JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

Smart Traffic Light and Density Control System

[1]Zubair Hussain Wani, [2]Aafaq Altaf Wani, [3]Amin Fayaz
[1,3]Department of EEE, NMIT, Bangalore, Karnataka, India
[2]Department of EEE, BMSIT&M, Bangalore, Karnataka, India

Abstract: One of the major problems in our country is mismanagement of traffic. Traffic congestions is a result of failure of signals and poor law enforcement. The existing infrastructure cannot be expanded more to deal with this problem, so better management of traffic is the only option available. It has a negative impact on our economy, environment and overall quality of life. Hence traffic management has emerged as one of the big challenges for Engineers, planners, policy makers in urban areas. These days with the significant increase in India's population which consequently results in traffic congestion in most metropolitan cities, developing a smart traffic management system is essential to detect the path with the shortest travel time is critical in the case of emergencies like health services and fire management etc. The aim of our project is to develop a smart traffic management system which can effectively control the density of the traffic by scheduling traffic light signals through Automation and To trace the number plates of vehicles violating the law. Along with these our system can also co-ordinate with emergency service vehicles like Ambulance, Fire brigade, police patrol vans etc. To ensure that these vehicles reach their destinations on time and hence save lives.

***Index Terms* – Traffic Density Control, Number Plate Detection, Emergency Vehicles**

I. Introduction

The Indian city management system may be a mixture of the many interdependent structures that play a vital part in traffic management. It may be collectively of the smart city's key aspects. The planet is moving in no time and for continuous development it must still move this fashion. Modern transport, on the contrary, does not provide individuals with a smooth transportation scheme. Excessive traffic jams cause the strained and frustrated motorists to delay reaching the workplace or home, waste fuel, wear and tear on cars or perhaps a force. Furthermore, a growing population leads straight to rising issues linked to traffic like over-speed, accidents, hit and run, and so on. During long traffic jams unnecessary criminal activities such as mobile snatching of public etc. Smart traffic management has therefore developed as a compulsory necessity for a prosperous civilization. In most developing countries like India a fixed time schemes are preferred. This sort of traffic controlling is being looked largely through centrally controlled systems. Considering this, the net of Things, which has proved its worth in almost everything in standard of living, is considered as a tool for managing the traffic through central server. The number of cars passing through a passage some distances before the active traffic congestions points can be transmitted to the traffic flow control station in our suggested job. The timely data obtained for the city's traffic jam node can be conveyed via internet and cloud to handle car entry. For counting of vehicles in real time, we will be using image processing applications in OpenCV. Our proposal is being implemented cheaply and requires very little infrastructure. Therefore, the proposed system manages the traffic on local and centralized servers by exploiting the concepts of IoT and Intelligent Image Processing. First, the real-time video data is acquired in the proposed system for Indian urban settings. It is then split into frames and then blob detection is done after binary transformation and noise removal and lastly the count is estimated using the suggested technique of car counting. Traffic data representation in statistical form can also be useful for emergency services like Ambulance and fire brigade etc.

1. OPEN CV

OpenCV is an opensource computer vision library. This library is written in C and C++ and runs under Linux, Windows and Mac OS X. There is active development on interfaces for Python, Ruby, Mat lab, and other languages. OpenCV was designed for computational efficiency and with a strong focus on real time applications. OpenCV is written in optimized C and can take advantage of multicore processors. OpenCV automatically uses the appropriate IPP library at runtime if that library is installed. One of OpenCV's goals is to provide a simple-to-use computer vision infrastructure that helps people build fairly sophisticated vision applications quickly. The OpenCV library contains over 500 functions that span many areas in vision, including factory product inspection, medical imaging, security, user interface, camera calibration, stereo vision, and robotics. Because computer vision and machine learning often go hand-in-hand, OpenCV also contains a full, general-purpose Machine Learning Library (MLL). This sub library is focused on statistical pattern recognition and clustering. The MLL is highly useful for the vision tasks that are at the core of OpenCV's mission, but it is general enough to be used for any machine learning problem.

2. IDLE

IDLE is a learning environment of Python Integrated Development and python code is easy to program by a user. IDLE can also be used to perform a single statement, modify it and execute python scripts just like how it works in python shell

3. IDE

IDE stands for Integrated Development Environment. A code editor, build automation, Debugger and several tools are a package of software development. So, IDE software have all this tools.

II. IDENTIFICATION OF PROBLEM

Conventional traffic management includes the use of timers which is not efficient enough for managing traffic and traffic authority having a hard time to control the traffic. Irrespective of density of traffic in the lanes, the lanes varying with high or low densities will have to wait for the same time period so they have to wait for longer time even when traffic density in their respective lane is relatively low until traffic police intervenes. Over the last four years Bangalore, Mumbai and Delhi are consistently being featured on world's top 10 most congested cities for traffic. In fact, Mumbai was at No1 in the year 2018 and has remained in the top 5 ever since. Mumbai, Delhi, Bangalore and Pune have congestion levels of 53%,47%,51% and 42% respectively. Sitting through traffic is one of the least pleasant things people do and some cities as mentioned earlier have it so much worse than others. Most of the traffic signals are working on old school tech generally fixed green light sequence and this sequence is determined without taking into concern the presence of emergency vehicles such as Ambulances, Police cars, Fire engines etc, which gets stuck in traffic jam ang get delayed which leads to loss of life and property so we need a more appropriate method to schedule emergency vehicles in traffic. The increased density of traffic if not kept in check can not only increase the response time of emergency services but also increases the chances of them meeting Accidents. Since the emergency vehicles crossing an intersection at high speeds on red light can be dangerous and can cause tragic accidents. Traffic authority have been facing the issues of vehicles violating traffic rules specifically jumping red signal, since decades!

2.1 METHODOLOGY

The automatic counting of vehicles will be obtained for a chosen destination by using Image processing techniques. To acquire real time traffic flow video footage through road we are using a camera system. After acquiring the video footage, a portable microprocessor system was used to work on this data that is received from camera system. Since, the average number of vehicles per time is specified to communicate the same information to the central control system the same processing system was used.

2.1.1 Counting of vehicles by using image processing: After having the video footage from the camera system, the microprocessor does the processing using Open CV software. To acquire the number of vehicles passing through the respective lane, the amount of work done on image is shown below in the following steps: -

• The prior step involves the extraction of the individual frames from the obtained video footage

• By using the Gaussian filter, foreground image was passed to lessen the noise and to sequentially extract the objects present in the image, a morphological operation of closing was applied.

• In the following step, the processed image was converted into binary form depending on the threshold value pixel value was converted in the form of 1 or 0. For generating this threshold value we are using, OTSU bimodal threshold method. Because it is easier to detect the contours of objects in such kind of image

2.1.2 Information Transmission for traffic management system using internet: By using internet, the process of communication between final user server system and camera guided processing system can be elaborated in below four steps as followed

• Using Open CV, fetching per second vehicle count data can be obtained and after fetching the respective time interval for the present average value of vehicles, which is stored inside the variable that is updated after every fixed time interval.

• Using python interfacing of IoT cloud with microprocessor, portable microprocessor system and a server is used to send processed information remotely to control room and it should be enabled with a WIFI connection. We are using IoT cloud for the server and for time database, we must have a bonding with the processor so we are using Python language to do so.

• We had created a new database on IoT cloud so that we can establish the connection with it after interfacing. Then, credentials like API (Application Programming Interface) keys, End point URL (Universal Resource Locator) are used to connect this to database using Python.

• We are sending vehicle count data to Real Time Cloud; the vehicle count data was sent to server remotely after connecting to database. From Open CV, vehicle count data is sent to IoT cloud using internet through Python with the timestamp. In the above steps, once the data had been sent, the process repeats itself after one second to continuously provide monitoring details of the capturing system.

2.1.3 Vehicle detection and tracking: The vision system's input data consists of image sequences taken from a camera. These images show the environment in front of the car- the road, other cars, bridges, and trees next to the road. The primary task of the system is to distinguish the cars from other stationary and moving objects present in the images and recognize them as cars. Because of the continuously landscape along the road and various lighting conditions that depends on the time of day and weather are not known in advance thus making it a challenging task. Recognition of vehicles that suddenly enter the scene is difficult. With varying speeds, sizes and appearances cars and trucks come into the scene. Using multiple consecutive image frames, analysis of motion information is carried out to describe how passing vehicles are recognized. Then with the help of adaptive feature-based method, we describe how vehicles in the far distance shows very relative motion between themselves and the camera assisted car, can be recognized. Immediate recognition from one or two images is sophisticated and only works robustly under cooperative conditions examples: enough brightness, contrast between vehicles and background. Therefore, if immediate recognition of object is not possible then, our system evaluates several image frames and employs its tracking capabilities to recognize vehicles.

III. PROJECT FRAMEWORK

3.1 BLOCK DIAGRAM

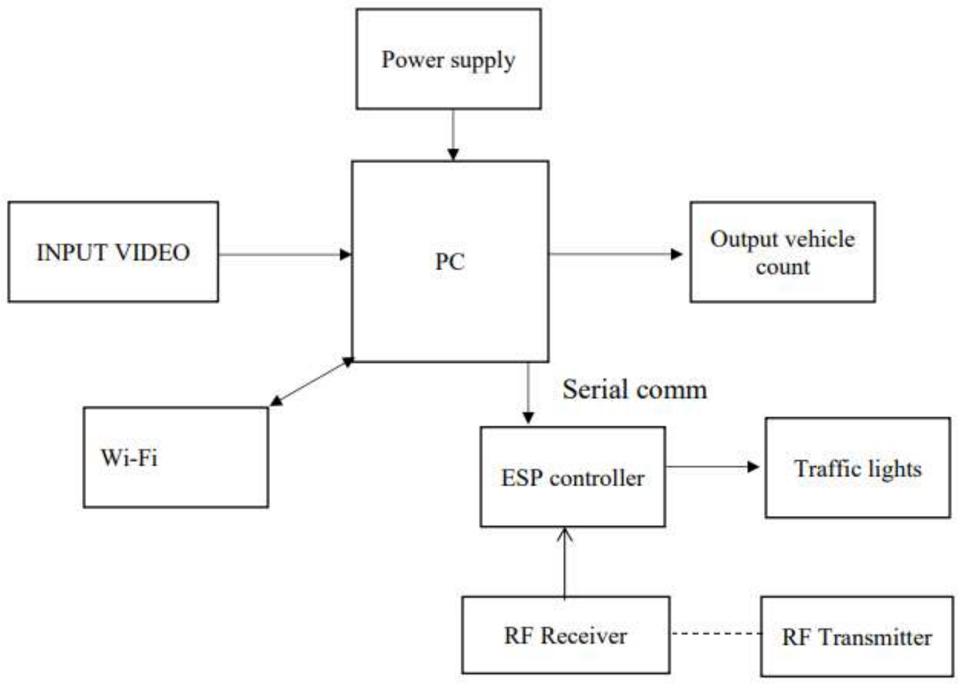


Fig 3.1 : Block Diagram

3.2 WORKING

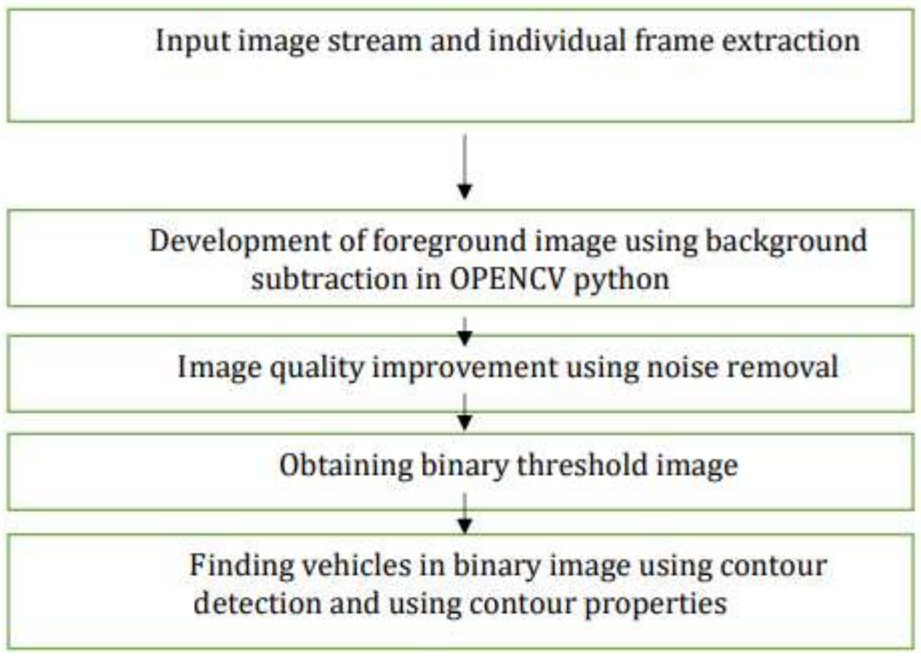
• Initially, the code required for Operation of Traffic lights & Emergency service management is dumped into the ESP 32 Microcontroller. Then, the reference videos are given as input in IDLE 3.9 Software for extracting information about the density of traffic in different lanes & for Tracing number plates of traffic signal violators.

• For controlling the density of traffic in different lanes image processing concept is used for object detection. So, the lane having higher Traffic density than the other lanes will go green first for a fixed time period after that the signal goes Red, to avoid congestion of traffic in that lane. This happens in such a manner that, the Image Processing program sends this data about the density of the traffic in a peculiar lane with respect to other lanes to the ESP32 Microcontroller which process this data, generates and transmits a signal which in turns changes the traffic light in that lane. After that, the next lane with highest density of vehicles at that moment will go green. This whole process continues in a loop. Here this image processing is possible by the python program compiled through IDLE software using Open-Source Computer Vision library. This whole process takes place in real time thus it is more efficient than the conventional method of traffic management, which employs the use of Timers for changing Traffic lights in all the lanes in a synchronized manner.

• For the Traffic signal violation, Violators who jumped the red signal their vehicles can be traced back to the registered owners, who gets fined for this violation. Initially, the real time footage is fed into the python program. The input data is in video format but processing happens image by image that is 25image/sec. For Background Subtraction, Imutils are used for converting images into gray scale for edge detection. Thus, noise is reduced and improving image quality. So, the vehicles which crosses the Traffic junction during red light can be traced by their number plates and thus Traffic management authority is given this information and they debit the challan amount from the violator’s account.

• The Image processing taking place for density control and Traffic signal violation is explained below

Image processing:


Input image stream and individual frame extraction
Development of foreground image using background subtraction in OPENCV python
Image quality improvement using noise removal
Obtaining binary threshold image
Finding vehicles in binary image using contour detection and using contour properties

• For assisting of Emergency services like Ambulance, Fire Brigade, Police vehicles etc., We are using Radio Frequency (RF) Module. It consists of a RF Transmitter & RF Receiver. The RF receiver is installed at the traffic signal and connected to the ESP 32 Microcontroller, the RF Transmitter can either be installed in the emergency vehicle or can also be operated by Traffic police nearby. Firstly, the code required for scheduling of traffic light in Normal condition as well as in the case where emergency service vehicles are involved is Dumped into the Microntroller board which is compiled and executed through Arduino IDE Software. Secondly, When the RF Transmitter module is operated, it transmits a radio wave and modulates it to carry data and this modulated RF signal is received and demodulated by the RF Receiver module. This data is sent to the microcontroller board which in turns schedules the Traffic light signal depending on the lane in which the emergency vehicle is present then, that specific lane will turn Green.

3.2 Image Processing

3.2.1 Image Fundamentals:

• Pixel is the basic block of an image, since every image is made up of a set of pixels. A Pixel is taken in account the color or light intensity which appears at a specific place in the reference image. By assuming the image as a grid, each square represents a single pixel. For example, consider the fig 3.2.1(a)

• Most pixels are represented in two ways:

1) Grayscale channel
2) Colo

Fig 3.2.1(a) This image is 1000 pixels wide and 750 pixels tall, for a total of 1000x750=750;000 total pixels.

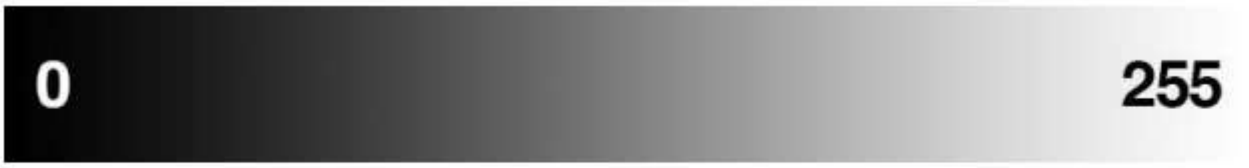
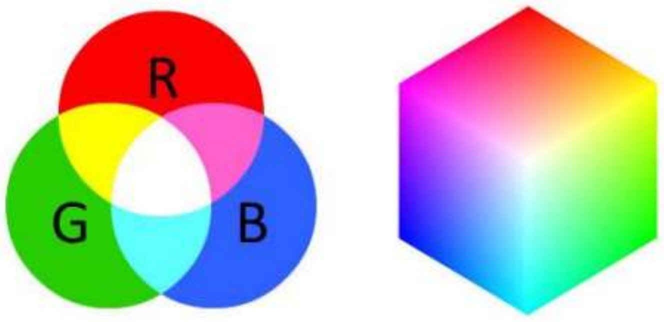


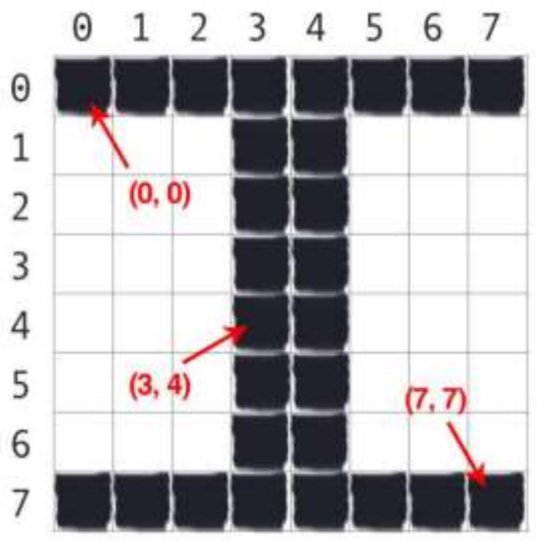
Fig 3.2.1(b) Image gradient demonstrating pixel values going from black (0) to white (255)

• Each pixel is a scalar value between 0 and 255 in a grayscale image, where zero corresponds to “black” and 255 being “white”. Values between 0 and 255 are different shades of gray, here the values closer to 0 are darker and values closer to 255 are lighter.
• The grayscale gradient image in Figure 3.2.1(b) demonstrates darker pixels on the left-hand side and progressively lighter pixels on the right-hand side. Color pixels; however, are normally represented within the RGB color space. The Pixels with in the RGB color space are not a scalar value like that in a grayscale channel image instead, the pixels are represented by a list of three values: one value for the red component, one for Green, and another for Blue. The color in the RGB color model is defined as the amount of Red, Green, and Blue contained in a single pixel.
• The values of each red, green and blue channel are defined within the range [0;255] for a total of 256 “shades”, where 0 indicates no representation and 255 demonstrates full representation. Provided that the pixel value only must be within the range [0;255], we normally use 8-bit unsigned integers to represent the intensity. As we’ll see once we build our first neural network, we’ll often preprocess our image by performing mean subtraction or scaling, that require us to convert the image to a floating-point data type. Keeping this point in mind as the data types used by libraries loading images from disk (such as OpenCV) will often need to be converted the images directly before we apply learning algorithms.
• We can combine our three red, green, and blue values into an RGB tuple in the form (red, green, blue), which represents a given color in the RGB color space. The RGB color space represents an additive color space, where the more of each color is added, the pixel becomes more brighter and closer to white. We are able to visualize the RGB color space in Figure 4.3 (left). As you will see, adding red and green ends up in yellow. Adding red and blue leads to pink. And adding all three red, green, and blue together, we create white.



3.2.1(c) Left: The RGB color space is additive. The more red, green and blue you mix together, the closer you get to white. Right: The RGB cube

Since the image coordinate system is explained briefly in the above paragraph, as the image is denoted by a pixel grid so let’s assume the grid to be a small part of graph paper. The upper left corner of the image has the origin points of [0;0]. Coordinates of X and Y increases as we go down. As we can see a picture of I that is 8x8 grid that has a 64 pixel, we need to remember our counting starts from 1 rather than 0. Since the python language is concerned about the zero factor so we take four columns towards the right of the pixel and five rows down the point ranges from [3;4] .



3.2.1(d)

3.2.2 Images as NumPy Arrays:

Fig 3.2.2: Loading an image named example.png from disk and displaying it to our screen with OpenCV.

• Image processing libraries such as OpenCV and scikit-image represent RGB images as multidimensional NumPy arrays with shape (height, width, depth). Because of the matrix notation height comes before the width.
• While explaining dimensions of the matrix, we usually denote it as rows x columns, where height is represented by number of rows in an image and width is represented by number of columns in an image. The depth will still remain the depth and RGB channels should be in the reverse order to store in OpenCV is mandatory.

3.2.3 Background Subtraction:

Background subtraction is the ability to learn and identify the foreground mask. In simple terms, it is able to eliminate the background portion in an image. It has several uses in our day-to-day life like for object segmentation, tracking pedestrian enhancing security counting the number of visitors, density of vehicles in traffic etc. here the output is an image which is binary segmented and essentially gives information about the non-stationary object in the image. The major drawback from this concept of figuring out the non-stationary portion is the shadow of the vehicle will be moving and could be classified in foreground

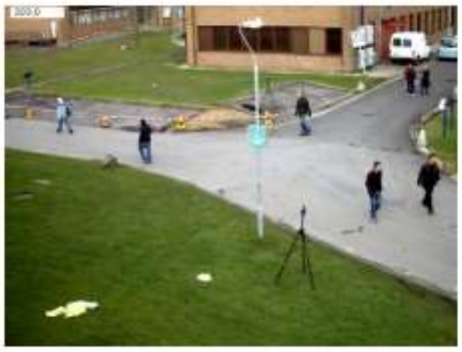


Fig 3.2.3(a) Original video frame:

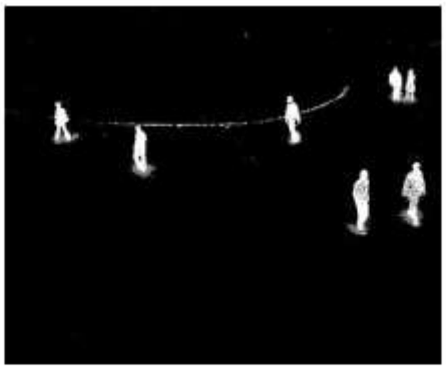


Fig 3.2.3(b)Background subtracted video frame

3.2.4 Gaussian Filtering:

In this method, a Gaussian kernel is used instead of a box filter consisting of equal filter coefficients. This is done with function, cv2.GaussianBlur(). The width and height of the kernel should be specified which generally is positive and odd. The standard deviations in the X and Y directions that is sigma X and sigma Y respectively, should also be specified. Sigma Y is taken as equal to sigma X, in the case where only sigma X is only specified. In the case where both are given as zeroes, they are calculated from the kernel size. This approach is highly effective in removing Gaussian noise from the image. For creating a Gaussian kernel, we can use the function, cv2.getGaussiankernel() and for Gaussian blurring, the code can be written as blur=cv2.GaussianBlurring(image, (5,5),0)

Result

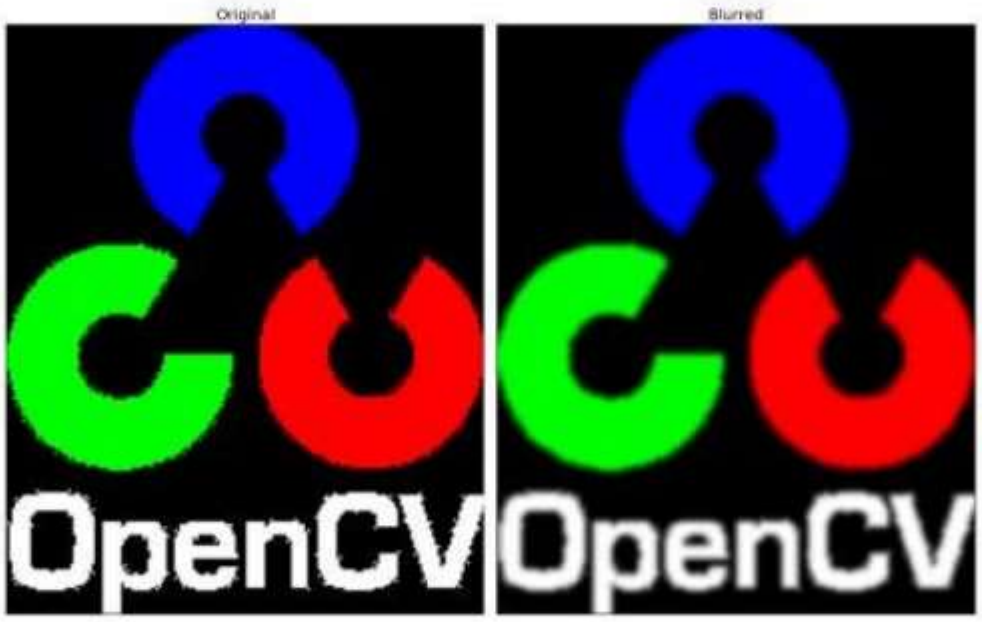


Fig3.2.4 Gaussian Filtering

3.2.5 Thresholding:

If pixel value is greater than the threshold value then, it is assigned one value preferably white, otherwise it is assigned another value preferably black. cv2.threshold is the function used here. Source image is taken as the first argument, which is a greyscale image. Second argument is the threshold value which is used for classifying the pixel values. The final argument that is third argument is the max Val, which represents the value to be given is more than the threshold value

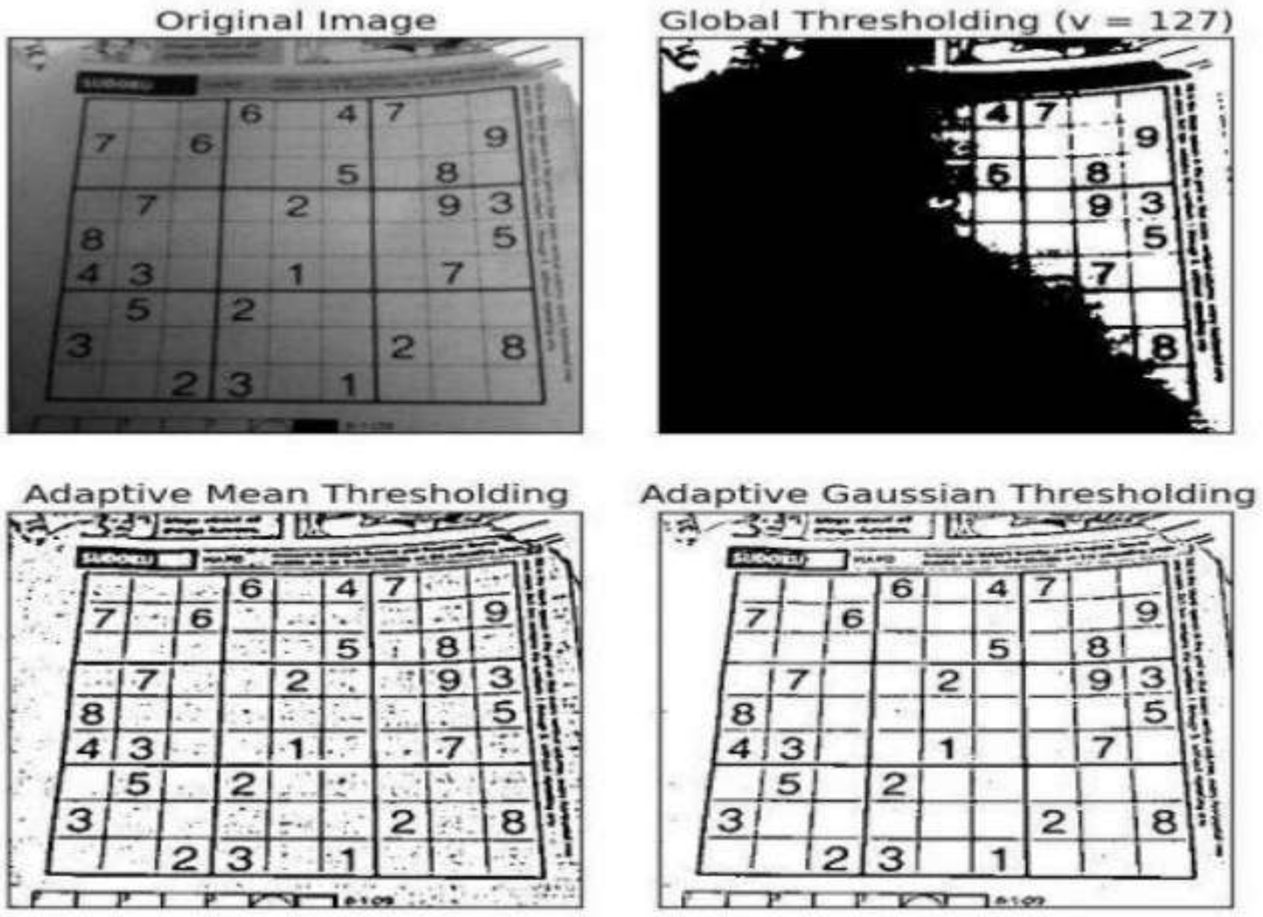

Original Image
Global Thresholding (v = 127)
Adaptive Mean Thresholding
Adaptive Gaussian Thresholding

Fig 3.2.5 Thresholding

3.2.6 Contours:

In simple words, it is a curve joining all the continuous points along the boundary line, which is off same color or intensity. They are very useful for object detection, recognition and shape analysis. For better accuracy use binary images. Threshold or canny edge detection should be applied before finding counter.

• Source image is modified by the contours function. So, if you want source image even after finding counters, already store it to some other variables.

• As we know that In OpenCV, finding counters is like identifying white object from black background. Thus, object to be found should be white and background should generally be black. Below image of a rectangle demonstrate this technique. Just draw a circle on all the coordinates in the contour array (drawn in blue color). First image shows points I got with cv2.CHAIN_APPROX_NONE (734 points) and second image shows the one with cv2.CHAIN_APPROX_SIMPLE (only 4 points). Thus, saving memory.

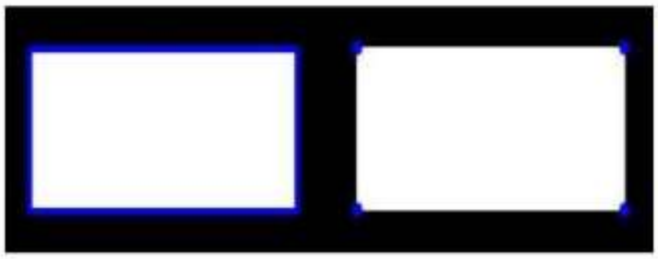


Fig 3.2.6 Contours

IV. RESULTS, CONCLUSION AND FUTURE SCOPE

When we compile the code for Traffic density control in IDLE, we can see the input video frames as seen in the above figure 5.3. From this result frame we can say that density of Traffic in Lane 2 is greater than it's threshold value as a result of which Traffic signal at Lane 2 will turn Green for a fixed time interval to accommodate high density of vehicles in that particular Lane. After a certain time, the next Lane with higher density of Traffic than its threshold value will go Green for a fixed time interval to accommodate this increased density of Traffic. This process continues in a loop detecting and analyzing Traffic density in each Lane and the Lane with higher density of Traffic than its threshold value will go Green next at that certain instant of time

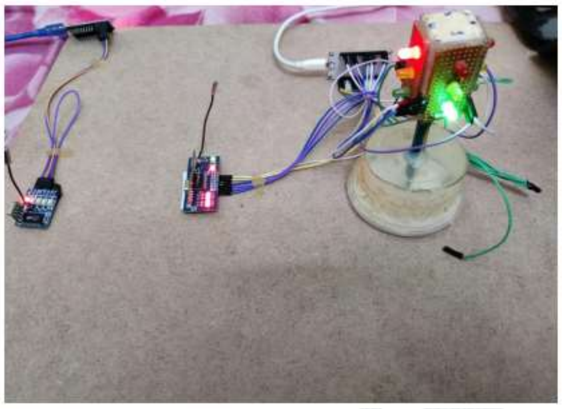


Fig 4.1(a) Demonstrating scheduling of Traffic signal using ESP32 microcontroller in normal case.

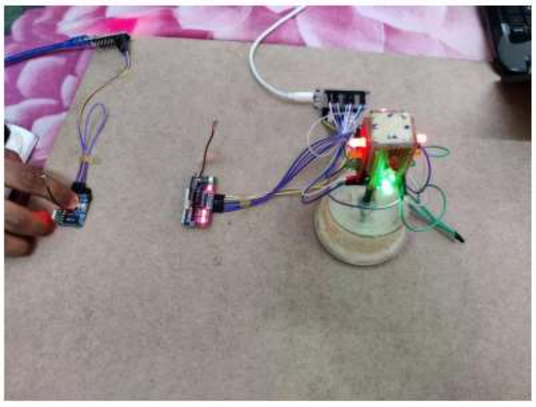


Fig 4.1(b): It demonstrates assisting of Emergency vehicles by scheduling of Traffic signals Using RF Module.

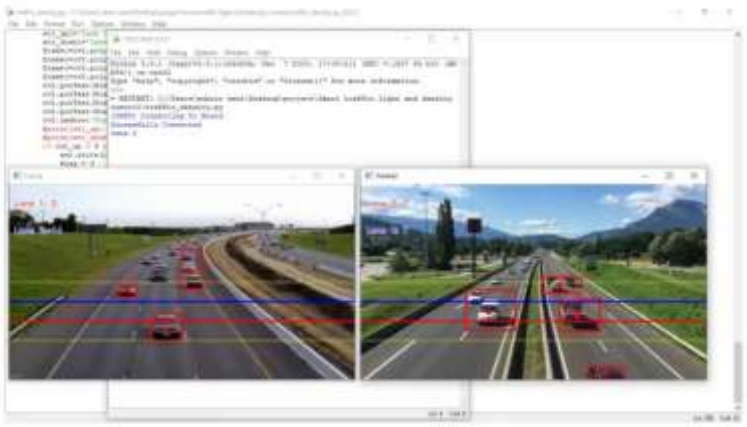
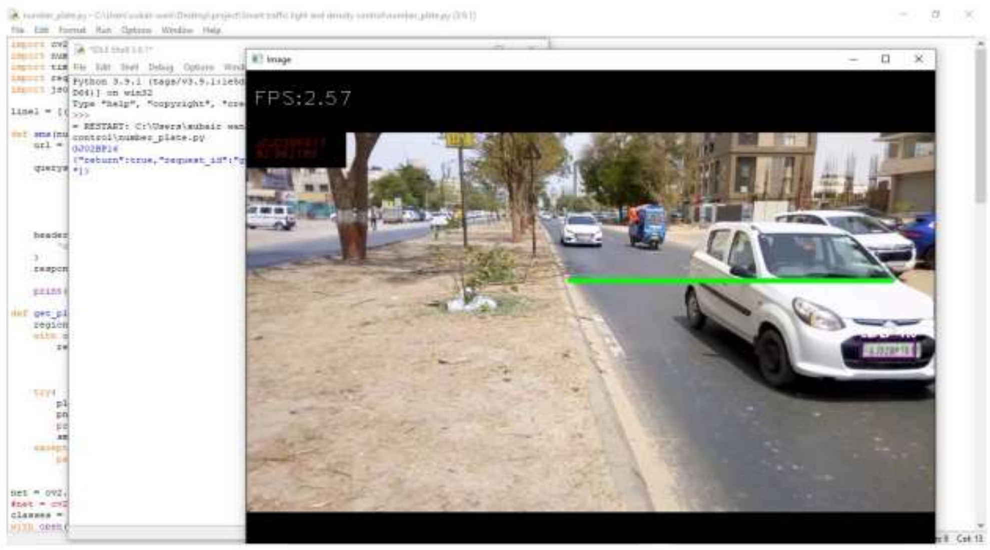


Fig 4.1(c): It shows Traffic density control by image processing.



```
Python 3.9.1 (tags/v3.9.1:1e5d
D64)] on win32
Type "help", "copyright", "cre
>>>
= RESTART: C:\Users\zubair wan
control\number_plate.py
GJ02BP14
{"return":true,"request_id":"
"]}
```

Fig 4.1(d): Represents of vehicle number plate tracing using image processing.

When we compile the code for Number plate detection in IDLE, we can see the input video on screen. When any vehicle jumps Red signal, we can detect their number plates and trace it back to the registered owner of the vehicle and by collaborating with Traffic management authority, we can sent an e-Challan to the offender's phone number.

4.2 CONCLUSION: Mainly our project objective is to change the time of the signal depending on the vehicles and Based on the number of vehicles, image processing can take care of the time alteration. If there is a jammed traffic in a particular lane it gives more time for that particular lane to dispose. Microcontroller plays a vital role in adjusting the time gap difference between the intervals of time which is usually known as dynamic time adjustment systems. Secondly, by using image processing we can keep a control on traffic rule violators. In case of emergencies ambulances and fire brigades can pass through hassle free by using a simple transmitter and receiver RF module which are installed near traffic signal.

4.3 FUTURE SCOPE: We can develop a Smart Traffic Management System which can not only detect vehicles that jumps Red signal but also detects other Traffic violation like Honking in silent zone, Using vehicle without a permit or in unsafe condition, Violation of emission and noise pollution standards, Driving without insurance, not wearing helmet, not wearing seatbelts, Blocking Emergency vehicles etc.

V REFERENCES:

➢ L. P. J. Rani, M. K. Kumar, K. S. Naresh and S. Vignesh, "Dynamic traffic management system using infrared (IR) and Internet of Things (IoT)," Third International Conference on Science Technology Engineering Management (ICONSTEM). https://issuu.com/irjet/docs/irjet-v7i8223

➢ T. e. a. Osman, "Intelligent traffic management system for a cross section of roads using computer vision," in Computing and Communication Workshop and Conference (CCWC), 2017 IEEE 7th Annual, 2017 https://ieeexplore.ieee.org/document/7868350

➢ V. S. A. M. S. D. K. K. Swathi, "Traffic Density Control and Accident Indicator Using WSN," International Journal for Modern Trends in Science and Technology. https://www.ijmtst.com/documents/15.IJMTST020409.pdf

➢ Mahesh Lakshmi Narasimhan, "IoT Based Traffic Management System", March 2016. https://www.researchgate.net/publication/310036684_IoT_Based_Traffic_Management_ System

➢ S. Rane, A. Dubey and T. Parida, "Design of IoT Based Intelligent Parking System Using Image Processing Algorithms," IEEE International Conference on Computing Methodologies and Communication, pp. 1049–1053, 2017. https://ieeexplore.ieee.org/document/8282631

➢ H. O. Al-Sakran, "Intelligent traffic information system based on the integration of Internet of Things and Agent technology," International Journal of Advanced Computer Science and Applications (IJACSA), vol. 6, no. 2, pp. 37–43, 2015. https://www.researchgate.net/publication/276382050_Intelligent_Traffic_Information_System_Based_on_Integration_of_Internet_of_Things_and_Agent_Technology

➢ M. R. Rahman and S. Akhter, "Real time bi-directional traffic management support system with gps and websocket," in 2015 IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing, Oct 2015, pp. 959–964. http://ieeexplore.ieee.org/document/7363185/

➢ S. Nawrin, M. R. Rahman, and S. Akhter, "Exploring k-means with internal validity indexes for data clustering in traffic management system," in International Journal of Advanced Computer Science and Applications, 2017. https://www.researchgate.net/publication/315747115_Exploreing_KMeans_with_Internal_Validity_Indexes_for_Data_Clustering_in_Traffic_Management_S system

➢ D. Sherly, "Internet of things based smart transportation systems," in International Research Journal of Engineering and Technology, 2015. https://ieeexplore.ieee.org/document/9221208

➢ L. Lu, "Wise-paas introduction," Internet, 2017. http://scholar.google.co.in/scholar?q=%EF%83%98%09L.+Lu,+%E2%80%9CWisepaas+introduction&hl=en&as_sdt=0&as_vis=1&oi=scholart