation which could then be used for many applications namely traffic control, security, and safety both by government agencies and commercial organizations.

In [3], since traffic congestion has been a big concern in big cities all over the world, this paper proposes a dynamic traffic control system after taking into consideration the traffic density at the intersections by real-time video feeds and image processing. We can do background subtraction methods and for the foreground, detection using the MOG algorithm to keep the count of the cars in each lane. The traffic lights at the intersections will be changed dynamically according to the conditions of traffic that will be detected from the video feeds. In this way, we can optimize the use of traffic light and control the signal according to loads of traffic in each lane.

# **FRAMEWORK**

Theoretically, the framework should look like the graph below where CCTV cameras are installed in every lane on top of the traffic signal. These cameras have to be good, of high resolution and be able to work in multiple conditions including ones which have low light settings. They should be installed in high position and capture bird-eye view so as to capture the most area of the road. Each of the camera will be connected with a Raspberry Pi with Car Counting model installed. The video will be fed into the Raspberry Pi in real time. The output of the Raspberry Pi is collected every 1 second and in Json format. These outputs will then be fed to the server which then run the algorithm to get the result of the lane which should be given green light and the duration of the light. It is also connected to a web application that allows user interaction and the traffic signal. After all calculation and process, the traffic light outputs the color of the light corresponded to the output made by the server. The client-side web-app can also see number of car in each lane and control the traffic signal and change the mode of algorithm.

CCTV Cameras

CCTV Cameras

Raspberry Pi

Raspberry Pi

Server

Traffic Signal

Client-side web-app

Due to the lack of hardware and permission to install cameras on top of traffic lights and intervene the traffic light system, the previous framework is not feasible to implement. Alternatively, building parts separately and simulation is possible to prove that the system will work and make it as ready as it could for real life application. The chosen framework is described in the graph below.

Videos

Car Counting Model

Find best algorithm

Client-side Web app

Simulation

Server

The video is the replacement for the visual input captured by CCTV, which serve the main purpose of testing the car counting model. The car counting model is trained and implemented using various approaches, which then give output in the format of a Json object. These outputs will be same as the input received by the Server (which is represented by a faded dash line). The reason for Server not receiving direct output from the model is that the server needs real-time input from all 4 lanes while the car counting model which then be installed into the raspberry pi contains the input of only 1 lane. Simulation is used to find the best algorithm for the server and stream live simulated data to feed into the Server. The Server analyze the data and stream live data to Client-side web-app. The picture below represents the crossroad with labeled lane from A to D that I tried to simulate.

A picture containing text, military vehicle, control panel

Description automatically generated

A



Figure 1

Building the prototype of the traffic light and getting live data from 4 lanes in 1 crossroad has been considered. However, since we do not have the control over the traffic signal of that crossroad, the nature of the number of cars increases or decreases when the algorithm allots a green light in a particular lane has not been fulfilled. Hence, it would not show the goodness of the algorithm and we cannot see the reasonable changes on the web app when the light shifts.

# **CAR COUNTING MODEL**

The model is built to serve the purpose of counting number of cars and trucks in each lane (lane go straight, lane turn right).

The first model that I attempt to build is from [4] that I have to make adjustment. Instead of counting number of cars in total, I modified the code to count the number of cars in each lane in current time. 4 lines are drawn namely A1, A and B1, B in a colored droi as shown in the picture below.

A high angle view of a road

Description automatically generated with low confidence

The initial count in each lane is 0. Every time cars go past the line A1 and B1, the number of cars in the respective lane increase by 1. While cars go past the line A and B, there will be a reduction of 1 applied to the number of cars in each lane. The algorithm seems right at first when it successfully detects cars and whether the cars have pasted the line or not. The model is stable for the first few minutes but after some time, the count has a lot of error. This error results from the cumulative summation, while at each frame, a wrong car detection and tracking could result in a dramatical error in the car count after a period of time. The model even shut the result up to 10 - 15 cars per lane after only 5 minutes of operating while the actual number of cars in each road is just 3-4. This could affect the algorithm and the system dramatically. That is why this model and the idea of counting car by cumulative summation are not accepted.

Different approach has to take place in order to tackle this issue and provide a more reliable model. In the next model, cumulative summation is not applied. Instead, we analyze the video frame by frame and count the number of cars in each frame. While it seems to be an easier approach, the computational cost is an issue. The model is built using OpenCV. In the model, 2 drois of different colors are drawn for each lane. The model will then take a frame of a video every 1 second and detect the number of car and truck/bus in each droi. The output consisting of the number of cars and truck/bus in each lane will be dumped in Json format to be passed in later stage. The images below are the output of the model. As in the picture and the terminal, the model has a very high accuracy rate, which makes it very reliable and not vulnerable to the counting error costed by computer vision model.

A picture containing outdoor, scene, way, road

Description automatically generated

Graphical user interface, text

Description automatically generated

The model will not analyze frame but only take 1 frame every 60 frame (because it is a 60 frame/ second video) and then set the delay for 1 seconds to make it real-time. This decision was made because there is only a minor fluctuation in the traffic load in 1 second, which means the performance is more or less the same. While this approach will reduce the computational cost by 60 times and make the output more stable and readable. This model is reliable enough to be put for later stage of the project.

# **SIMULATION**

After making a suitable model for real-life application, simulation is then used to find the best model and simulate the real-life traffic to feed in the server, which then be displayed to the client-side web app. In this project, the simulation will simulate the crossroad next to Mc Donald Lidcombe, Sydney, Australia 2141. The crossroad looks like in Figure 1 where the label of each lane is as in the image and with the following characteristic: Road A has 2 lanes (1 straight and 1 right)

* Road B has 2 lanes (1 straight and 1 right)
* Road C has 4 lanes (3 straight and 1 right)
* Road D has 4 lanes (3 straight and 1 right)

*\*\*The load will be assigned according to number of lanes in each road*

In the simulation, we simulate the number of cars which come in each lane and number of cars leaving when the light is green. We do this simulation with the grain of 1 second, which is similar to our car counting model because I want to make the experience and the delay of the input is similar to having a model integrated in the algorithm. We first begin the simulation with get\_truncated\_normal function which is to get a number in a normal distribution with given mean and standard deviation with limits on both ends. The check\_normal\_dist is to check get\_truncated\_normal function and return a histogram to better visualize the distribution. The car\_initialization serve the purpose of initialize the number of cars in each lane. It will return a list of traffic load with length of 2, in which the first element represents the number of car while the second element is the number of trucks. The car\_generator is a function to generate more car with the grain of 1 seconds. It will simulate the number of cars which come in 1 lane in 1 second in real-life. While car\_leave function’s main purpose is to simulate number of car leaving in 1 lane in 1 second when the light is green. Convert\_load and time\_wait function is to convert the load to feed in the algorithm and get the time allotted for green light respectively.

The first algorithm is written in algo1 function, it accepts the input of the traffic load and return the corresponding lanes in which the green light should be given. The algo1 will iterate through a list of rules that is stored in a list of tuples. It will allow 2 lane which is not blocking each other go green at the same time. We named the lane as below

A picture containing text, military vehicle, control panel

Description automatically generated

B

D

C

A

From lane A, the algorithm at 1 time will accept 1 one the following scenario:

* Lane A go straight and right
* Lane A and Lane B both go straight
* Lane A go straight, and Lane D turn right
* Lane A go right, and lane C go straight

Same for other lane, the logic is to maximize the lane go in one time while preventing lane blocked.

The function real-life is to simulate the real-life traffic rules at the crossroad in this example. It sequentially allows traffic to go in the following order:

* Lane A go straight and right for 22 seconds
* Lane B go straight and right for 22 seconds
* Lane C and D go right for 30 seconds
* Lane C and D go straight for 70 seconds

It seems that the rule that this traffic light use is pretty reasonable considered the load traffic in each lane. But as we continue with this project later. We can see that this rule is far from optimize.

Lastly, the evaluation function is evaluating the point of each algorithm. The formular for evaluation is:

Point\_every\_second =number of car \* factor

The choice of alpha is very important, for this simulation, alpha is set to 1.02, meaning with 1 lane road, with 4 cars, the factor will be 1.08 while with 10 cars, the factor will be 1.21. As the number of cars stuck in a signal increase, the factor increase, which makes the evaluation point of the algorithm increase.

Now, let’s start with the simulation. The simulation function accepts 2 parameters, which is the number of iterations and the algorithm used. First, we initialize the traffic load in each lane in a dictionary. Each letter represents the label of the lane. Each contains a list with the length of 2. The first element is the list representing the traffic loads for the lane that go straight. The second element is the traffic loads for the lane that go right. This intuition is kept throughout the simulation. We run a while loop which simulates the nature of the traffic in each iteration, cars will be added to each lane using car\_generator function. The green and green\_time variables store the rule and green light time allotted for each lane. When the light is green, car\_leave function is called to simulate car reduction while car\_generator is also called since when old cars leave the traffic light, new cars come in. The choice of mean and standard deviation is very important to simulate the nature of the crossroad. With that being said, experiments have been done to come up with the mean of X and standard deviation of Y in number of cars come in and X, Y for number of trucks comes in respectively. The mean and standard deviation of car leave is 0.55 and 0.2 respectively when that number in truck leave is 0.45 and 0.2. when the green light time ends, the for loop with 2 times iteration is ran to simulate 2 second delayed in real life when green light shift from one to another. During the while loop, evaluation function is called to keep track of the score in each algorithm. When the light shifts, the number of iterations increase by 1 and it will keep looping when we reach the expected number of iterations in the algorithm. During the while loop, the algo2 is implemented on top of the algo1. Same logic is share between the two. The only difference is that the green time will stop automatically when the lanes which are given the green light have lower traffic load than a certain threshold (0.5 in this case).

To collect the evaluation score, save\_value function is made to run all 3 algorithms namely algo1, algo2 and real\_life with the given number of iterations. These results will be saved in a csv file called “output.csv”.

Table

Description automatically generated

From the table above, we can see the huge difference in the performance of real\_life and the 2 algorithms. The performances differ even more when the number of iteration increase. The algo1 performs slightly better than algo2 in the simulation with the iteration number of 10 but worse in others. Reason for this maybe the initialization of algo2 at the simulation with 10 iterations is worse the initialization of algo1. Nevertheless, we can conclude the algo2 is the best algorithm and will be used for the Server.

# **SERVER**

The Web-app is created to display the simulation result. It is built using Django framework. It receives input from the simulation and stream it onto the front-end. The server contains functions to handle user log in, log out and front-page rendering. All the functions needed for simulation are in base.py. Importing all crucial function and code adjustment is essential for simulation on the server. To handle data streaming, Django Channels framework is used to make the web app more dynamic and allow developers to send data continuously to the front-end. All channel implementations are in consumer.py. As soon as the Web Socket is connected to the server, it will start the simulation and send the data to the front-end every 1 second. The reason for 1 second delay is to make the code run more stable and it also simulates our car counting model’s output.

There are many different approaches. One is to rewrite the simulation code to JavaScript. By doing that, the role of data streaming is neglected. And it would be much easier to make the web app more interactive and dynamic. However, when the car counting model is integrated into the framework, the use of data streaming from back-end (Django) to front-end is very important. That is why data streaming is chosen to make this web-app.

Although, smart traffic signal using computer vision sounds like a very good idea. It has its own drawback, one of which is the camera is installed outdoor and will have to work under different environment settings. Hence, it can reduce the accuracy of the car counting model and reduce the overall performance of the application. To tackle this problem, the back-end uses weather API and check for the weather at that location every 10 minutes. The algorithm will automatically switch back to the real\_life rule whenever the weather is bad (raining) to maintain the performance of the application.

# **CLIENT-SIDE WEB APP**

The client-side of the web app receives data sent by the back-end and provide user interface for user interaction. First, when going on to the route of the web application, the login page appears. Users have to type in their correct credential in order to reach the control panel.

Rectangle

Description automatically generated with low confidence

After providing correct username and password, users will be directed to the control panel where they display simulated information about the traffic load in each lane of the road(number of cars and trucks) and the lane which is currently be given a green light.

Table

Description automatically generated with medium confidence

The main page has the header containing the log out button to log out of the account. The front page has 4 boxes representing for 4 roads in the crossroad with similar label as in figure 1. The dots after the direction of the lane represent the color of the signal at that lane. The number below the direction is the total number of seconds of green light left. When the light is read, the default value for those field is 0.

As mentioning earlier, the application is using weather API to get the weather current time every 10 minutes. If it is raining, the current the algorithm will be switched back to real\_life. The Mode on the top right corner of the UI is to display the current algorithm that the Server is running. By default, the mode will be algo1.

When finish using the application, users can log out of the account and this will take them back to the login page.

Graphical user interface, application

Description automatically generated with medium confidence

Users who do not have an account will fail to have access to the control panel. To create new users, current users have to go to /admin route and sign in with their credential. Because this is a traffic system control panel, not everyone will have access and can create account to use the cite. Only authorized users will be allowed to log in and create new users.

Graphical user interface, text, application, chat or text message

Description automatically generated

Although the web-app has succeeded in streaming data to user, it is still not very interactive. Future development will be made so that users can authorize the algorithm and be able to change the algorithm and even custom the explicit lane to be given a green light (without breaking the traffic rule)

# **CONCLUSION**

The development of computational power in the past 1 decade has made computer vision possible, which paves way to many interesting ideas and practical applications. Smart traffic signal using computer vision is not and exception. It has successfully detected traffic loads, which make it possible for traffic controlling to reduce time waste and resources.

The web application serves as the way for users to interact with the system to make it more intelligent.

This project has given a solution to optimizing the use of traffic light to minimize traffic congestion using computer vision as a tool to measure traffic load.

# **REFERENCE**

[1]. (2013) The Public Transportation System of High Quality in Taiwan by Shi, Y.Q. and Yang, X.G. 2013 IEEE Eleventh International Symposium on Autonomous Decentralized Systems (ISADS), Mexico City, 6-8 March 2013, 1-6. https://doi.org/10.1109/ISADS.2013.6513426.

[2]. (2009) Continuing Evolution of Travel Time Data Information Collection and Processing by Tarnoff, P.J., Bullock, D.M., Young, S.E., et al. Transportation Research Board Annual Meeting, Washington, 11-15 January 2009, Paper #09-2030.

[3]. (1990) Center for Transportation Research, the University of Texas at Austin. Infrared Detectors for Counting, Classifying, and Weighing Vehicles by Gamer, J.E., Lee, C.E. and Huang, L.R, 3-10-88/0-1162.

[4] Ivy car counting model by nicholaskajoh

https://github.com/nicholaskajoh/ivy

1. [↑](#footnote-ref-1)