**CBSE PROJECT**

**COMPUTER SCIENCE(083)**

**2021-2022**

**TRAFFIC MONTORING SYSTEM**

**A picture containing text, outdoor, building, car

Description automatically generated**

By:- Devam Tanwani

Class:- XII A

**INDEX**

|  |  |  |
| --- | --- | --- |
| S.No | Description | Page No. |
|  | Certificate of Originality/ Acknowledgement | 1-2 |
|  | Synopsis   * + Problem Statement   + Unique Contribution   + Approach   + Target Audience   + Societal Benefits   + Future Scope | 3-5 |
|  | Modules used | 6 |
|  | User-defined functions used | 6 |
|  | Final Code | 7-10 |
|  | Output Screenshots | 11-13 |

**Certificate of Originality**

This is to certify that the project work titled “Traffic Monitoring System” has been presented by

Devam Tanwani.

This is a genuine work carried out under the supervision of

Ms. Priyanka Malhotra.

This also certifies that Devam Tanwani is a student of class XII at Bluebells School International,  
Kailash, New Delhi.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Signature of the Student) (Signature of Teacher In-Charge)

**Acknowledgement**

I would like to express my sincere gratitude to Ms. Priyanka Malhotra, my Computer Science

teacher. Without her guidance and support this project would not have been possible.

I would also like to thank my peers for their support and inputs which have been truly useful.

[**TRAFFIC**](https://www.pixelsolutionz.com/ai-automation/) **MONITORING SYSTEM**

**SYNOPSIS**

**Problem Statement-**

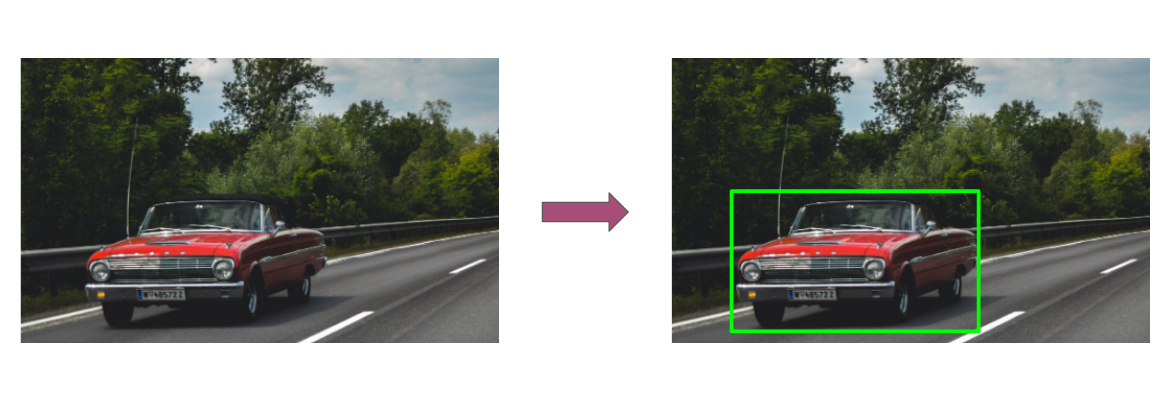
* Heavy traffic is a major problem in most of the modern cities around the world.
* In cities, where the number of vehicles continuously increases faster than the available traffic infrastructure to support them congestion is a difficult issue to deal with.
* At least 10-15 mins are wasted whilst being stuck in traffic or sometimes even more which is a lot time consuming.
* Traffic lights mainly work on a pre-recorded time but not every time this works and is ineffective during peak hours
* Rising traffic congestion is an inescapable condition in large and growing metropolitan areas across the world.
* It creates several negative concerns for the environment and society such as increasing in number of traffic accidents, economic impacts, and high levels of greenhouse emissions.
* This problem affects many aspects of the modern society, including economic development, traffic accidents, increase in greenhouse emissions, time spent, and health damages.
* In this context, modern societies can rely on traffic management system to minimize traffic congestion and its negative effects.



**Unique Contribution-**

* Reduces travel time for each and every person
* Keeps them in a better state of mind, thus increases work efficiency which helps in achieving economic growth (SDG)
* The congestion on the roads poses as an hinderance for ambulances, fire trucks and other emergency vehicles. Our project helps these vehicles reaching their destinations on time thus facilitating in the reduction of response time.

**Approach-**

* This project processes traffic images.
* Firstly, with the help of the program all the vehicles inside a given image will be identified i.e. outlined by separate boxes for each vehicle. 
* A graph would be plot with respect to the image and with that, coordinates of the boxes shall be marked
* A limit would be set for maximum distance allowed so that the time limit shall always not be too high with respect to vehicle density along a road.
* The y coordinate of the top corner of the outlined square of the vehicle which would be closest to the maximum allowed distance shall be noted distance would be measured from the starting point.
* A suitable time for the green light would be calculated in accordance with the distance measured.
* Graphical user interface, application

  Description automatically generatedWith the help of this AI traffic signals would have efficient and variable time durations.

A picture containing outdoor, city, street, way

Description automatically generated

**Target Audience-**

* Almost all people living in a city travel one place to another via roadways.
* More and more people using roadways is the root of the problem itself and it will keep growing with increase in population.
* This project is meant to smoothen the functioning of traffic signals to provide better time duration considering the density of vehicles along a road.
* Therefore, this project would be important for the functioning of society without many delays and would be of help to each and every citizen.

**Societal Benefits-**

* Economic growth
* Less delays
* Increased efficiency

With this project, we hope to bring traffic congestion to a bare minimum.

**Future Scope-**

* This project uses images right now.
* Next step for this project could be the use of high-definition CCTV cameras which could be placed along a road so it could take the pedestrian as the starting/end point and measure distance.
* These type of cameras as shown in the image which are already in place to monitor skipping of red light can be used for this project.
* The camera can take a photo of the road when only a few seconds are left in red light signal and process that image to get suitable time for green light to take on.
* There are hundreds of joins along roads whereby traffic lights work in sync. The cameras can be used for same purpose to coordinate to keep decreasing traffic on each side efficiently.

**Modules used-**

1. cv2
2. cvlib.object\_detection
3. matplotlib.pyplot
4. math
5. tabulate
6. pygame

**User-Defined Functions used-**

1. **menuDrivenSelection() -** to create a menu accessed by the user.
2. **calculateImage()** **-** to identify the vehicles in an image
3. **difference() -** to subtract the y-coordinate of the farthest car and the y- coordinate of the beginning of the image.
4. **find() -** to find the suitable time for the green light.
5. **Mysql\_connector() -** to connect python and MySQL and store data in the

form of table.

**FINAL CODE**

import cv2

import matplotlib.pyplot as plt

import cvlib as cv

from cvlib.object\_detection import draw\_bbox

from buttons import \*

import math

from tabulate import tabulate

import pygame

pygame.font.init()

pygame.mixer.init()

pygame.init()

width,height=1100,600

SCR=pygame.display.set\_mode((width,height))

pygame.display.set\_caption("TRAFFIC")

BG = pygame.transform.scale(pygame.image.load(os.path.join("Assets","background.jpg")), (width, height))

FAINT\_AQUA=(102,178,255)

FAINT\_RED=(186, 28, 28)

def menuDrivenSelection():

    MaxInputNumber = 5

    FileName = ''

    print("MENU")

    print("Type 1 for Image 1")

    print("Type 2 for Image 2")

    print("Type 3 for Image 3")

    print("Type 4 for Image 4")

    print("Type 5 for Image 5")

    global choice

    choice = input("Which image would you like to select ? ")

    while not (choice.isdigit() and 0 < int(choice) <= MaxInputNumber) :

        print("Invalid Option. Please try again")

        choice = input("Which image would you like to select ? ")

    print("User selected Image" , choice)

    choice = int(choice)

    if choice == 1:

        FileName = "1.jpg"

    elif choice == 2:

        FileName = "2.jpg"

    elif choice == 3:

        FileName = "3.jpg"

    elif choice == 4:

        FileName = "4.jpeg"

    elif choice == 5:

        FileName = "5.jpg"

    else:

        print("This file does not exist.")

    plt.imshow(cv2.cvtColor(cv2.imread(FileName), cv2.COLOR\_BGR2RGB))

    plt.show()

    while True:

        conf = input("\nAre you sure you want to select this image(Y/N)? ")

        if conf in ["y", "Y"]:

            break

        elif conf in ["n","N"]:

            menuDrivenSelection()

        else:

            print("Invalid Option. Please try again")

    calculateImage(FileName)

def difference(maxY,minY):

    diff = maxY - minY

    return diff

def calculateImage(FileName):

    global vehicle

    vehicle = 0

    im = cv2.imread(FileName)

    maxY = im.shape[0]

    # print(maxY)

    bbox, label, conf = cv.detect\_common\_objects(im)

    minY = bbox[0][1]

    maximumAllowedPixels = 150

    for i in range(len(bbox)):

        box = bbox[i]

        if box[1] < maximumAllowedPixels:

            pass

        else:

            minY = min(minY, box[1])

            if label[i] in ["car" , "bus" , "truck"]:

                vehicle += 1

    # print(minY)

    output\_image = draw\_bbox(im, bbox, label, conf)

    plt.imshow(cv2.cvtColor(output\_image, cv2.COLOR\_BGR2RGB))

    plt.show()

    # print('Number of vehciles in the image is', vehicle)

    diff = difference(maxY,minY)

    # print("difference is ", x)

    def find():

        diff = difference(maxY,minY)

        time = diff/20

        global times

        times = math.ceil(time)

        if 10 < vehicle <= 20:

            times = times

        elif 20 < vehicle <= 30:

            times += times\*0.5

        elif 30 < vehicle <= 40:

            times += times\*1

        elif vehicle > 40:

            times += times\*1.5

        if times > 60:

            times = 60

            print("The most suitable time for the green light would be" , times , "seconds")

            print("Max time is capped at the 60 second limit.")

        else:

            print("The most suitable time for the green light would be" , times , "seconds")

    find()

    def mysql\_connector():

        import mysql.connector

        mydb = mysql.connector.connect(host='localhost',user='root',passwd='52FO54',database='enigma')

        mycursor = mydb.cursor()

        d ='image'+ str(choice)

        z=str(d)

        x = vehicle

        q = maxY

        a = minY

        w = diff

        y = times

        query="""INSERT INTO traffic VALUES (%s,%s,%s,%s,%s,%s)"""

        tup1=(z, x, q, a, w, y)

        mycursor.execute(query,tup1)

        query2='Select \* from traffic where Image\_No=(%s)'

        tup2=(z,)

        mycursor.execute(query2,tup2)

        myrecords = mycursor.fetchall()

        l=[['Image\_No','No\_of\_Vehciles','MaxY','MinY','Difference','Time(sec)']]

        for x in myrecords:

            y=list(x)

            l.append(y)

        print(tabulate(l))

        mydb.commit()

        print(mycursor.rowcount,'record added')

    mysql\_connector()

##menuDrivenSelection()

def menu\_font(size):

    return pygame.font.Font("Assets/Gravedigger-8BOZ.ttf", size)

def get\_font(size):

    return pygame.font.Font("Assets/RosmatikaRegular-BWA45.ttf", size)

def main\_menu():

    run=True

    while run:

        SCR.blit(BG, (0,0))

        MOUSE\_CHECK = pygame.mouse.get\_pos()

        devam\_button=Button(image=pygame.image.load("Assets/samir.png"), pos=(550, 200),

                            text\_input="START", font=get\_font(25), base\_color="#9ae9ef", hovering\_color= FAINT\_AQUA)

        quit\_button=Button(image=pygame.image.load("Assets/samir.png"), pos=(550, 400),

                            text\_input="QUIT", font=get\_font(25), base\_color="#a70000", hovering\_color= FAINT\_RED)

        menu\_text=menu\_font(30).render("Traffic Monitoring System", True, "#eeedec")

        menu\_rect=menu\_text.get\_rect(center=(550,40))

        SCR.blit(menu\_text,menu\_rect)

        for button in [devam\_button,quit\_button]:

            button.changeColor(MOUSE\_CHECK)

            button.update(SCR)

        for event in pygame.event.get():

            if event.type==pygame.QUIT:

                run=False

                #pygame.quit()

            if event.type==pygame.MOUSEBUTTONDOWN:

                if devam\_button.checkForInput(MOUSE\_CHECK):

                    menuDrivenSelection()

                if quit\_button.checkForInput(MOUSE\_CHECK):

                    run=False

        pygame.display.update()

    pygame.quit()

main\_menu()

**OUTPUT SCREENSHOTS**

**A picture containing text, outdoor, road, street

Description automatically generated**

**Text

Description automatically generated**

**Graphical user interface, website

Description automatically generated**

****

****

**Table

Description automatically generated**

**A screenshot of a computer

Description automatically generated with medium confidence**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***