CENTER FOR DEVELOPMENT OF ADVANCED COMPUTING

Dr. D. Y. Patil Pratishthan's

Institute for Advanced Computing and Software Development

Documentation On

"Traffic Violation Detection (Red-light) System Using OpenCV"

- PG-eDBDA -

September - 2020

Submitted By:-

Group No: 18

Omkar Kumbhar-1526

Sumeet Gadewar-1547

Mr. Prashant Karhale
Center Coordinator

Mr. Akshay Tilekar.
Project Guide

ACKNOWLEDGEMENT

We have taken efforts in this project. However, it would not have been possible without the kind support and help of many individuals and organizations. We would like to extend our sincere thanks to all of them. We are highly indebted to Mr. Prashant Karhale, Centre Co-Ordinator IACSD CDAC, Akurdi Pune for constant supervision as well as for providing the opportunity.

We would like to express our gratitude towards our faculties, project Guide Mr. Akshay Tilekar, Internal Project Guides Mr. Rahul , Mr. Manish and Mrs. Tejal, for their kind co-operation and encouragement which help us in completion of this project.

Our thanks and appreciations also go to the colleagues in developing the project and people who have willingly helped us out with their abilities.

Index

List of Figure

Abstract

An automatic traffic red-light violation detection system was implemented, which may play a big role in transportation management in smart cities. The system mainly relies on modern computer vision techniques, which was implemented in OpenCV under Python environment. Mainly, the system consists of object detector and object tracker which work in an integrated manner in order to precisely keep position of the existing cars. The primary task of the system is to eventually indicate locations of violating vehicles. The output showed accurate results, as all violating vehicles were detected and distinguished precisely.

Introduction

Good driving is about being prepared for every eventuality. From following traffic rules, every small thing adds up to your safety. Take this quiz to know if you're a cautious and safe enough driver. The increasing number of vehicle(cars) in cities can cause high volume of traffic, and implies that traffic violations become more critical nowadays in India This causes more accidents that may endanger the lives of the people. To solve the alarming problem and prevent such unfathomable consequences, traffic violation detection systems are needed. For which the system enforces proper traffic regulations at all times, and recognize those who does not follow. A traffic violation detection system must be realized in real-time as the authorities track the roads all the time. Hence, traffic enforcers will not only be at ease in implementing safe roads accurately, but also efficiently; as the traffic detection system detects violations faster than humans. This system can detect traffic light violation in real-time. monitor traffic and take action against the violations of traffic rules.

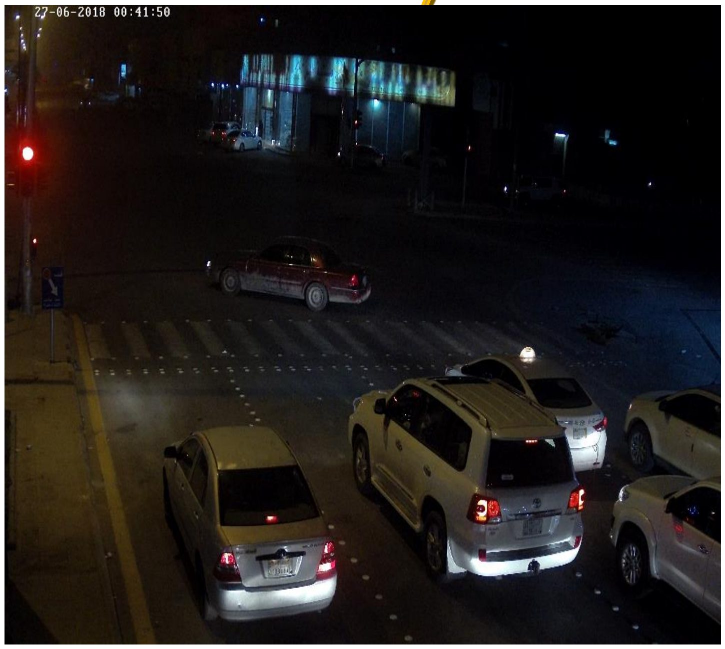


Figure 1 Original Image

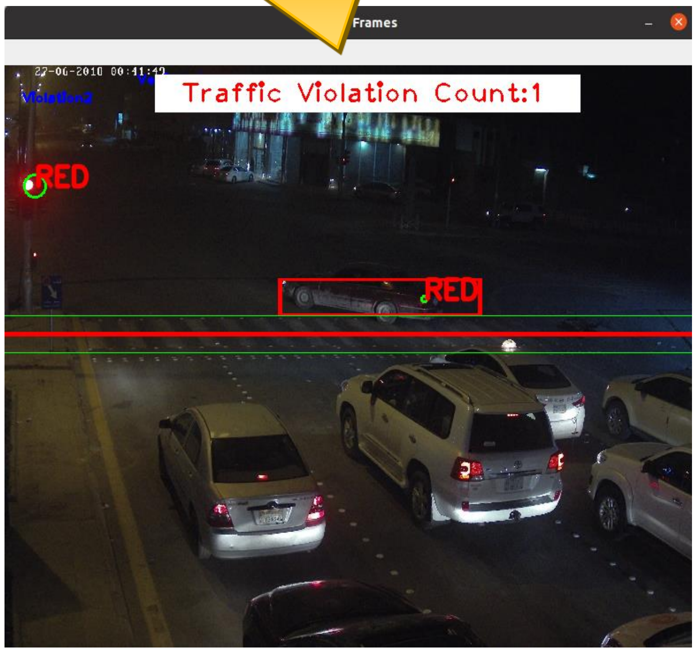


Figure 2 Detected Image

Keeping an accurate localization on each vehicle can be very tedious task, tracking the first frame vehicles is not enough, due to incoming and outgoing vehicles of the scope of the images. To resolve this, a merging between tracking and detection tasks is needed. The tracking of the vehicles is done at every frame, but every five frames a new detection occurs, then for each detected bounding box a measure of IoU (Intersection over Union) is done with the current tracking bounding boxes. If a detected box matches a tracking box with a relatively good percentage then it's the same box that in the append in tracking list, but the new detected bounding box has to be more accurate than the tracking one, so an adjustment operation occurs to the tracking object bounding box.

Objectives

The goal of the project is to automate the traffic signal violation detection system and make it easy for the traffic police department to monitor the traffic and take action against the violated vehicle owner in a fast and efficient way. Detecting and tracking the vehicle and their activities accurately is the main priority of the system.

Survey

1) Over 1,37,000 people were killed in road accidents in 2013 alone, that is more than the number of people killed in all our wars put together.

2) 16 children die on Indian roads daily.

3) 5 lives end on Delhi’s roads everyday.

4) There is one death every four minutes due to a road accident in India.

5) Drunken driving is one of the leading causes of road fatalities.

6) One serious road accident in the country occurs every minute and 16 die on Indian roads every hour.

7) 1214 road crashes occur every day in India.

8) Two wheelers account for 25% of total road crash deaths.

9)20 children under the age of 14 die every day due to road crashes in in the country.

10) 377 people die every day, equivalent to a jumbo jet crashing every day.

11) Two people die every hour in Uttar Pradesh – State with maximum number of road crash deaths.

12) Tamil Nadu is the state with the maximum number of road crash injuries

13) Top 10 Cities with the highest number of Road Crash Deaths (Rank –Wise):

Delhi (City)

Chennai

Jaipur

Bengaluru

Mumbai

Kanpur

Lucknow

Agra

Hyderabad

Pune

Figure 3 Accident Image

Reffernce @ https://sites.ndtv.com/roadsafety/important-feature-to-you-in-your-car-5/

Software Life Cycle Model

The **Waterfall model** is a breakdown of project activities into linear sequential phases, where each phase depends on the deliverable of the previous one and corresponds to a specialization of tasks. The approach is typical for certain areas of engineering design. In software development, it tends to be among the less iterative and flexible approaches, as progress flows in largely one direction ("downwards" like a waterfall) through the phases of conception, initiation, analysis, design, construction, testing, deployment and maintenance.

The waterfall development model originated in the manufacturing and construction industries; where the highly structured physical environments meant that design changes became prohibitively expensive much sooner in the development process. When first adopted for software development, there were no recognized alternatives for knowledge-based creative work.

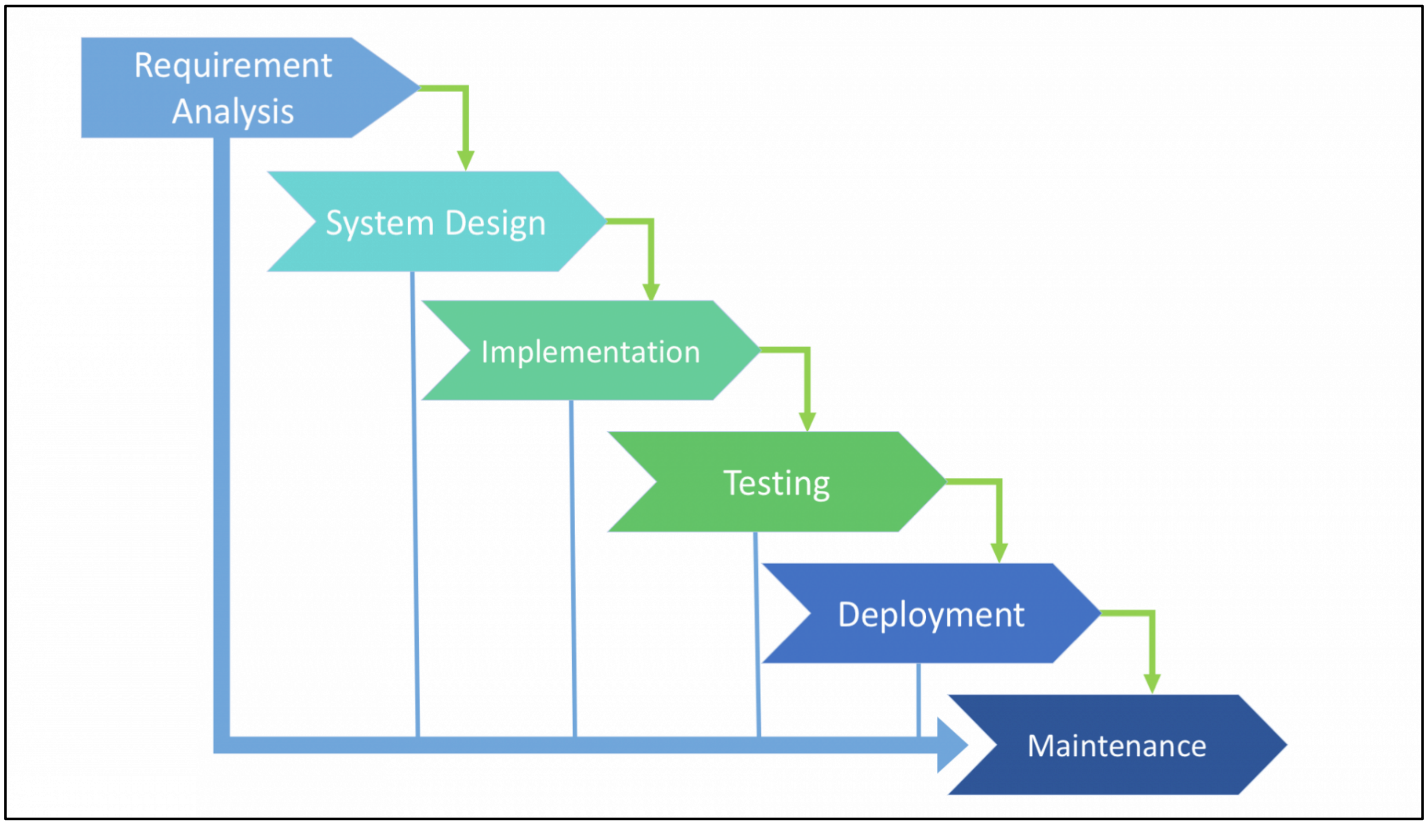


Figure 4 Software Life Cycle

Fig reff :- https://existek.com/blog/sdlc-models/

Overall Description

Dataset

Figure 5 Video Set

Technologies used

1)Python

2)Machine learning

Why python?

AI projects differ from traditional software projects. The differences lie in the technology stack, the skills required for an AI-based project, and the necessity of deep research. To implement your AI aspirations, you should use a programming language that is stable, flexible, and has tools available. Python offers all of this, which is why we see lots of Python AI projects today. From development to deployment and maintenance, Python helps developers be productive and confident about the software they're building. Benefit that make Python the best fit for machine learning and AI-based projects include simplicity and consistency, access to great libraries and frameworks for AI and machine learning (ML), flexibility, platform independence, and a wide community. These add to the overall popularity of the language. Simple and consistent Python offers concise and readable code. While complex algorithms and versatile workflows stand behind machine learning and AI, Python's simplicity allows developers to write reliable systems. Developers get to put all their effort into solving an ML problem instead of focusing on the technical nuances of the language. Additionally, Python is appealing to many developers as it's easy to learn.

Python code is understandable by humans, which makes it easier to build models for machine learning. Many programmers say that Python is more intuitive than other programming languages. Others point out the many frameworks, libraries, and extensions that simplify the implementation of different functionalities. It's generally accepted that Python is suitable for collaborative implementation when multiple developers are involved. Since Python is a general-purpose language, it can do a set of 11complex machine learning tasks and enable you to build prototypes quickly that allow you to test your product for machine learning purposes.

Extensive selection of libraries and frameworks

Implementing AI and ML algorithms can be tricky and requires a lot of time. It's vital to have a well-structured and well-tested environment to enable developers to come up with the best coding solutions. To reduce development time, programmers turn to a number of Python frameworks and libraries. A software library is pre-written code that developers use to solve common programming tasks. Python, with its rich technology stack, has an extensive set of libraries for artificial intelligence and machine learning.

Imports:

OpenCV

OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products. Being a BSD-licensed product, OpenCV makes it easy for businesses to utilize and modify the code.

Numpy:

Numpy is used to for mathematical operations. This package provides easy use of mathematical function.

What is Python good for?

Here's a table of common AI use cases and technologies that are best suited for them. We recommend using these: Platform independence Platform independence refers to a programming language or framework allowing developers to implement things on one machine and use them on another machine without any (or with only minimal) changes. One key to Python's popularity is that it's a platform independent language. Python is supported by many platforms including Linux, Windows, and macOS. Python code can be used to create standalone executable programs for most common operating systems, which means that Python software can be easily distributed and used on those operating systems without a Python interpreter. What's more, developers usually use services such as Google or Amazon for their computing needs. However, you can often find companies and data scientists who use their own machines with powerful Graphics Processing Units (GPUs) to train 13their ML models. And the fact that Python is platform independent makes this training a lot cheaper and easier. Great community and popularity In the Developer Survey 2018 by Stack Overflow, Python was among the top 10 most popular programming languages, which ultimately means that you can find and hire a development company with the necessary skill set to build your AI- based project. If you look closely at the image below, you'll see that Python is the language that people Google more than any other. In the Python Developers Survey 2017, we observe that Python is commonly used for web development. At first glance, web development prevails, accounting for over 26% of the use cases shown in the image below. However, if you combine data science and machine learning, they make up a stunning 27%.

Requirement Specification

Initial nonfunctional requirement will be:

 Getting the dataset (videos/Images) which can provide developer enough data to Test the model.

 Maintain the minimum variance and bias so the model is successfully work.

Initial functional requirement will be:

 Selecting the appropriate algorithms.

 Determining the appropriate input format to Algorithm.

Test the model.

Hardware Requirement:

 Processor: Intel i3 and above

 RAM: Minimum 6GB

 GPU: 1GB

 OS: Windows, Linux

Software Requirement:

 Anaconda Navigator

 Jupyter NoteBook / Spyder Studio Code

 Google ColabPlan of Project Execution

Plan of Project Execution

Feasible study phase

 Study of different Object detection frameworks OpenCV,Yolo,CNN,R-CNN,Faster-R-CNN,SSD,etc.

 Different types of Object detections like face, mask, dog are studied and corresponding.

After detail study, labelling in done by segregating the Videos and Images

Requirement analysis

 The Video pre-processed such as short Videos or Real time camera footage.

 There is a huge database so basically the images with better resolution and angle are selected. After selection of videos/images we should have deep knowledge about the different formats.

Design phase

 Designing algorithm for getting more accuracy and satisfaction.

 Classify the videos on the basis of Signal (Red, Green, Yellow)

Implementation phase

 Using different transformation technique Simple Thresholding, Dilation.

Testing phase

 After the model is trained successfully the software can identify the

Video species is contained in the database.

 After successful training and pre-processing, comparison of the test

videos and trained model takes place for the object detection .

System Design

Flowchart of the System:

The flowchart of the algorithm is represented in Figure

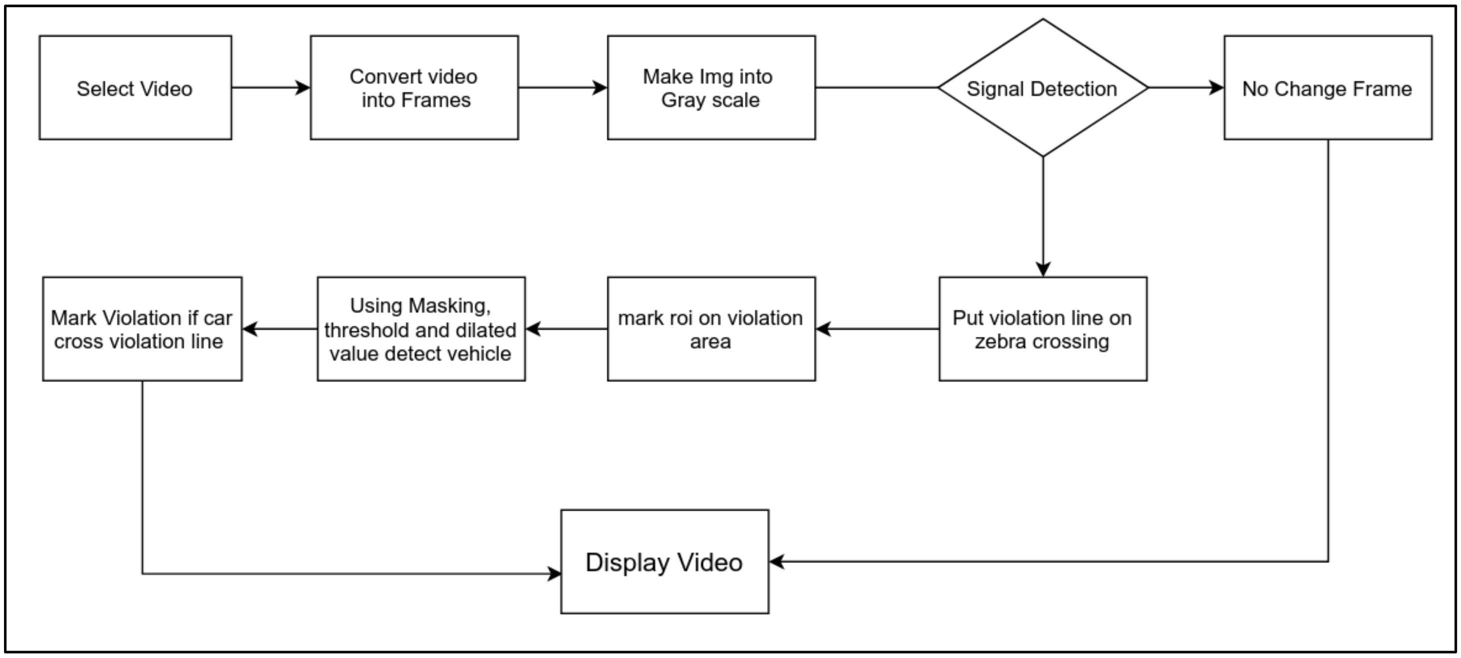

Select Video
Convert video into Frames
Make Img into Gray scale
Signal Detection
No Change Frame
Put violation line on zebra crossing
mark roi on violation area
Using Masking, threshold and dilated value detect vehicle
Mark Violation if car cross violation line
Display Video

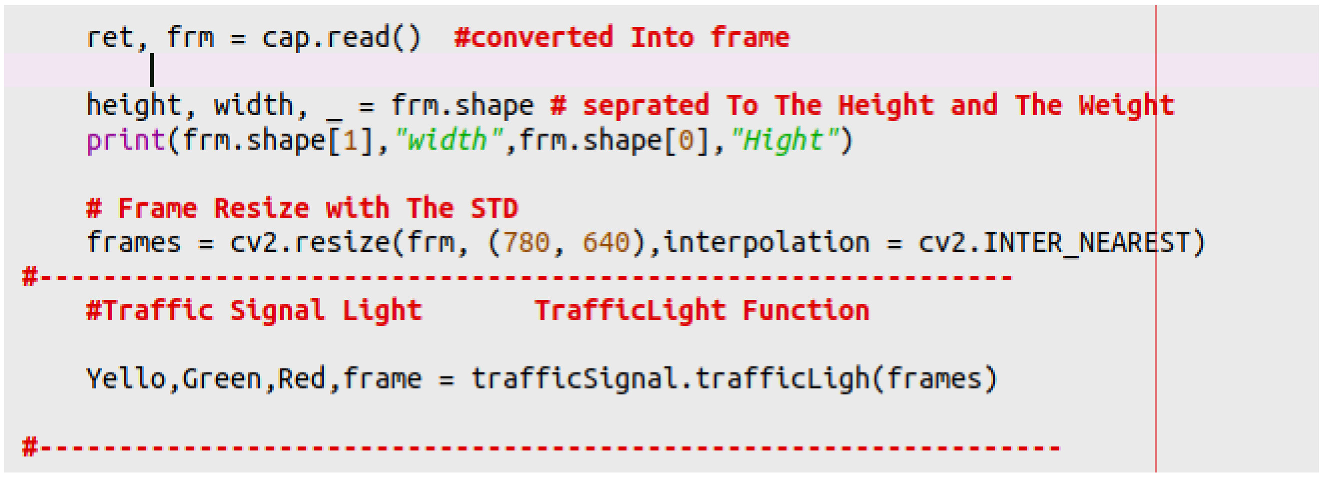
Figure 6 System Design

The above flowchart describes the working flow of the project. First convert the Video into frames then pre-processing (thresholding & dilation). Then algorithm was later implemented in python and deployed on web. The model was later tested against raw Videos.

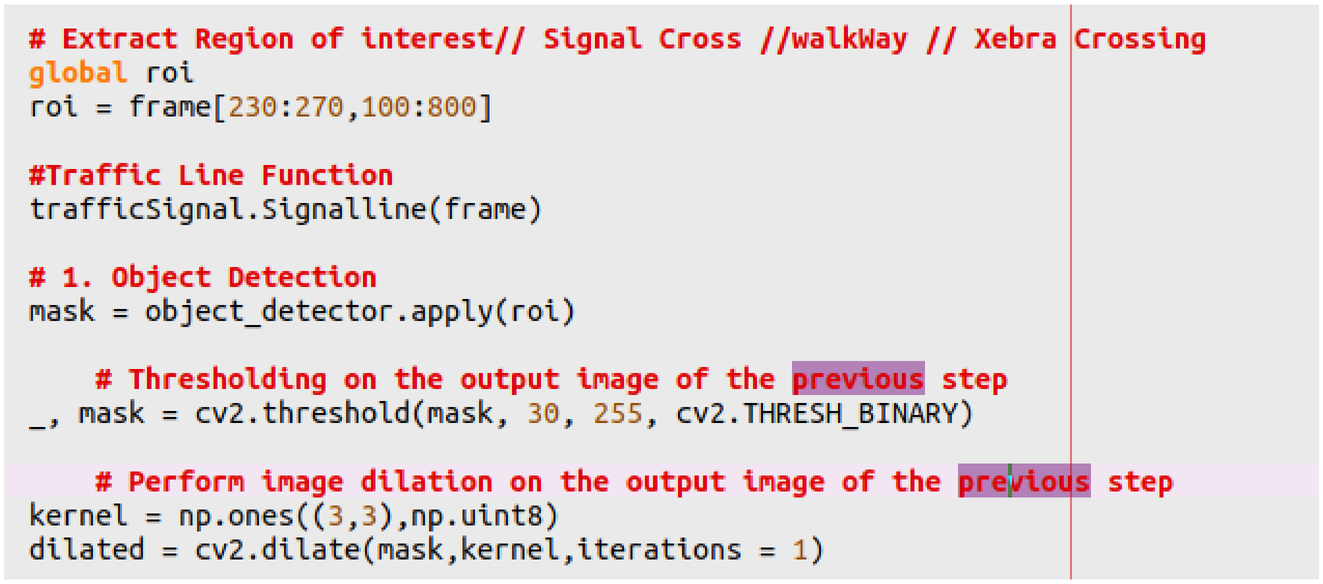
Data Pre-processing

- The video is converted into frames.
- Set height and width of input frames (780, 640).
- The frame is processed in to Signal detection using the threshold of RGB .
- Region Of Interest (RoI) for The Object Detection Area
- RoI frame converted into “Gray scale”
- RoI Converted To “Thresholing”
- RoI Converted To “Dilate”

Code :-



```
    ret, frm = cap.read()  #converted Into frame

    height, width, _ = frm.shape # seprated To The Height and The Weight
    print(frm.shape[1],"width",frm.shape[0],"Hight")

    # Frame Resize with The STD
    frames = cv2.resize(frm, (780, 640),interpolation = cv2.INTER_NEAREST)
#-----------------------------------------------------------------
    #Traffic Signal Light       TrafficLight Function

    Yello,Green,Red,frame = trafficSignal.trafficLigh(frames)

#--------------------------------------------------------------------
```



```
# Extract Region of interest// Signal Cross //walkWay // Xebra Crossing
global roi
roi = frame[230:270,100:800]

#Traffic Line Function
trafficSignal.Signalline(frame)

# 1. Object Detection
mask = object_detector.apply(roi)

    # Thresholding on the output image of the previous step
_, mask = cv2.threshold(mask, 30, 255, cv2.THRESH_BINARY)

    # Perform image dilation on the output image of the previous step
kernel = np.ones((3,3),np.uint8)
dilated = cv2.dilate(mask,kernel,iterations = 1)
```

Model Building

Algorithm Research and Selection:

Euclidean Distance

To measure Euclidean Distance in Python is to calculate the distance between two given points. These given points are represented by different forms of coordinates and can vary on dimensional space. Finding the Euclidean Distance in Python between variants also depends on the kind of dimensional space they are in.

$$distance = \sqrt{\sum_{i=0}^{n}(x_i - y_i)^2}$$

Euclidean distance or Euclidean metric is the “ordinary” straight-line distance between two points in Euclidean space.

Find the distance between the current frame and the referenced frame using Euclidean distance for each object.

If the distance between two object value is less than 50, this is the same object otherwise create a new label and assign it.

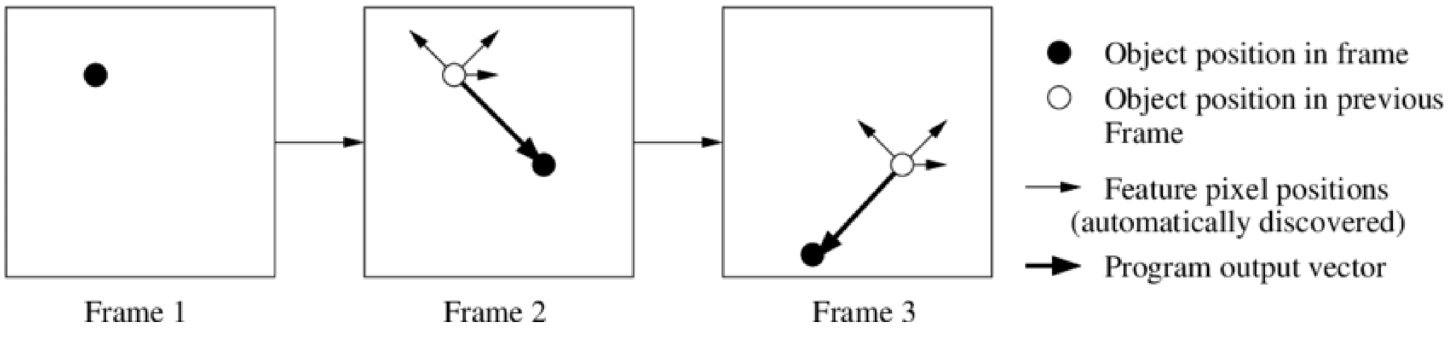

Object position in frame
Object position in previous Frame
Feature pixel positions (automatically discovered)
Program output vector
Frame 1
Frame 2
Frame 3

Figure 7 Frames 1

Detecting objects is an important task in distance measurement systems where the performance of vehicle detection algorithm acts in proportion to the distance measurement performance. Therefore, before measuring the vehicle distance, an efficient vehicle detection algorithm is applied.

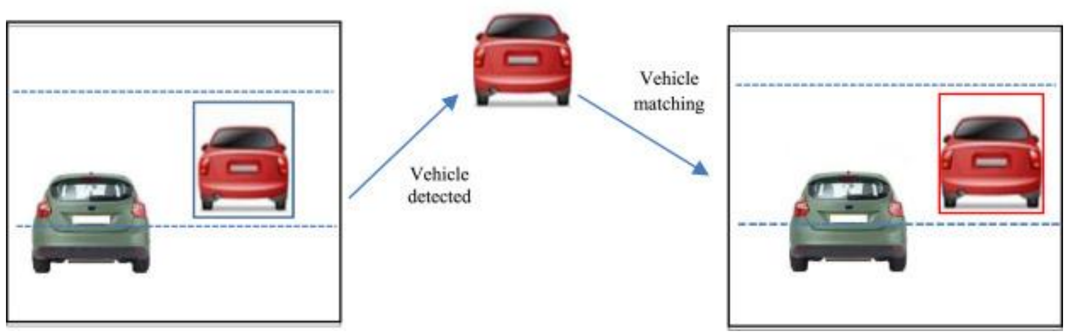
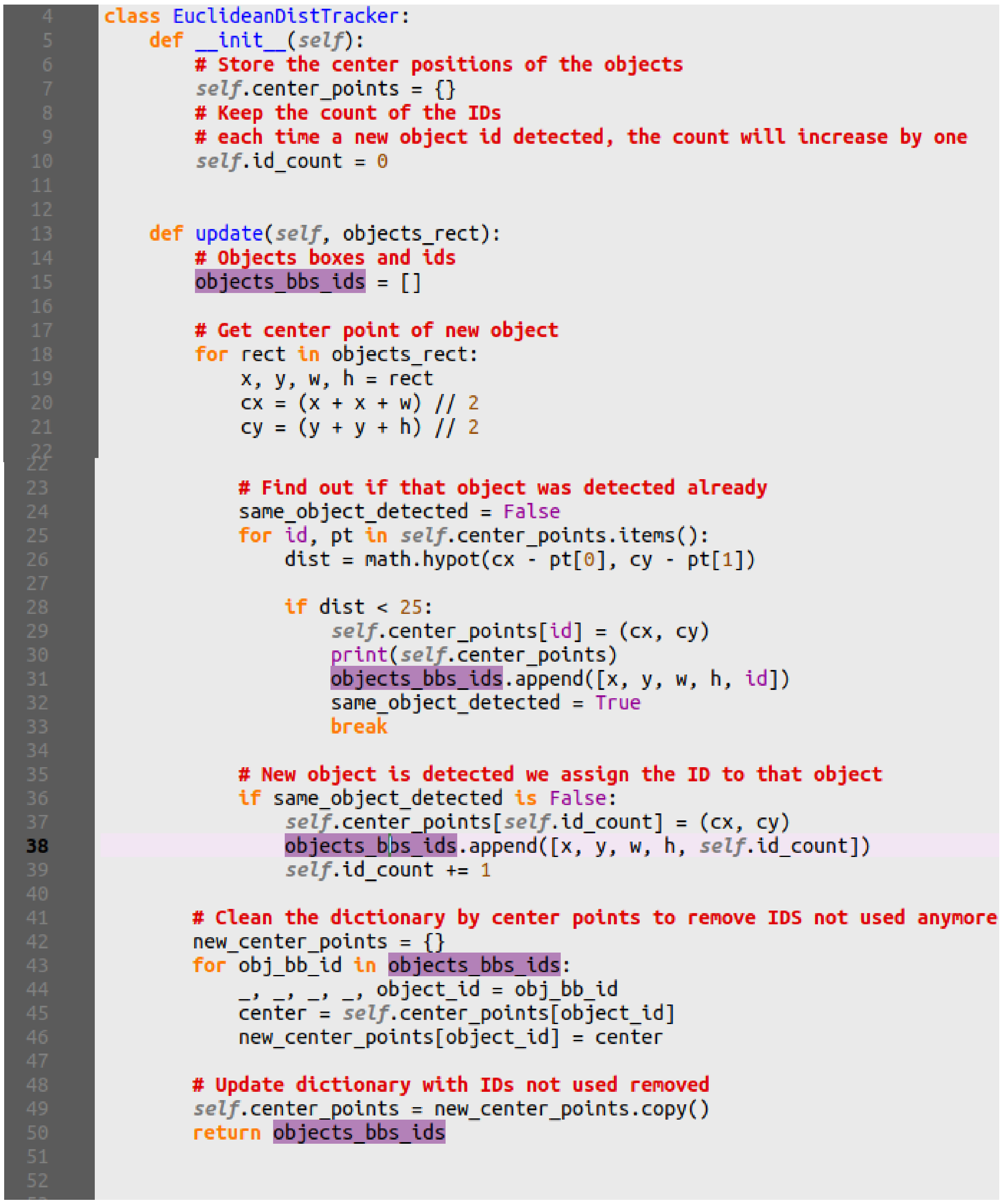


Figure 8 Frame 2

Code for Object Detection



```python
class EuclideanDistTracker:
    def __init__(self):
        # Store the center positions of the objects
        self.center_points = {}
        # Keep the count of the IDs
        # each time a new object id detected, the count will increase by one
        self.id_count = 0


    def update(self, objects_rect):
        # Objects boxes and ids
        objects_bbs_ids = []

        # Get center point of new object
        for rect in objects_rect:
            x, y, w, h = rect
            cx = (x + x + w) // 2
            cy = (y + y + h) // 2

            # Find out if that object was detected already
            same_object_detected = False
            for id, pt in self.center_points.items():
                dist = math.hypot(cx - pt[0], cy - pt[1])

                if dist < 25:
                    self.center_points[id] = (cx, cy)
                    print(self.center_points)
                    objects_bbs_ids.append([x, y, w, h, id])
                    same_object_detected = True
                    break

            # New object is detected we assign the ID to that object
            if same_object_detected is False:
                self.center_points[self.id_count] = (cx, cy)
                objects_bbs_ids.append([x, y, w, h, self.id_count])
                self.id_count += 1

        # Clean the dictionary by center points to remove IDS not used anymore
        new_center_points = {}
        for obj_bb_id in objects_bbs_ids:
            _, _, _, _, object_id = obj_bb_id
            center = self.center_points[object_id]
            new_center_points[object_id] = center

        # Update dictionary with IDs not used removed
        self.center_points = new_center_points.copy()
        return objects_bbs_ids
```

Signal Detection

HSV

cv2.HoughCircles

HSV means Hue-Saturation-Value, where the Hue is the color. Saturation is the greyness, so that a Saturation value near 0 means it is dull or grey looking. And Value is the brightness of the pixel.

HSV color space defines color with the terms Hue, Saturation, and Value. Although a mixture of Colors is used in RGB, HSV uses color, saturation and brightness values. Saturation determines the vitality of the color, while brightness refers to the brightness of the color. The HSI space separates the nephew component in a color image from the hue and saturation, which are color-bearing information.

The hue, saturation and brightness values used in HSV space are obtained from the RGB color Cube. The brightness value is zero while the color and saturation values for the Black color in the HSV space can take any between 0 and 255. In white, the brightness value is 255.

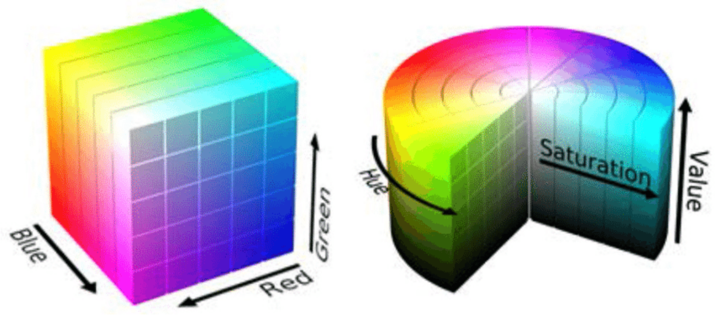


Figure 10 HSV Color Range

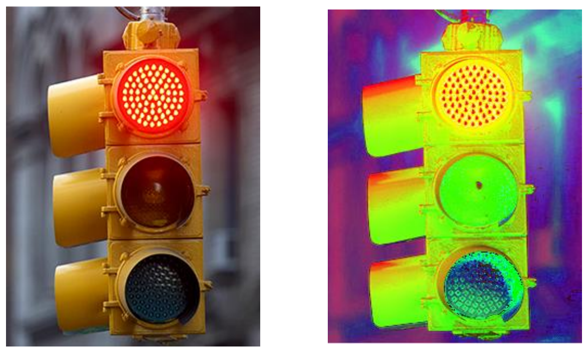


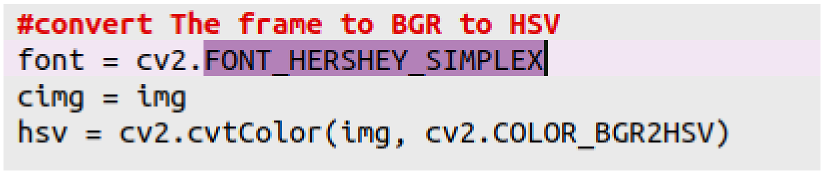
Figure 9 Original Signal Img to HSV

Fig reff:-https://www.researchgate.net/figure/RGB-left-and-HSV-right-color-spaces_fig1_310474598

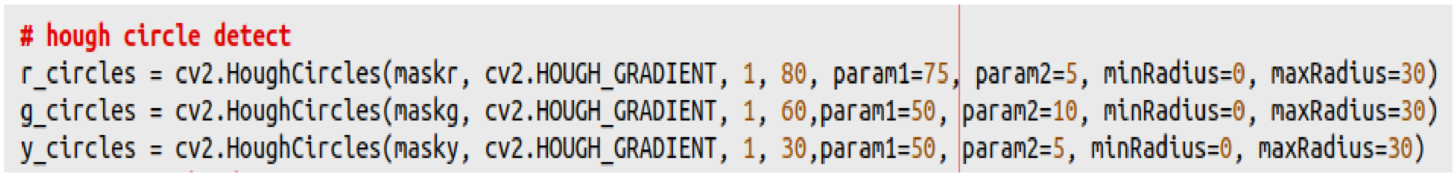
In order to detect Traffic Light we use The colour circles in frames, you’ll need to make use of the cv2.HoughCircles function. It’s definitely not the easiest function to use, but with a little explanation, I think we’ll get the hang of it.

Convert BGR to HSV

cv2.cv.HOUGH_GRADIENT method is the only circle detection method supported by OpenCV and will likely be the only method for some time), an accumulator value of 1.5 as the third argument, and finally a minDist of 100 pixels.



```
#convert The frame to BGR to HSV
font = cv2.FONT_HERSHEY_SIMPLEX
cimg = img
hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
```



```
# hough circle detect
r_circles = cv2.HoughCircles(maskr, cv2.HOUGH_GRADIENT, 1, 80, param1=75, param2=5, minRadius=0, maxRadius=30)
g_circles = cv2.HoughCircles(maskg, cv2.HOUGH_GRADIENT, 1, 60,param1=50, param2=10, minRadius=0, maxRadius=30)
y_circles = cv2.HoughCircles(masky, cv2.HOUGH_GRADIENT, 1, 30,param1=50, param2=5, minRadius=0, maxRadius=30)
```

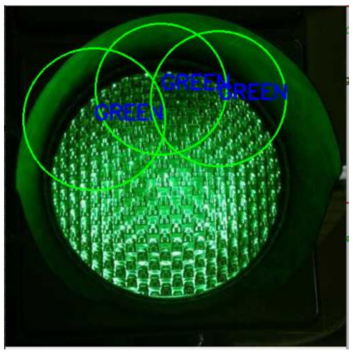


Figure 12 Green signal Detected

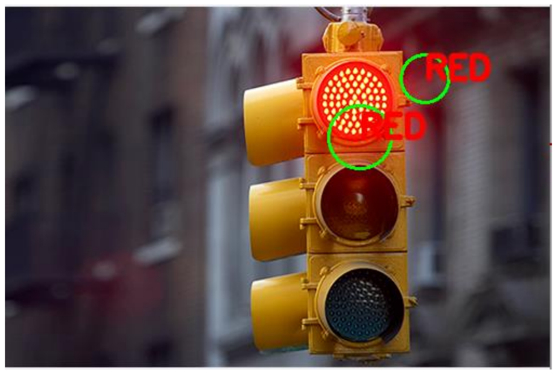
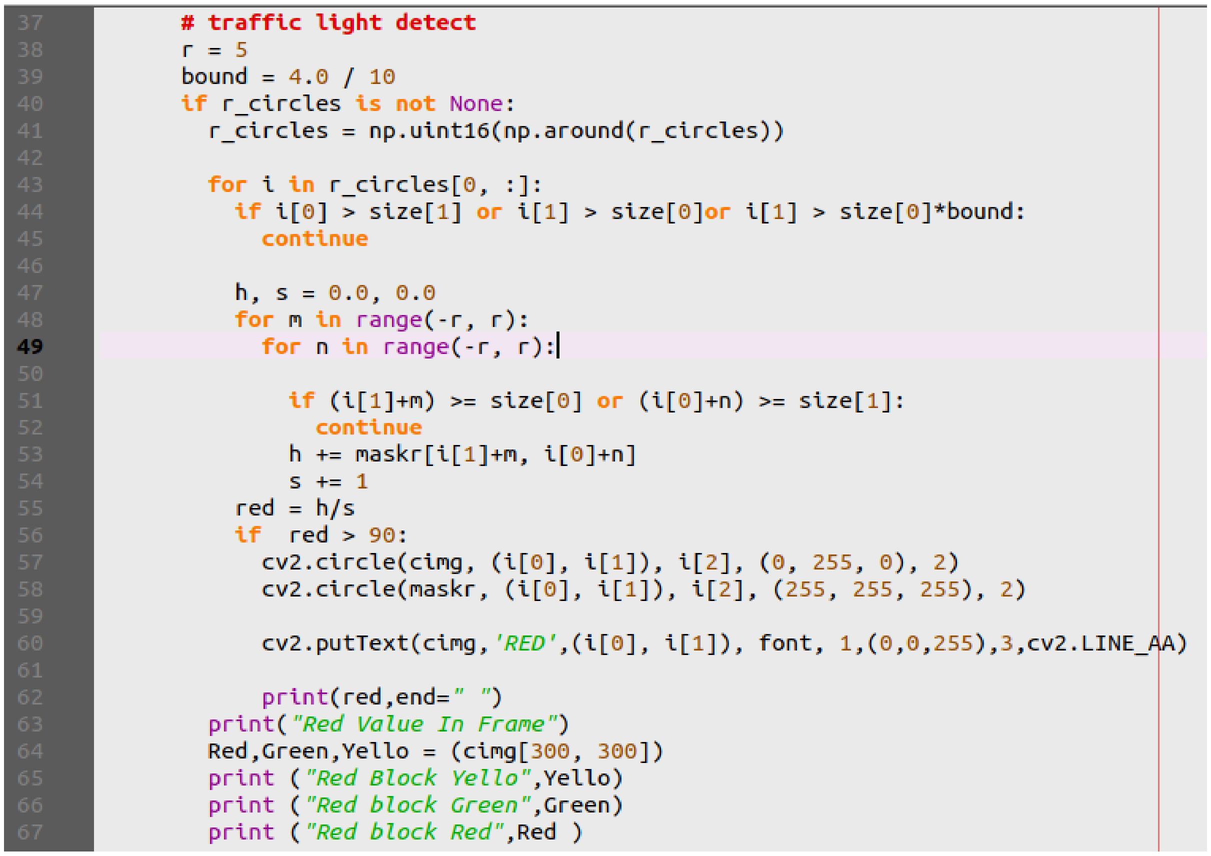


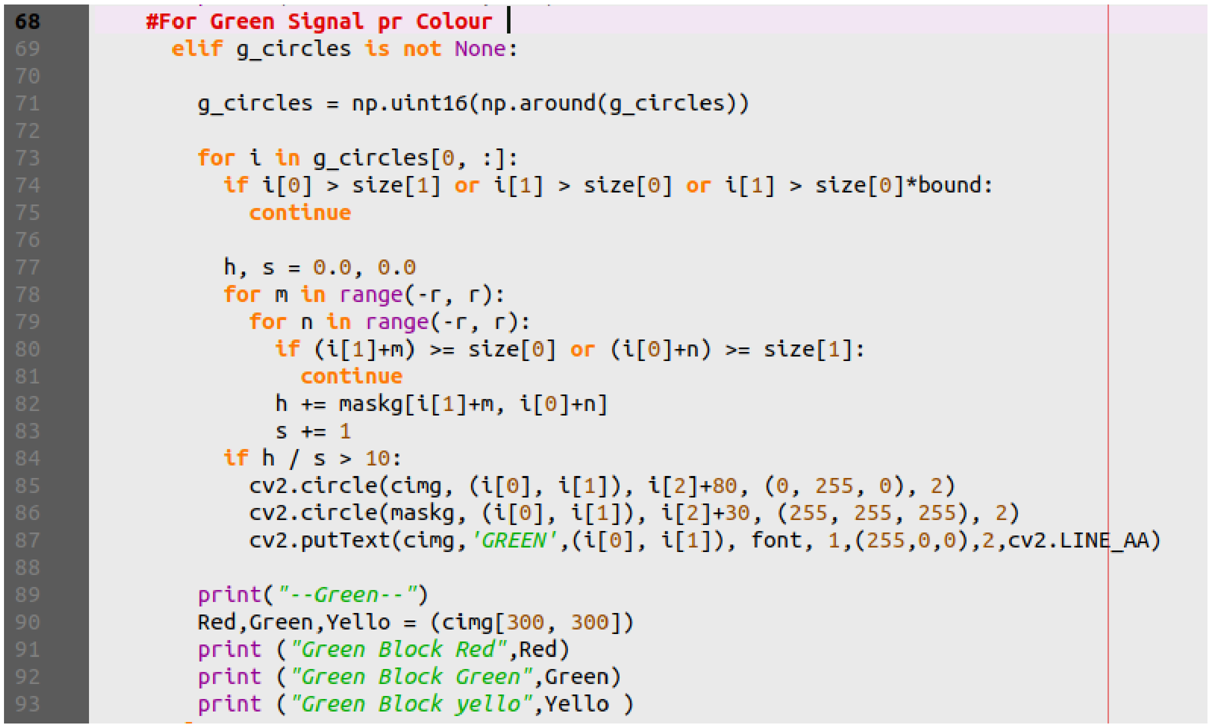
Figure 11 Red Signal Detected

Following Is the code for the Red Light Detection Screenshot:-For Red



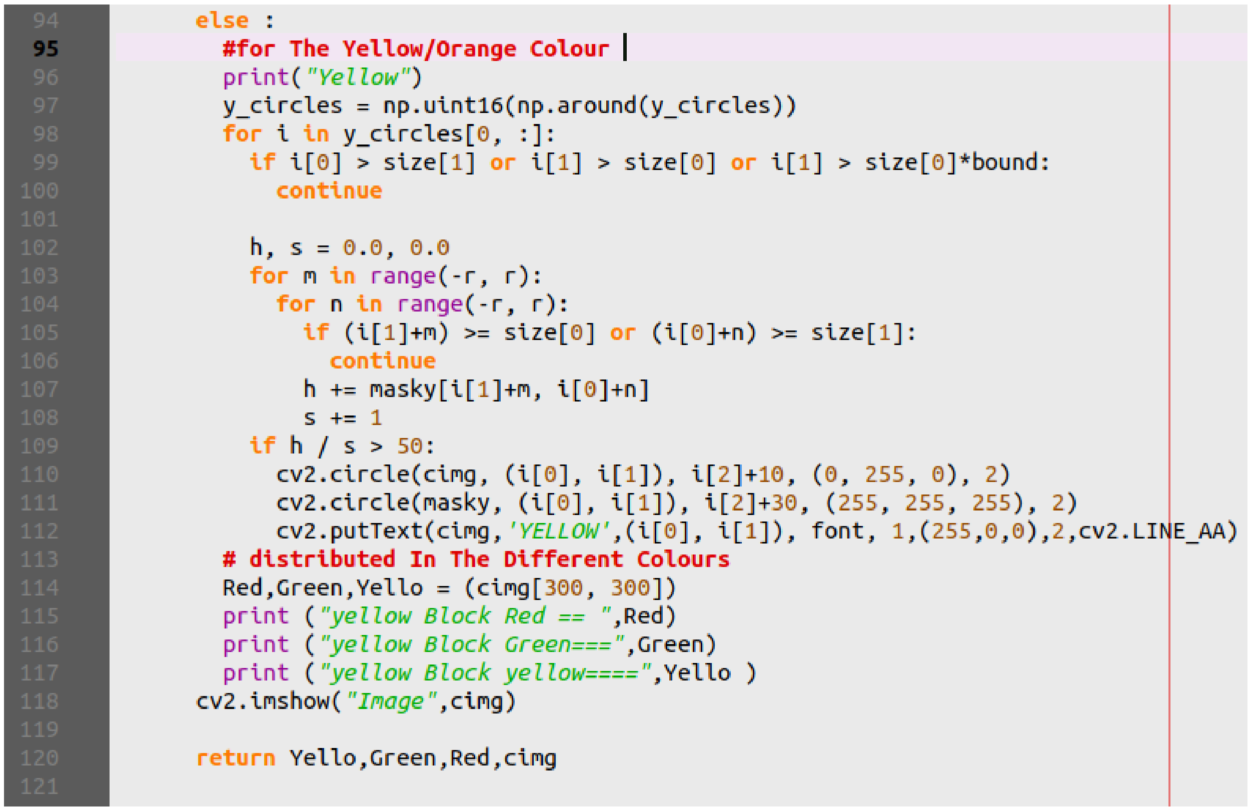
```
    # traffic light detect
    r = 5
    bound = 4.0 / 10
    if r_circles is not None:
      r_circles = np.uint16(np.around(r_circles))

      for i in r_circles[0, :]:
        if i[0] > size[1] or i[1] > size[0]or i[1] > size[0]*bound:
          continue

        h, s = 0.0, 0.0
        for m in range(-r, r):
          for n in range(-r, r):

            if (i[1]+m) >= size[0] or (i[0]+n) >= size[1]:
              continue
            h += maskr[i[1]+m, i[0]+n]
            s += 1
        red = h/s
        if  red > 90:
          cv2.circle(cimg, (i[0], i[1]), i[2], (0, 255, 0), 2)
          cv2.circle(maskr, (i[0], i[1]), i[2], (255, 255, 255), 2)

          cv2.putText(cimg,'RED',(i[0], i[1]), font, 1,(0,0,255),3,cv2.LINE_AA)

          print(red,end=" ")
      print("Red Value In Frame")
      Red,Green,Yello = (cimg[300, 300])
      print ("Red Block Yello",Yello)
      print ("Red block Green",Green)
      print ("Red block Red",Red )
```

Screenshot:-For Green



```
  #For Green Signal pr Colour
    elif g_circles is not None:

      g_circles = np.uint16(np.around(g_circles))

      for i in g_circles[0, :]:
        if i[0] > size[1] or i[1] > size[0] or i[1] > size[0]*bound:
          continue

        h, s = 0.0, 0.0
        for m in range(-r, r):
          for n in range(-r, r):
            if (i[1]+m) >= size[0] or (i[0]+n) >= size[1]:
              continue
            h += maskg[i[1]+m, i[0]+n]
            s += 1
        if h / s > 10:
          cv2.circle(cimg, (i[0], i[1]), i[2]+80, (0, 255, 0), 2)
          cv2.circle(maskg, (i[0], i[1]), i[2]+30, (255, 255, 255), 2)
          cv2.putText(cimg,'GREEN',(i[0], i[1]), font, 1,(255,0,0),2,cv2.LINE_AA)

      print("--Green--")
      Red,Green,Yello = (cimg[300, 300])
      print ("Green Block Red",Red)
      print ("Green Block Green",Green)
      print ("Green Block yello",Yello )
```

For The Yellow



```
      else :
        #for The Yellow/Orange Colour
        print("Yellow")
        y_circles = np.uint16(np.around(y_circles))
        for i in y_circles[0, :]:
          if i[0] > size[1] or i[1] > size[0] or i[1] > size[0]*bound:
            continue

          h, s = 0.0, 0.0
          for m in range(-r, r):
            for n in range(-r, r):
              if (i[1]+m) >= size[0] or (i[0]+n) >= size[1]:
                continue
              h += masky[i[1]+m, i[0]+n]
              s += 1
          if h / s > 50:
            cv2.circle(cimg, (i[0], i[1]), i[2]+10, (0, 255, 0), 2)
            cv2.circle(masky, (i[0], i[1]), i[2]+30, (255, 255, 255), 2)
            cv2.putText(cimg,'YELLOW',(i[0], i[1]), font, 1,(255,0,0),2,cv2.LINE_AA)
        # distributed In The Different Colours
        Red,Green,Yello = (cimg[300, 300])
        print ("yellow Block Red == ",Red)
        print ("yellow Block Green===",Green)
        print ("yellow Block yellow====",Yello )
      cv2.imshow("Image",cimg)

      return Yello,Green,Red,cimg
```

Design Specification

In this project, the system was implemented to accept the input as video of crossroad, which needed to observe and detect traffic red-light violations. System consists mainly of three components: violation line estimator, vehicles detector and vehicles tracker.

The figure below illustrates our proposed flow

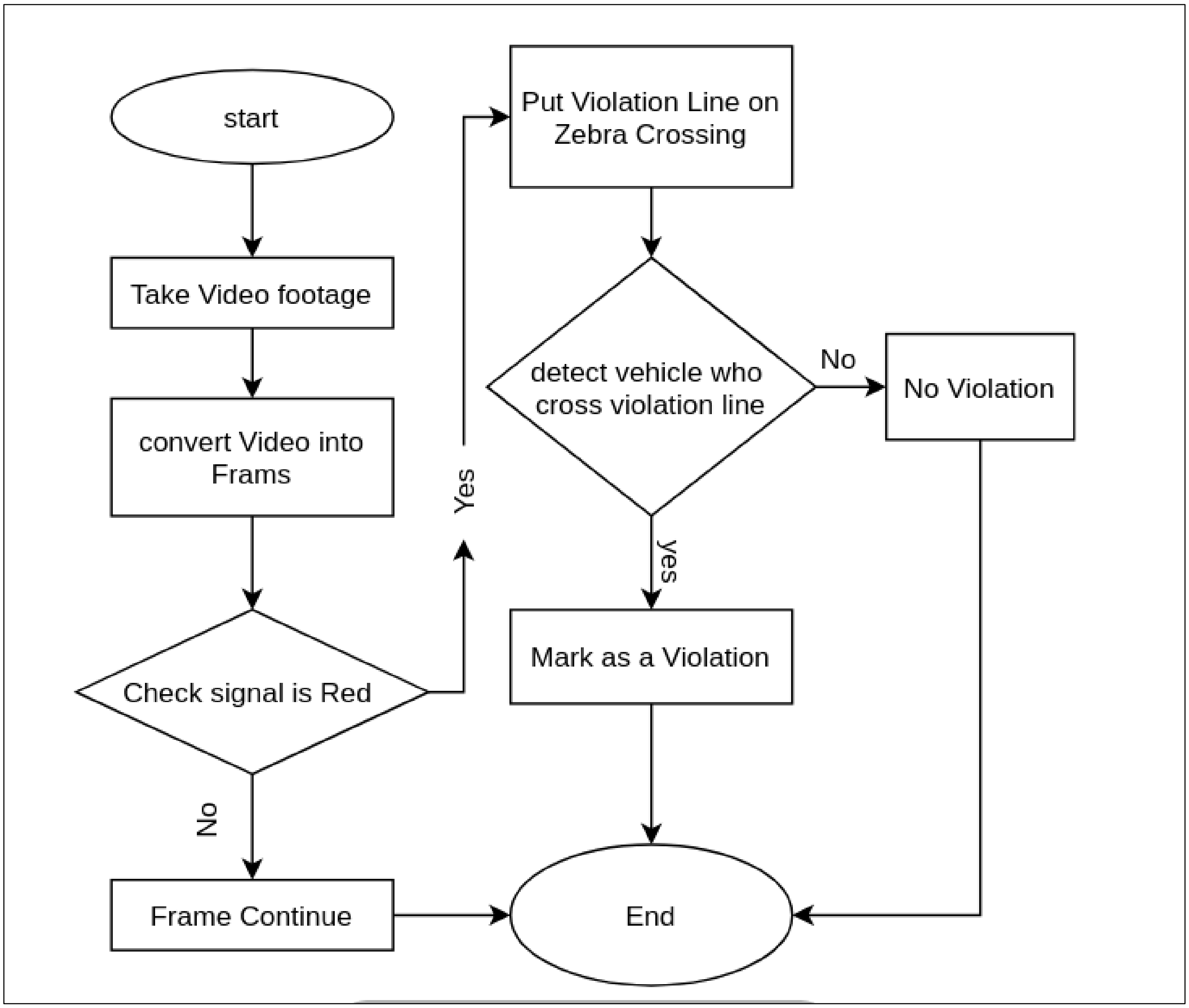


Figure 13 System Overview

Violation line estimation

A crucial part in building a system for vehicle violation detection is specifying the area for which a vehicle is considered to be in a violation state. Such task can be quite challenging and nearly unattainable due to the high levels of variability present in the input videos such as the depth of view,**missing of the zebra cross lines, single traffic line, vehicle on the lines** ,video capture perspective, location and distribution of traffic lights along with the specific description of the road. Consequently, it is clear that in order to obtain the desired region of violation, one must set direct assumptions which narrow down the generality of the problem and bring it closer to a technical level. Therefore, two main assumption regarding the input were made in order to gain knowledge about the violation region and proceed properly with the needed operations, these assumptions are:

• First, the region is to be represented as any vertical area within the video which surpasses a certain threshold horizontal line, referred to as the violation line, which splits the road into a regular and violating zone, as the main goal of the system evidentially becomes to determine the vertical location of this horizontal line.

• Second, in order to obtain clear info and indication towards the approximate vertical location of the violation line, it is required that a crosswalk exists between the regular and violating area, within the vicinity of the traffic light. The previously discussed assumptions are depicted in the following figure.

```
    #Traffic Line Function

 –-> trafficSignal.Signalline(frame)

 def Signalline(frame):

          #Red Signal Mark Line

       cv2.line(frame,(0,270),(800,270),(0,255,0),1)  ---- GreenLine  (Reference Line )

       cv2.line(frame,(0,290),(800,290),(0,0,255),3)  ---- RedLine  (Violation Line )

       cv2.line(frame,(0,310),(800,310),(0,255,0),1)  ---- GreenLine  (Reference Line )
```

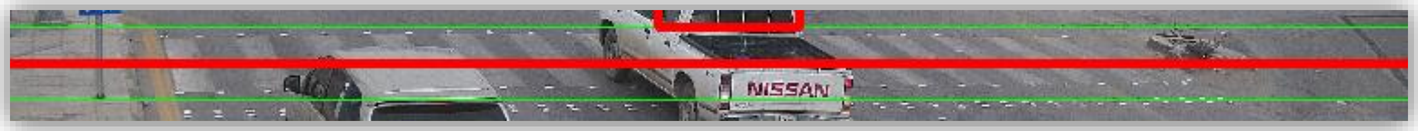


Figure 14 Violation Line

The figure shows the desired approximation of the violation line, as the location of the pedestrian cross-way is the main element taken into consideration. Therefore, as a first step it was necessary to obtain clear locations of the crosswalk components. In order to obtain locations of crosswalk components, a sequence of processing steps was done, summarized as follows:

1. Convert the image into the Frame (780, 640) .

2. As per the Video or footage we have to describe the violation line and reference. lines

3. We have to take the mean value according to signal red light and the traffic zebra crossing line.

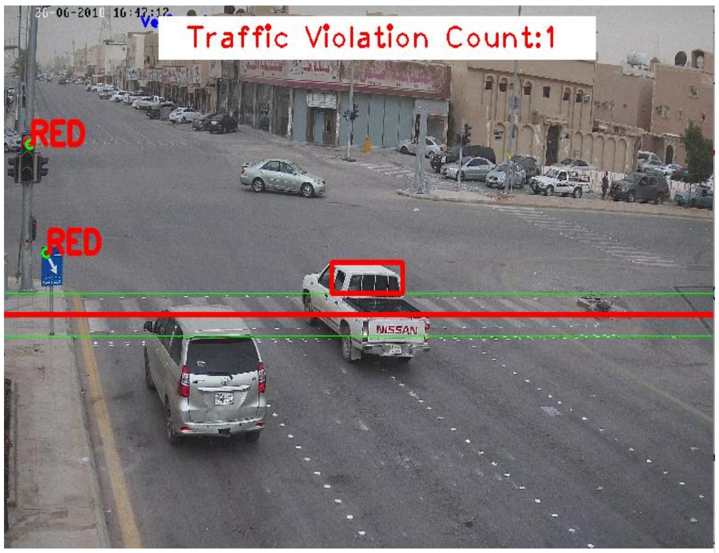


Figure 15 Violation Count 1

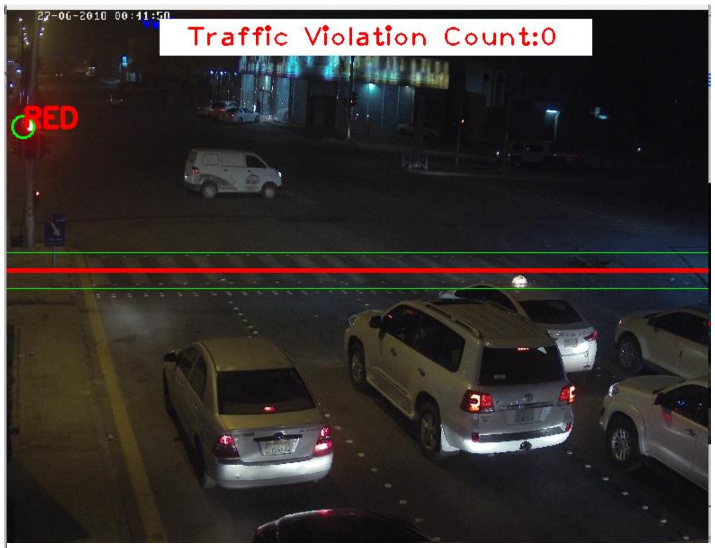
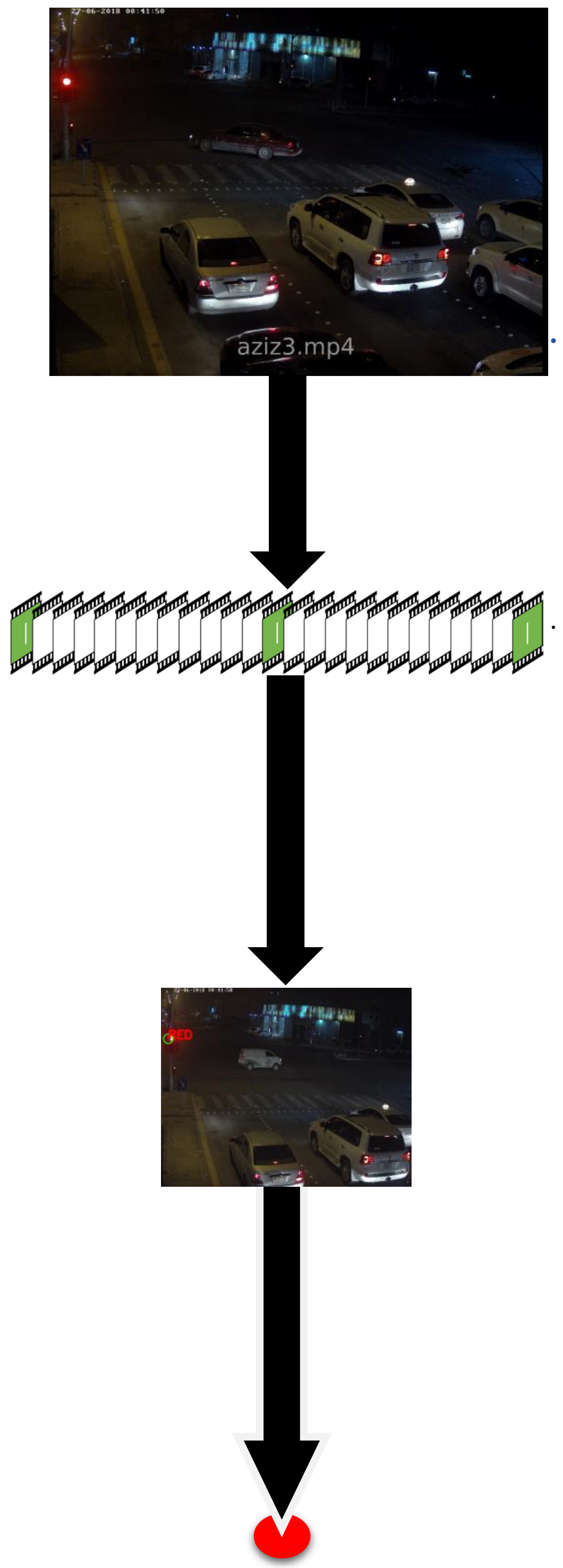


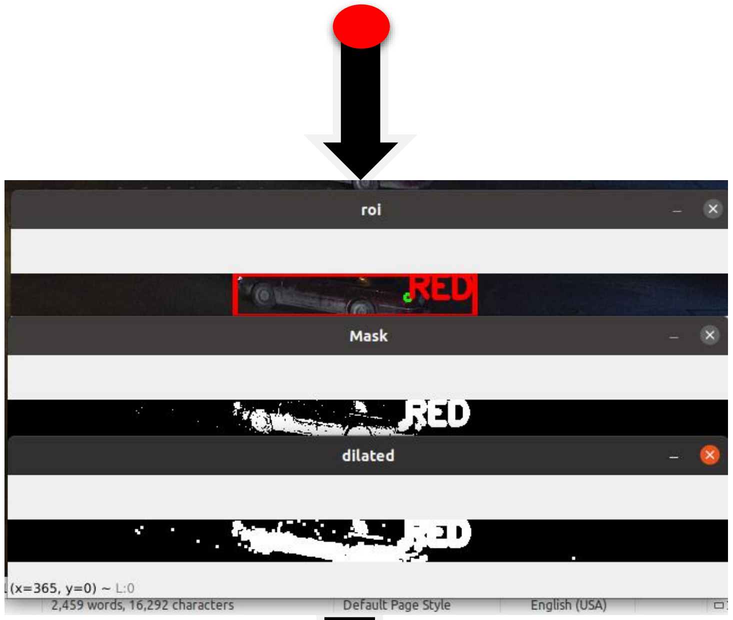
Figure 16 Violation Count 0

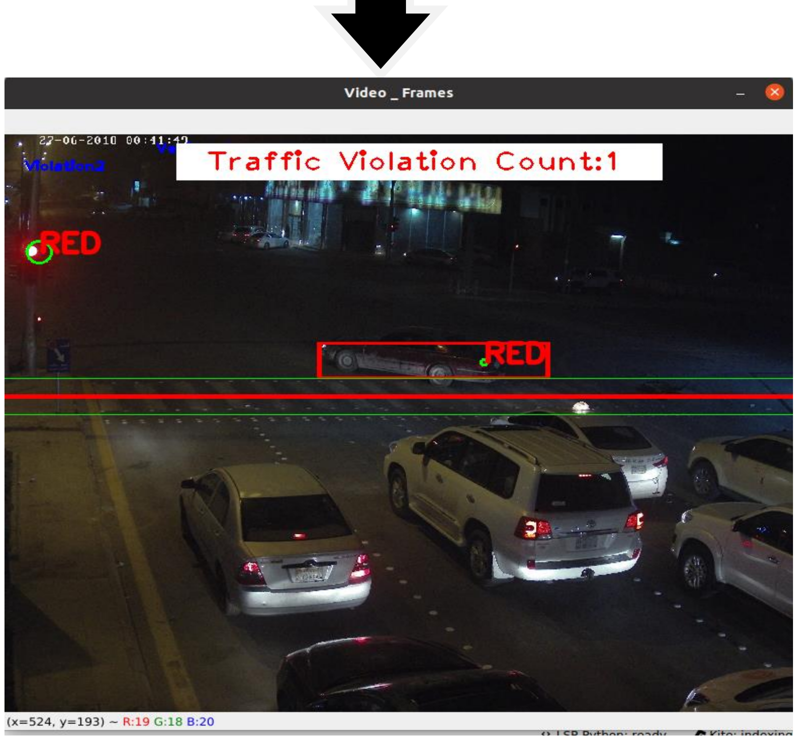
Working

1. Load the Video
2. Convert the Video into Frames
3. Load the Frame to Verify the Red Signal On or Off
4. If On the load the Frame To RoI (Region Of Interest)
5. Then Threshold The RoI Frame
6. Then dilation The RoI for Object Detection
7. Put The Signal Violation Red Line and Reference green Lines
8. Detect The object found in the RoI Frame by Thresholding Value > 800.
9. Count The Vehicle In the object In the frame detected in the RoI.

O/P




roi
Mask
dilated
RED
(x=365, y=0) ~ L:0
2,459 words, 16,292 characters
Default Page Style
English (USA)


Video _ Frames
Traffic Violation Count:1
RED
RED
(x=524, y=193) ~ R:19 G:18 B:20

Future Scope

The proposed system can be used for the real time detection of vehicles on the traffic signal, monitoring the vehicles entering and Exit the city or in specific area, parking area of mall, big Organization etc. It helps to Traffic management, transport management to take the decision . Capture the photo/image of Vehicle registration plates and store the Data in Database.

Conclusion

The designed algorithm was effectively able to detect the type of violation specified on this project which are denying traffic signal. The convergence of detection for the traffic violation mentioned is dissimilar, since it has a different threshold condition. In order to detect circles in images, you'll need to make use of the cv2.HoughCircles function. It's definitely not the easiest function to use, but with a little explanation, I think you'll get the hang of it. The system provides detection for traffic signal violation.

Reference

[1] Samir A. Elsagheer Mohamed. “Automatic Traffic Violation Recording and Reporting System to Limit Traffic Accidents” 2019 International Conference on Innovative Trends in Computer Engineering (ITCE’2019), Aswan, Egypt, 2-4 February 2019

[2] Hesson, J.; Harrington, K. “Systolic image processor for automatic

detection and recognition of traffic violations”. IEEE International

Conference on Acoustics, Speech, and Signal Processing, Volume

11, Apr 1986, pp:777 – 780.

[3] Ms. C. Sushmitha, J. Subhasree, Thushara. S.S, Mrs. V. Rajalakshmi.

“Traffic Rules Violation Detection System” International Research Journal of Engineering and Technology (IRJET)

[4] Hengliang Luo, Yi Yang, Bei Tong, Fuchao Wu, and Bin Fan. “Traffic Sign Recognition Using a Multi-Task Convolutional Neural” IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS

[5] https://sites.ndtv.com/roadsafety/important-feature-to-you-in-your-car-5/

Omkar Kumbhar

PG-Diploma in Big Data Analytics

BE Computer Engineer, APCOER, Pune

Worked on many Video/Image Processing and Machine Learning Projects

Sumeet Gadewar

PG-Diploma in Big Data Analytics

BE Computer Engineer, DYPTC, Pune

Worked on many Andriod,Flutter,Video/Image Processing and Machine Learning Projects