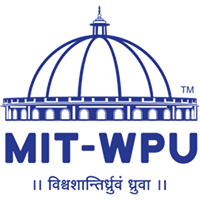
****

**Mini Project Report**

On

**Smart Traffic Management System**

# **Submitted by**

## **Siddhant Shivewar (PF01)**

## **Uday More (PF07)**

## **Suyash Ladda (PF13)**

## **Vishwa Singh (PF15)**

## **Yash Yadav (PF16)**

## **Sudhin Pillai (PF18)**

## **Vivek Ray (PF25)**

**Yash Mangukiya (PF60)**

## **Sohail Shaikh (PF64)**

**Under the guidance of**

**Prof. P.** **SOYGAONKAR and Prof. R. SURYAWANSHI**

**School of Computer Engineering and Technology**

**MIT World Peace University, Kothrud,**

**Pune 411 038, India**

### 

### **SCHOOL OF COMPUTER ENGINEERING AND TECHNOLOGY**

# CERTIFICATE

This is to certify that

## **Siddhant Shivewar (PF01)**

## **Uday More (PF07)**

## **Suyash Ladda (PF13)**

## **Vishwa Singh (PF15)**

## **Yash Yadav (PF16)**

## **Sudhin Pillai (PF18)**

## **Vivek Ray (PF25)**

## **Yash Mangukiya (PF60)**

## **Sohail Shaikh (PF64)**

## of T. Y. B. Tech. successfully completed Mini Project Report in

## **Smart Traffic Management System**

to my satisfaction and submitted the same during academic year 2019-20 Trimester VII as part of Embedded and Internet of Things Laboratory subject.

**Prof. P.** **Soygaonkar and Prof. R. Suryawanshi Dr. M. V. Bedekar**

**(Mini Project Guides) (Program Head)**

**Place:** School of Computer Engineering and Technology, MIT-WPU, Pune

**Date:** \_\_\_\_\_\_\_\_\_\_\_\_\_

**Table of Contents**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Topic** | | **Page No.** |
| **1** | **Introduction** | | **5** |
| **2** | **Related Work** | | **5-6** |
|  | **2.1** | **Literature Survey /Analysis of existing methods** | **5-6** |
| **3** | **Proposed Work** | | **6-12** |
|  | **3.1** | **Problem Statement** | **6** |
|  | **3.2** | **Social Relevance** | **7** |
|  | **3.3** | **Architecture/Model** | **7-10** |
|  | **3.4** | **Hardware and Software Requirement** | **10-11** |
|  | **3.5** | **Results obtained (with Screen shots of Results)** | **11-12** |
| **5** | **Conclusion** | | **13** |
| **6** | **References** | | **14** |

***ABSTRACT***

With this project, we are trying to solve 2 problems.

1} The first one being that the current adaptive signals are too expensive and complicated for implementation. According to the cost database on the current adaptive signal systems cost between 6,000 to 60,000 USD per intersection. And since it uses piezo-electric sensors, the implementation takes a lot of time. To solve this, we have developed a camera based adaptive signal timer.

*SOLUTION:*

We count the number of vehicles waiting at the signal and then depending on the count, we find the new value of the timer and dynamically reduce the timer.

2} Using the same camera and the system, we can also catch the signal jumpers which solves a major traffic rule violation problem.

*SOLUTION:*

We can capture a picture of the car that is crossing the stop line and send them to the cloud based storage then OCR can be used on the number plate and then the owner can be fined.

The model proposed is built on Raspberry pi 3b+ and uses the camera module of Raspberry Pi and the cloud based storage system(FIREBASE).

1. **Introduction**

Although automobile has made urban life a lot easier, it has created a whole new set of problems to solve. To name a few, traffic congestion, rule violation, speeding and accidents are something people see a lot of in their daily commute. In these scenarios, the work of handling this manually is burdensome. However, we can use technology to automate some if not all the task. In our proposition, we are trying to solve the congestion due to static-timer traffic signals and penalize those who break the signal with just *a* Raspberry Pi 3 B+ and a camera. Using OpenCV v4, we can create simple modules to run these tasks.

1. **Related work**

The most conventional method for making adaptive traffic signals is using actuating sensors like piezoelectric plates. Many such methods are already implemented in many places. However, these are on the more expensive side since the construction cost for such methods is a lot.

Computer vision-based methods to manage traffic without human assistance have been proposed earlier. A proposed method [1] in 2005 used Kalman’s evaluation to estimate the number of vehicles and pedestrians to manage the signal using fixed mode. Since then object detection methods have become more accurate and less taxing on the processor. The famous Viola-Jones algorithm [2] for object detection using Haar-like features can now be implemented with ease with OpenCV methods. There are more accurate methods based on deep learning like Single-Shot multi-box detector [3] and MobileNet which is a lighter version of the same for object detection in traffic [4].

These same methodologies are used for catching the signal rule violators. The more challenging part is the handling part for the same. The goal will be to get the license plate number of the signal breaker and penalize them automatically. This task is further broken down into detection and translation. The same object detection algorithms can be used for the detection part and advanced methods like the Tesseract engine can be used for building the OCR. A number plate recognition method is presented in [5].

For our purpose of the model, we have used the simple background subtraction method roughly inspired by the MOG foreground extraction. This allows us to detect any object that strolls into the background thus doing the work of an object detection method.

1. **Proposed Work**

***3.1 Problem Statement***

The goal is to make a raspberry pi-based traffic control system that times itself according to the number of vehicles on the road and captures those who try to violate the signal. The timer value will be based on a simple weighted formula F(count). The pictures of the jumpers will be saved on the firebase cloud and a program on the central system will fetch these images and will try to translate and store the license number of the vehicle.

***3.2 Social Relevance***

The traffic problems in urban cities have increased a lot. Specially in a country like India, where our traffic department falls short on handling this matters. This causes many discrepancies and encourages some people to violate the traffic rules.

This application will help the authorities manage and control the traffic more efficiently which will reduces traffic congestion, rule violation and accidents caused by them.

***3.3 Architecture/Model***

The major library used for making this model is OpenCV V4. This has a lot of functions that comes in handy for complex images processing.

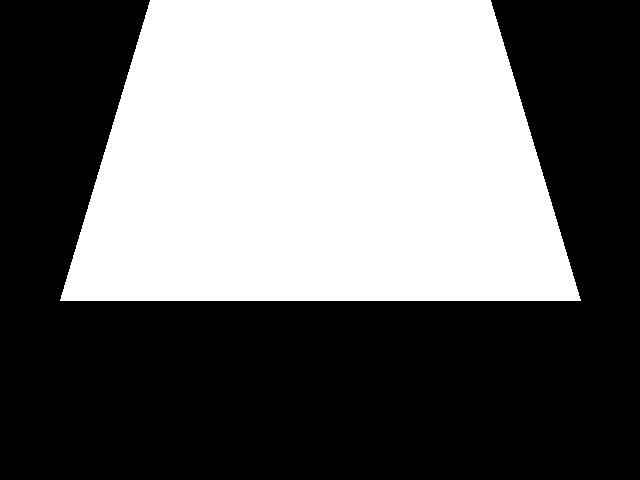
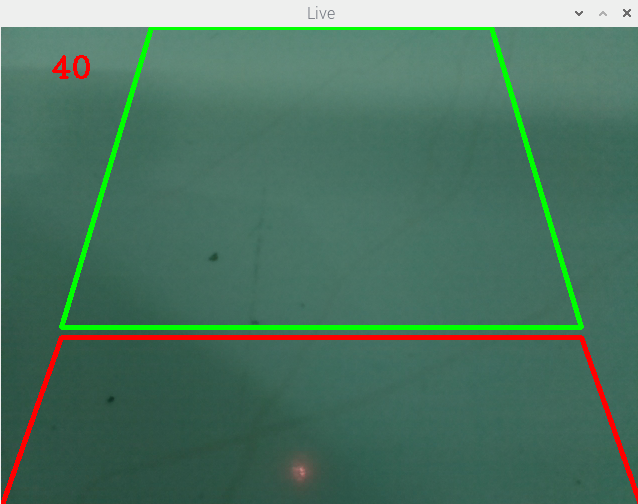
The first step was to create the region of interest for the vehicles and the jumpers. Since we cannot slice a trapezoid segment for the region of interest form a 2D matrix, we resorted to using mask for the frame. We drew the trapezoid on a blank image and replaced the 0 pixels inside with the value 255.

Fig. Mask for region of interest



The mask was then applied to the frame using logical AND operation. This removed the unwanted region form the frame.

For the counting system, we can use simple background subtraction to get a mask of the moving object for the model’s sake. In the real application, using a Haar classifier will be more practical since it will not detect all the moving object in the background.

For detecting with background subtraction, we will have to use contours on the get the boundaries of the vehicles. To avoid irrelevant contours, we will only select those with area greater than 1000. OpenCV has built-in functions to make bounding rectangles around the contours. By counting these rectangles, we can determine the number of vehicles waiting at the signal. The time subtraction will be done after every 5 frames. The current formula to subtract the frames is ***Time – α x Number of vehicles.***

**Turn signal green**

**Yes**

**Time = 0?**

**Initiate**

**wait time**

**Wait for ~5 sec**

**Count the vehicles**

**No**

**Subtract**

***D + N\*α* from the remaining time**

The same mechanism can be used for detection vehicles violating the signal. The bounding rectangles can be extracted after every 10 frames and can be saved to a local directory. Once the signal is not following its red routine, we can use the opportunity to upload the image to the firebase DB for further processing.

**Apply background subtraction**

**Yes**

**Draw bounding rectangle**

**Send image to cloud**

**Bound relevant contour for text**

**Define ROI**

**Contour area > 5000p**

**Apply Thresholding**

**No**

**Save image of the License plate**

The saved image will now be used at a central system. The purpose of this system will be to translate the number plates of the captured car and save them to the database. The database can be used then as a reference as to which vehicles has to be fined for signal violation. To extract the number plate, we can used thresholding-based methods on OpenCv or we can used an SSD model made on tensorflow.

For the translation, we will be using the PyTesseract module working on the tesseract engine since it is one of the best open-source OCRs currently available.



***3.4 Hardware and Software requirements***

**Hardware Requirements:**

* RaspberryPi 3B +
* PiCamera
* Nvidia Powered GPU for the main system

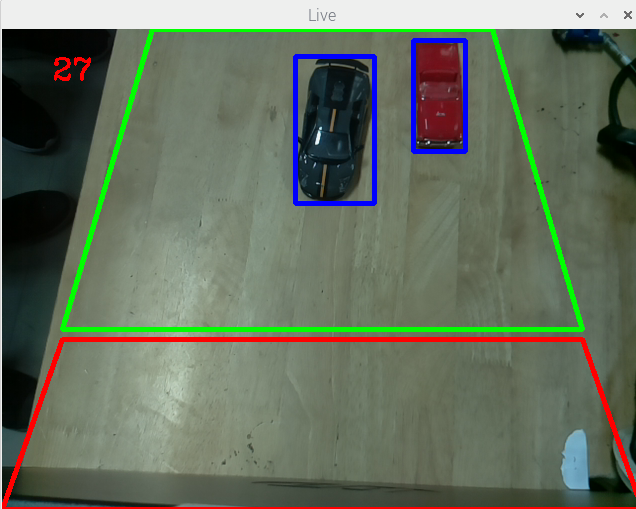
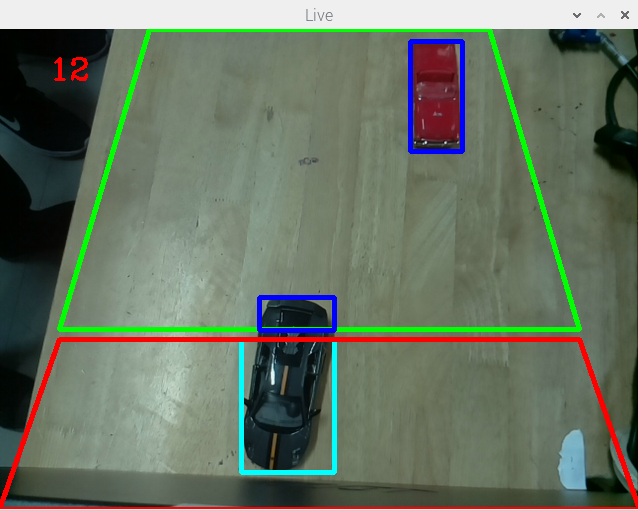
**Software Requirements:**

**For Rpi 3B+:**

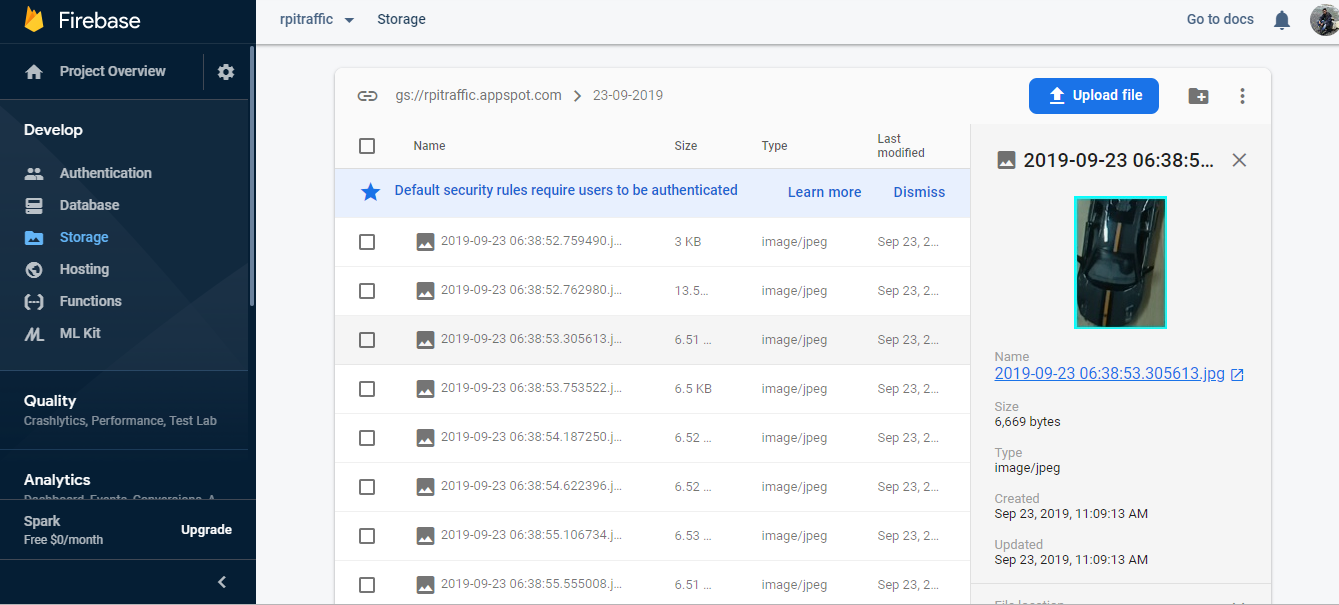
* Raspbian OS
* Python 3.6
* OpenCV v4
* Numpy

**For central system:**

* Python 3.6
* Tesseract engine and PyTesseract
* TensorFlow

***3.5 Screen prints***

Screen. Vehicles being detected



Screen. Pictures being saved on the cloud



Picture of the model

**IV. Conclusion**

This project gives us a cheap alternative for adaptive signals and provides us with a practical method for catching and penalizing traffic rule violators. More functionalities can be added to this model like handling emergency vehicles, detecting accidents etc. As technology becomes cheaper and faster, this model will become more feasible to implement and will make traffic management more convenient.

***References***

[1] Computer Vision Application: Real Time Smart Traffic Light Ángel Serrano, Cristina Conde, Licesio J. Rodríguez-Aragón, Raquel Montes, and Enrique Cabello; Feb, 2005.

[2] Rapid Object Detection using a Boosted Cascade of Simple Features Paul Viola Michael Jones , 2001.

[3] SSD: Single Shot MultiBox Detector Wei Liu , Dragomir Anguelov , Dumitru Erhan , Christian Szegedy , Scott Reed , Cheng-Yang Fu , Alexander C. Berg.

[4] Multi-Object Detection in Traffic Scenes Based on Improved SSD Xinqing Wang, Xia Hua , Feng Xiao, Yuyang Li, Xiaodong Hu and Pengyu Sun.

[5] Real-time Automatic License Plate Recognition Through Deep Multi-Task Networks Gabriel R. Gonc¸alve , Matheus A. Diniz, Rayson Laroca† , David Menotti† , William Robson Schwartz, 2018.