Internship Report

Classification and identification of heavy vehicles using image processing techniques

Submitted by

Hariram M
III(V) - Computer Science & Engineering
National Engineering College, Kovilpatti

as a part of the internship conducted by

Sponsored by

Ministry of Heavy Industries
Government of India

August - November, 2022

Table of Contents

Acknowledgements :

Mentor : Dr. A. Ramesh Babu

Team Members :

1. Devendra Johari
   IV - Computer Science & Engineering
   Graphic Era Deemed to be University

2. Mudit Khandelwal
   IV - Information Science & Engineering
   Dayananda Sagar College of Engineering

Abstract

In today’s world with the increasing number of vehicles day by day it’s not possible to manually keep a record of the entire vehicle.With the increasing number of vehicles in today’s world it’s not possible to manually keep a record of the entire vehicle. There need to be a man standing 24*7 to note down the number. It’s a time consuming process and requires manpower. Furthermore the data stored manually is not readable after a long time. So to overcome all these limitations here we tried to develop a system which would automatically detect the number plate and store it in its database. With the development of this system it becomes easy to keep a record and use it whenever required. The main objective here is to design an efficient automatic vehicle identification system by using vehicle number plates. The system first would capture the vehicle's image as soon as the vehicle reached the security checking area. The captured images are then extracted by using the segmentation process. Optical character recognition is used to identify the characters. The obtained data is then compared with the data stored in their database. The system is implemented and simulated on MATLAB and performance is tested on real images. This type of system is widely used in Traffic control areas, tolling, parking area.etc. This system is mainly designed for the purpose of security.

Keywords: Number Plate Recognition, Gray Processing, Image Acquisition, Image Binarization, Template Matching, Image to Text conversion, Easyocr,

Introduction

Problem Description :

The problem is to classify the various kinds of heavy vehicles from the web-camera. The classified vehicles have to be processed to capture the number plate of the vehicle. Then the captured number plate can be used to implement various systems with necessary functionalities such as registering the entry of the vehicle with vehicle registration number and time stamp value for maintaining the records of the vehicles entering into a place, etc.

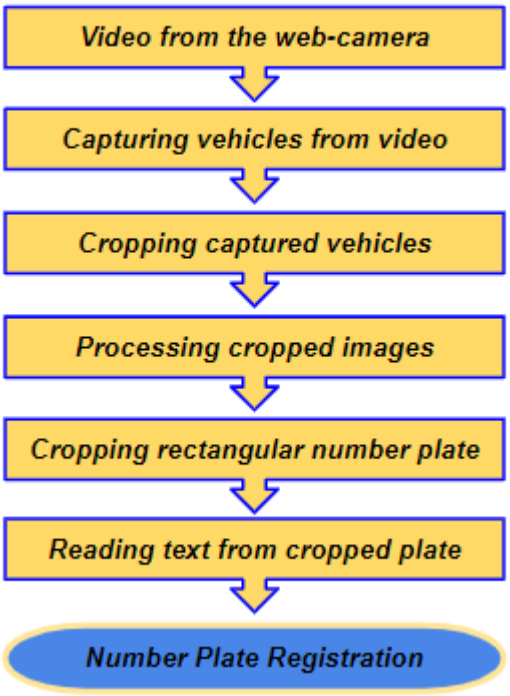
Scope & Objectives :

- This type of system is widely used in Traffic control areas, tolling, parking area .etc
- Can be used to register the number of vehicles entering a building or places like Shopping Malls, theatres, Institutions, Companies, etc.
- Can also be used for monitoring the vehicles crossing the street, houses, roads.

Work / activities carried out:

1. **Methodology adopted and workflow :**

   Here is the methodology which have been followed to implement the solution for the problem discussed.



2. **Elaborate on various activities within the workflow**

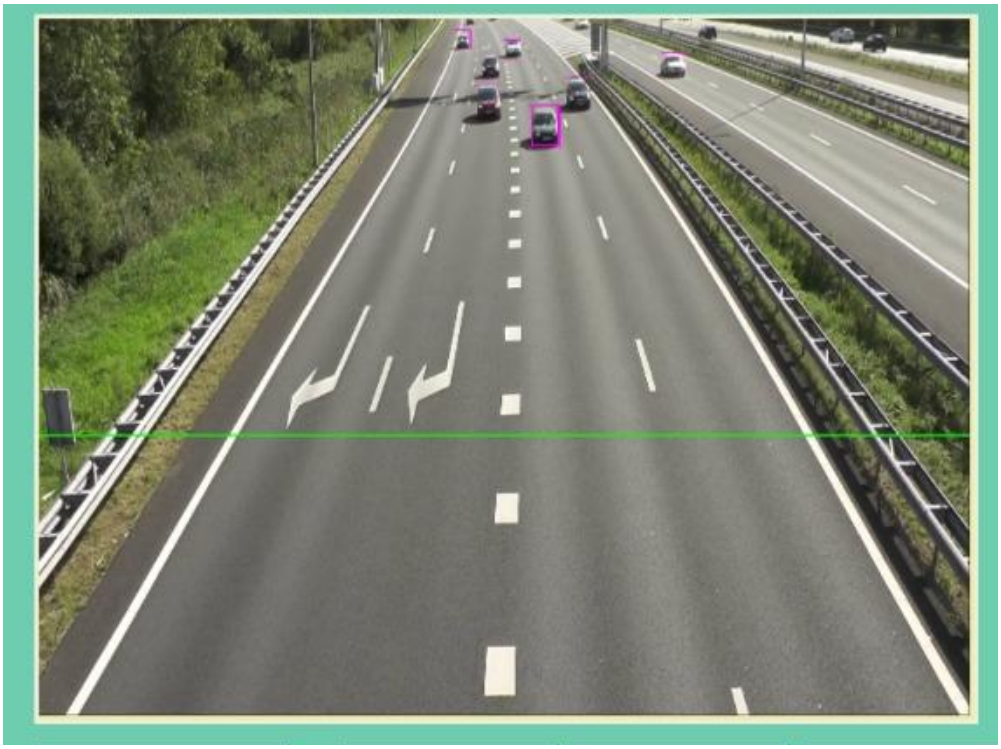
   2.1. Video from the web camera :

   - Selecting the video file from the web camera video recordings.
   - Also can directly choose web camera video streaming.

   2.2. Capturing vehicles from video

   - Searching every frame of the video for the vehicles using the yolo v5 trained model for the vehicle detection.

- A border box is drawn on the vehicles captured.
- The coordinates of the box drawn will be retrieved.



Classification of vehicles on video

2.3. Cropping captured vehicles

- The image of the vehicle is cropped from the frame of the video by using coordinates retrieved.
- The image will be directed to the processing part.

2.4. Processing cropped images

- The image cropped from the video frame is converted to gray scale and searches for the rectangular part with text in it.
- Then the rectangular border is drawn around the number plate.

Processing the vehicle classified

2.5. Cropping rectangular number plate

- The coordinates of the border drawn around the rectangular number plate is collected.
- The number plate part is cropped from the image of the vehicle by using the coordinates collected.

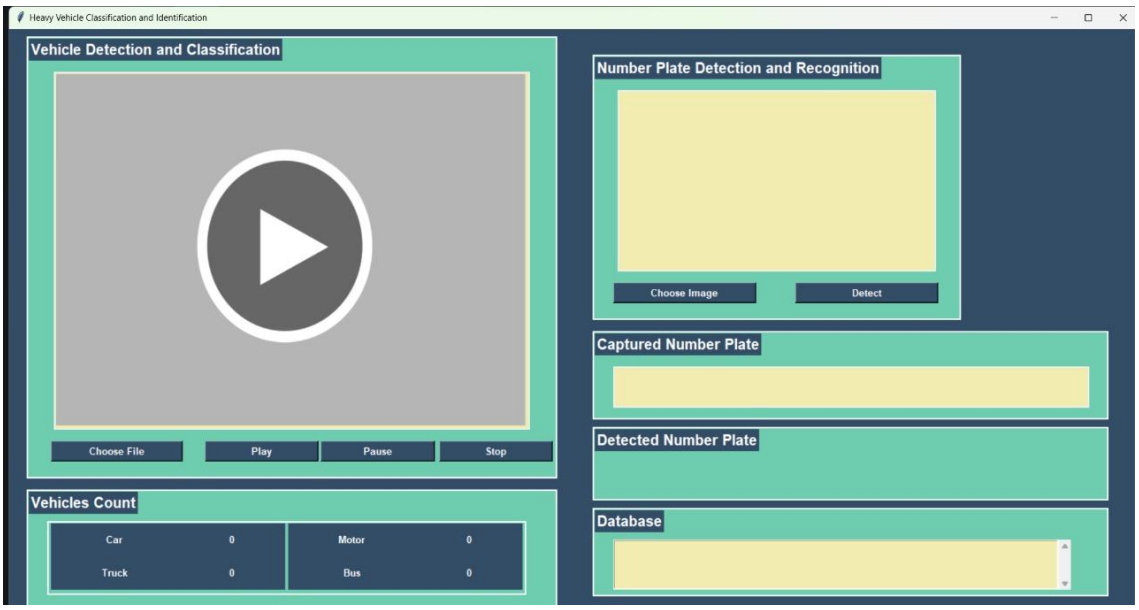
2.6. Reading text from cropped plate

- The number plate cropped image is used to find out the text.
- Easy-Ocr is used to read the text from the number plate image.
- It is displayed in the GUI.

Number Plate image & text recognised

2.7. Number Plate Registration

- The text read from the image is displayed in the GUI
- An excel sheet will be generated with the type of vehicle, Date, Time, Vehicle registration number.
- The count of different types of vehicles is also displayed on the GUI.
- The GUI also provides the facility of viewing the excel sheet within a small frame.



Final result

3. Result :

This system can be very useful for recording the entry of vehicles into any institutions, hospitals, Shopping malls, etc.

Experience/conclusion

It was a great experience going through this internship project. Got a great team to work & discussed a lot about the problems & solutions. Have got a chance to work under our Mentor. He has guided us every situation with his key ideas & has trained us to follow the pattern followed in companies. I have learned some soft skills like punctuality(in completing weekly targets before weekly discussion meetings with a mentor), Communication skills, etc. Lots of self-learning too.

References / Literature survey

[1] T.Pratheeba, “Morphology Based Text Detection and Extraction from Complex Video Scene," International Journal of Engineering and Technology Vol.2 (3), 200-206, 2010.

[2] Saeed Rastegar, Reza Ghaderi, Gholamreza Ardeshipr & Nima Asadi, " An intelligent control system using an efficient License Plate Location and Recognition Approach", International Journal of Image Processing (IJIP) Volume(3), Issue(5) 252, 2009.

[3] Wisam Al Faqheri and Syamsiah Mashohor, "A Real-Time Malaysian Automatic License Plate Recognition (M-ALPR) using Hybrid Fuzzy" ,IJCSNS International Journal of Computer Science and Network Security, VOL.9 No.2, February 2009.

[4] Satadal Saha1, Subhadip Basu, Mita Nasipuri, Dipak Kumar Basu,” License Plate Localization from Vehicle Images:An Edge Based Multistage Approach”, International Journal of Recent Trends in Engineering, Vol 1, No. 1, May 2009.

[5] Loumos, V.; Kayafas, E.,”License plate recognition from still images and video sequences: A survey” IEEE Transactions on Intelligent Transportation Systems, volume 9, issue 3, pages 377-391, September 2008.

[6] Ganapathy and W.L.D. Lui, "A Malaysian Vehicle License Plate Localization and Recognition System", Journal of Systemic, Cybernetics and Informatics, Vol. 6, No. 1, 2008.

[7] Roushdy M., "Comparative Study of Edge detection Algorithms Applying on the Grayscale Noisy Image Using Morphological filter", ICGST, International Journal of Graphics, Vision, and Image Processing GVIP, Vol. 6, Issue 4, pp. 17-23, , Dec. 2006.

[8] Chirag N. Paunwala, Suprava Patnaik, “A Novel Multiple License Plate Extraction Technique for Complex Background in Indian Traffic Conditions”, In Proceedings of International Journal of Image Processing, vol.4, issue2, 2007

[9] D. Zheng, Y. Zhao, and J. Wang, “An efficient method of License Plate location,” Pattern Recognit. Lett. vol. 26, no. 15, pp. 2431–2438, 2005.

[10] Feng Yang and Zheng Ma. “Vehicle License Plate Location Based on Histogramming and Mathematical Morphology”, Automatic Identification Advanced Technologies, pp: 89 – 94, 2005.

Source Code

```
#!/usr/bin/env python
# coding: utf-8

# In[8]:


import tkinter
import cv2
import PIL.Image, PIL.ImageTk
import time
import tkinter.font as font
from tkinter import Frame, Label, Canvas, Button, StringVar, Scrollbar, PhotoImage, Text, Listbox
from tkinter.filedialog import askopenfile
import cv2
import numpy as np
import torch
import os
import PIL.Image
import imutils
import easyocr
import csv
import sys
import sqlite3
from datetime import datetime
import warnings
warnings.filterwarnings('ignore')


# In[9]:


class MyVideoCapture:
    def __init__(self, video_source=0):
        # Open the video source
        self.vid = cv2.VideoCapture(video_source)
        if not self.vid.isOpened():
            raise ValueError("Unable to open video source", video_source)
```

```
        # Get video source width and height
        self.width = self.vid.get(cv2.CAP_PROP_FRAME_WIDTH)
        self.height = self.vid.get(cv2.CAP_PROP_FRAME_HEIGHT)

    def get_frame(self):
        if self.vid.isOpened():
            ret, frame = self.vid.read()
            if ret:
                # Return a boolean success flag and the current frame converted to BGR
                return (ret, frame)
            else:
                return (ret, [])
        else:
            return (ret, [])

    # Release the video source when the object is destroyed
    def __del__(self):
        if self.vid.isOpened():
            self.vid.release()


# In[16]:


class App:
    def __init__(self, window, window_title, video_source=0):
        self.window = window
        self.window_title = window_title
        self.video_source = video_source
        self.winWidth = self.window.winfo_screenwidth()
        self.winHeight = self.window.winfo_screenheight()

        self.initalizeVariables()
        self.modelInitialize()

        self.guiConfigure()

        if self.load_file == False:
            self.displayStartImage()

        self.window.mainloop()

    def databaseConfigure(self):
        column_name = ["vehicle_id", "vehicle_type", "vehicle_number", "timestamp"]

        rows = []
        k=1
```

```
        for i in self.result:
            rows.append([k,self.result[i]["type"], self.result[i]["vehicle_number"],
self.result[i]["timestamp"]])
            k= k+1

        with open("vehicles.csv", 'w') as csvfile:
            csvwriter = csv.writer(csvfile)
            csvwriter.writerow(column_name)
            csvwriter.writerows(rows)


    def guiConfigure(self):
        self.windowConfigure()

        self.frameConfigure()
        self.fontConfigure()
        self.labelConfigure()
        self.canvasConfigure()
        self.buttonConfigure()
        self.vehicleLabelConfigure()
        self.databaseListBoxConfigure()


    def resetAllValues(self):
        self.leftCounter = {2: 0, 3: 0, 5: 0, 7: 0}
        self.rightCounter = {2: 0, 3: 0, 5: 0, 7: 0}
        self.detectedText = ""
        self.pause = False
        self.load_file = False


    def windowConfigure(self):

        self.window.geometry(str(self.winWidth) + "x" + str(self.winHeight))
        self.window.title(self.window_title)
        self.window.configure(bg='#344D67')

        self.window.columnconfigure(0, weight=50)
        self.window.columnconfigure(1, weight=50)


    def frameConfigure(self):
         #Main Frame 1 - Second Col Frame
        self.main_frame1 = Frame(self.window, width = 750, height = 800, bg='#344D67')
        self.main_frame1.grid(column=0, row=0)
        self.main_frame1.grid_propagate(False)

        #Second Frame - Second Col Frame
```

```
        self.main_frame2 = Frame(self.window, width = 750, height = 750, bg='#344D67')
        self.main_frame2.grid(column=1, row=0)
        self.main_frame2.grid_propagate(False)

         # First Frame - Vehicle Detection Frame
        self.frame1=Frame(self.main_frame1, width = 720, height = 600, bd=1, bg='#6ECCAF',
highlightbackground="white", highlightthickness=2)
        self.frame1.grid(column=0, row=0, padx=15, pady=10, sticky=tkinter.N)
        self.frame1.grid_propagate(False)

        #Second Frame - Number Plate Detection Frame
        self.frame2 = Frame(self.main_frame2, width = 500, height = 360, bd=1,
bg='#6ECCAF', highlightbackground="white", highlightthickness=2)
        self.frame2.grid(column=1, row=0, padx=15, pady=10, sticky=tkinter.NW)
        self.frame2.grid_propagate(False)

        #Third Frame - Number Plate Photo
        self.frame3 = Frame(self.main_frame2, width = 700, height = 120, bd=1,
bg='#6ECCAF', highlightbackground="white", highlightthickness=2)
        self.frame3.grid(column=1, row=1, padx=15, pady=5)
        self.frame3.grid_propagate(False)

        #Fourth Frame - Count Frame
        self.frame4 = Frame(self.main_frame1, width = 720, height = 165, bd=1,
bg='#6ECCAF', highlightbackground="white", highlightthickness=2)
        self.frame4.grid(column=0, row=1, padx=15, pady=5)
        self.frame4.grid_propagate(False)

        self.frame5 = Frame(self.frame4, width = 650, height = 100, bd=1, bg='#6ECCAF',
highlightbackground="white", highlightthickness=2)
        self.frame5.grid(column=0, row=2, padx=25, pady=10)
        self.frame5.grid_propagate(False)

        self.frame7 = Frame(self.frame5, width = 325, height = 90,bg='#344D67')
        self.frame7.grid(column=0, row=0, padx=2)
        self.frame7.grid_propagate(False)

        self.frame8 = Frame(self.frame5, width = 325, height = 90, bg='#344D67')
        self.frame8.grid(column=1, row=0, padx=2)
        self.frame8.grid_propagate(False)

        self.frame9 = Frame(self.main_frame2, width = 185, height = 360, bg='#344D67',
highlightbackground="#344D67", highlightthickness=2)
        self.frame9.grid(column=1, row=0, pady=10, padx=15, sticky=tkinter.NE)
        self.frame9.grid_propagate(False)

        self.frame10 = Frame(self.main_frame2, width = 700, height = 120, bd=1,
bg='#6ECCAF', highlightbackground="white", highlightthickness=2)
```

```
        self.frame10.grid(column=1, row=3, padx=15, pady=5)
        self.frame10.grid_propagate(False)

        #Sixth Frame
        self.frame6 = Frame(self.main_frame2, width = 700, height = 100, bd=1,
bg='#6ECCAF', highlightbackground="white", highlightthickness=2)
        self.frame6.grid(column=1, row=2, padx=15, pady=5)
        self.frame6.grid_propagate(False)

        self.frame1.columnconfigure(0, weight=1)
        self.frame1.columnconfigure(1, weight=1)
        self.frame1.columnconfigure(2, weight=1)
        self.frame1.columnconfigure(3, weight=1)

        self.frame2.columnconfigure(0, weight=1)
        self.frame2.columnconfigure(1, weight=1)
        self.frame2.columnconfigure(2, weight=1)

        self.frame5.columnconfigure(0, weight=1)
        self.frame5.columnconfigure(1, weight=1)

        self.frame7.rowconfigure(0, weight=1)
        self.frame7.rowconfigure(1, weight=1)

        self.frame8.rowconfigure(0, weight=1)
        self.frame8.rowconfigure(1, weight=1)

        self.frame7.columnconfigure(0, weight=1)
        self.frame7.columnconfigure(1, weight=1)

        self.frame8.columnconfigure(0, weight=1)
        self.frame8.columnconfigure(1, weight=1)

        self.frame9.rowconfigure(0, weight=1)
        self.frame9.rowconfigure(1, weight=1)
        self.frame9.rowconfigure(2, weight=1)
        self.frame9.rowconfigure(3, weight=1)


    def fontConfigure(self):
        self.hText = font.Font(family='FangSong',size=15,weight='bold')
        self.bText = font.Font(family='FangSong',size=10,weight='bold')

    def labelConfigure(self):
        self.label1 = Label(self.frame1, text="Vehicle Detection and Classification",
bg="#344D67",fg="white", font=self.hText)
        self.label1.grid(row=0, column=0, sticky=tkinter.W, columnspan=2)
```

```
        self.label2 = Label(self.frame2, text="Number Plate Detection and Recognition",
bg="#344D67",fg="white", font=self.hText)
        self.label2.grid(row=0, column=0, sticky=tkinter.W, columnspan=4)

        self.label3 = Label(self.frame3, text="Captured Number Plate",
bg="#344D67",fg="white", font=self.hText)
        self.label3.grid(row=0, column=0, sticky=tkinter.W)

        self.label4 = Label(self.frame4, text="Vehicles Count", bg="#344D67",fg="white",
font=self.hText)
        self.label4.grid(row=0, column=0, sticky=tkinter.W)

        self.label6 = Label(self.frame6, text="Detected Number Plate",
bg="#344D67",fg="white", font=self.hText)
        self.label6.grid(row=0, column=0, sticky=tkinter.W)

        self.label7 = Label(self.frame10, text="Database", bg="#344D67",fg="white",
font=self.hText)
        self.label7.grid(row=0, column=0, sticky=tkinter.NW)

        self.number_plate = Label(self.frame6, text=self.detectedText,
bg="#6ECCAF",fg="white", font=self.hText)
        self.number_plate.grid(row=1, column=2, padx=5, pady=10, sticky=tkinter.EW)

    def canvasConfigure(self):
        #Display Vehicles Video
        self.canvas1 = Canvas(self.frame1, width = 640, height = 480, bd=1, bg='#F3ECB0')
        self.canvas1.grid(row=1,column=0, padx=30, pady=15, columnspan=4)

        #Display Image
        self.canvas2 = Canvas(self.frame2, width = 426, height = 240, bd=1, bg='#F3ECB0')
        self.canvas2.grid(row=1,column=0, padx=25, pady=15, columnspan=4)

        # Number Plate Captured
        self.canvas3 = Canvas(self.frame3, width = 640, height = 50, bd=1, bg='#F3ECB0')
        self.canvas3.grid(row=1,column=0, padx=25, pady=15)

        # Number Plate Captured
        self.canvas4 = Canvas(self.frame10, width = 640, height = 50, bd=1, bg='#F3ECB0')
        self.canvas4.grid(row=1,column=0, padx=25, pady=10)

    def buttonConfigure(self):
        #open File Button
        self.open_video_btn = Button(self.frame1, text='Choose File', height= 1,
width=23,bg="#344D67",fg="white", command=self.select_video, font=self.bText)
        self.open_video_btn.grid(row=2,column=0, sticky=tkinter.W, padx=30)

        #play Button
```

```
        self.play_video_btn = Button(self.frame1, text='Play', command=self.playVideo, height=
1, width=20,bg="#344D67",fg="white",font=self.bText)
        self.play_video_btn.grid(row=2,column=1)

        self.pause_video_btn = Button(self.frame1, text='Pause', height=
1,command=self.pauseVideo, width=20,bg="#344D67",fg="white",font=self.bText)
        self.pause_video_btn.grid(row=2,column=2, padx=3)

        self.stop_video_btn = Button(self.frame1, text='Stop', height=
1,command=self.stopVideo, width=20,bg="#344D67",fg="white",font=self.bText)
        self.stop_video_btn.grid(row=2,column=3, padx=3)

        self.open_photo_btn = Button(self.frame2, text='Choose Image', height= 1,
width=23,bg="#344D67",fg="white", command=self.open_photo ,font=self.bText)
        self.open_photo_btn.grid(row=2,column=0, sticky=tkinter.W, padx=25)

        self.detect_plate_btn = Button(self.frame2, text='Detect', height= 1,
width=23,bg="#344D67",fg="white",command=self.capture ,font=self.bText)
        self.detect_plate_btn.grid(row=2,column=1, padx=10)

    def vehicleLabelConfigure(self):
        self.car_label = Label(self.frame7, text="Car",bg="#344D67",fg="white",font=self.bText)
        self.car_label.grid(row=0,column=0)

        self.car_label_text = Label(self.frame7, text=str(self.leftCounter[2] +
self.rightCounter[2]),bg="#344D67",fg="white",font=self.bText)
        self.car_label_text.grid(row=0,column=1)

        self.truck_label = Label(self.frame7,
text="Truck",bg="#344D67",fg="white",font=self.bText)
        self.truck_label.grid(row=1,column=0)

        self.truck_label_text = Label(self.frame7, text=str(self.leftCounter[7] +
self.rightCounter[7]),bg="#344D67",fg="white",font=self.bText)
        self.truck_label_text.grid(row=1,column=1)

        self.motor_label = Label(self.frame8,
text="Motor",bg="#344D67",fg="white",font=self.bText)
        self.motor_label.grid(row=0,column=0)

        self.motor_label_text = Label(self.frame8, text=str(self.leftCounter[3] +
self.rightCounter[3]),bg="#344D67",fg="white",font=self.bText)
        self.motor_label_text.grid(row=0,column=1)

        self.bus_label = Label(self.frame8, text="Bus",bg="#344D67",fg="white",font=self.bText)
        self.bus_label.grid(row=1,column=0)
```

```
        self.bus_label_text = Label(self.frame8, text=str(self.leftCounter[5] +
self.rightCounter[5]),bg="#344D67",fg="white",font=self.bText)
        self.bus_label_text.grid(row=1,column=1)

    def databaseListBoxConfigure(self):
        # attaching it to root window
        self.listbox = Listbox(self.canvas4, width=100, height=4)
        self.listbox.grid(row=0, column=0, sticky=tkinter.NS)
        # attaching it to root window
        self.scrollbar = Scrollbar(self.canvas4)
        self.scrollbar.grid(row=0, column=1, sticky=tkinter.NS)
        # Insert elements into the listbox
        self.listbox.config(yscrollcommand = self.scrollbar.set, background="#F3ECB0",
selectbackground="#ADE792", fg="#212121")

        self.scrollbar.config(command = self.listbox.yview)


    def initalizeVariables(self):
        self.pause = True
        self.load_file = False
        self.filename=StringVar()
        self.detectedText=""
        self.leftCounter = {
            2: 0, # car
            3: 0, # motor
            5: 0, # bus
            7: 0 # truck
        }
        self.rightCounter = {
            2: 0, # car
            3: 0, # motor
            5: 0, # bus
            7: 0 # truck
        }

        self.result = {}
        self.yHeight = 40


        self.ListBBox=[]
        self.delay=15

    def modelInitialize(self):
        self.model = torch.hub.load('ultralytics/yolov5','yolov5s', _verbose=False)
        self.model.classes = [2, 3, 5, 7]
        self.model.conf =0.4
        self.model.iou=0.8
```

```
    def VehicleDetection(self, frame):
        results = self.model(frame)
        resBBox = results.pandas().xyxy[0]
        DetectedBBox = resBBox.values.tolist()
        BBox = []
        pmin = np.array([-1, -1])
        pmax = np.array([-1, -1])
        pmid = (pmin+pmax)/2
        BBox.append([pmin, pmax, pmid, -1, -1, -1, -1])
        for xmin, ymin, xmax, ymax, conf, cl, nama in DetectedBBox:
            IdPrev = -1
            xmid = (xmin+xmax)/2
            ymid = (ymin+ymax)/2
            BBox.append([xmin, ymin, xmax, ymax, xmid,
                    ymid, conf, cl, nama, IdPrev])
        return BBox

    def VehicleTracking(self):
        if len(self.ListBBox) >= 2:
            CurrentBBox = self.ListBBox[-1]
            PrevBBox = self.ListBBox[-2]
            for IndexLast in range(1, len(CurrentBBox)):

                xminc, yminc, xmaxc, ymaxc, xmidc, ymidc,  confc, clc, namac, IdPrevC =
CurrentBBox[
                    IndexLast]
                rCocok = 100000000000
                IndexCocok = -1
                for IndexPrev in range(1, len(PrevBBox)):

                    xminp, yminp, xmaxp, ymaxp, xmidp, ymidp,  confp, clp, namep, IdPrevp =
PrevBBox[
                        IndexPrev]
                    v = np.array([xmidc-xmidp, ymidc-ymidp])
                    RTot = np.linalg.norm(v)
                    if IndexCocok == -1:
                        rCocok = RTot
                        IndexCocok = IndexPrev
                    else:
                        if RTot < rCocok:
                            rCocok = RTot
                            IndexCocok = IndexPrev
                if IndexCocok > -1:
                    self.ListBBox[-1][IndexLast][9] = IndexCocok
        return self.ListBBox
```

```
    def printVehicleImage(self, frame ,xminc, yminc, xmaxc, ymaxc):
        x = int(xminc)
        y = int(yminc)
        w = int(xmaxc)
        h = int(ymaxc)
        vehicle_image = frame[y:h, x:w]
        return vehicle_image



    def printVehicleType(self,frame, clc, xminc, yminc, xmaxc, ymaxc):
        vehicle_info = {}
        vehicle_type = ""
        if clc == 2:
            vehicle_type = "Car"
        elif clc == 3:
            vehicle_type = "Motor"
        elif clc == 5:
            vehicle_type = "Bus"
        elif clc == 7:
            vehicle_type = "Truck"
        vehicle_info["type"] = vehicle_type
        vehicle_info["image"] = self.printVehicleImage(frame, xminc, yminc, xmaxc, ymaxc)
        vehicle_info["timestamp"] = str(datetime.now().time())
        return vehicle_info, vehicle_type


    def VehicleCounting(self, frame, MidLineY):
        if len(self.ListBBox) >= 2:
            CurrentBBox = self.ListBBox[-1]
            PrevBBox = self.ListBBox[-2]
            for IndexLast in range(1, len(CurrentBBox)):
                xminc, yminc, xmaxc, ymaxc, xmidc, ymidc,  confc, clc, namac, IdPrevC =
CurrentBBox[
                    IndexLast]

                if IdPrevC > -1:
                    xminm, yminm, xmaxm, ymaxm, xmidm, ymidm = PrevBBox[IdPrevC][0:6]
                    LewatBatas = (ymidc-MidLineY)*(ymidm-MidLineY)
                    if LewatBatas <= 0:
                        Arah = ymidc-MidLineY
                        if Arah >= 0:
                            self.rightCounter[clc] += 1
                            vehicle_count = self.rightCounter[clc] + self.leftCounter[clc]
                            vehicle_info, vehicle_type = self.printVehicleType(frame, clc, xminc, yminc,
xmaxc, ymaxc)
                            result_key_name = vehicle_type + "_" + str(vehicle_count)
                            self.update_count_gui()
```

```
                    # Number plate detection part
                    vehicle_image = vehicle_info["image"]
                    vehicle_image = cv2.resize(vehicle_image,(426,240),None, 0.5, 0.5)
                    self.vehicle_photo = PIL.ImageTk.PhotoImage(image =
PIL.Image.fromarray(vehicle_image))
                    self.canvas2.create_image(0, 0, image = self.vehicle_photo, anchor =
tkinter.NW)

                    self.detectedText = self.numberPlateDisplay(vehicle_info["image"])
                    displayedText = result_key_name + "       " + vehicle_type + "       " +
self.detectedText + "       " + vehicle_info["timestamp"]
                    self.listbox.insert(tkinter.END, displayedText)
                    self.listbox.config(yscrollcommand = self.scrollbar.set,
background="#F3ECB0", selectbackground="#ADE792", fg="#212121")
                    self.scrollbar.config(command = self.listbox.yview)
                    vehicle_info["vehicle_number"] = self.detectedText

                    self.result[result_key_name] = vehicle_info
                    self.number_plate.config(text=self.detectedText)

                else:
                    self.leftCounter[clc] += 1
                    vehicle_count = self.rightCounter[clc] + self.leftCounter[clc]
                    vehicle_info, vehicle_type = self.printVehicleType(frame, clc, xminc, yminc,
xmaxc, ymaxc)
                    result_key_name = vehicle_type + "_" + str(vehicle_count)
                    self.update_count_gui()
                    # Number plate detection part
                    vehicle_image = vehicle_info["image"]
                    vehicle_image = cv2.resize(vehicle_image,(426,240),None, 0.5, 0.5)
                    self.vehicle_photo = PIL.ImageTk.PhotoImage(image =
PIL.Image.fromarray(vehicle_image))
                    self.canvas2.create_image(0, 0, image = self.vehicle_photo, anchor =
tkinter.NW)
                    self.detectedText = self.numberPlateDisplay(vehicle_info["image"])
                    vehicle_info["vehicle_number"] = self.detectedText
                    displayedText = result_key_name + "       " + vehicle_type + "       " +
self.detectedText + "       " + vehicle_info["timestamp"]
                    self.listbox.insert(tkinter.END, displayedText)
                    self.listbox.config(yscrollcommand = self.scrollbar.set,
background="#F3ECB0", selectbackground="#ADE792", fg="#212121")

                    self.scrollbar.config(command = self.listbox.yview)
                    self.result[result_key_name] = vehicle_info
                    self.number_plate.config(text=self.detectedText)

    def numberPlateDisplay(self, vehicle_image):
        detectedText=""
```

```
        try:
            self.number_plate_image, self.gray_number_plate_image=
self.get_number_plate(vehicle_image)
            self.number_plate_image_display  =
cv2.resize(self.number_plate_image,(150,50),None, 0.5, 0.5)
            self.number_plate_image_display = PIL.ImageTk.PhotoImage(image =
PIL.Image.fromarray(self.number_plate_image_display))
            self.canvas3.create_image(0,0,anchor="nw",
image=self.number_plate_image_display)
            self.vehicle_number = self.get_text()
            detectedText=self.vehicle_number
        except:
            self.number_plate_image = cv2.imread('namePlateNotFound.png',
cv2.IMREAD_UNCHANGED)
            self.number_plate_image_display  =
cv2.resize(self.number_plate_image,(150,50),None, 0.5, 0.5)
            self.number_plate_image_display = PIL.ImageTk.PhotoImage(image =
PIL.Image.fromarray(self.number_plate_image_display))
            self.canvas3.create_image(0,0,anchor="nw",
image=self.number_plate_image_display)
            detectedText="Not Detected"
        return detectedText




    def DrawLastBoundingBox(self,frame):
        CurrentBBox = self.ListBBox[-1]
        for IndexLast in range(1, len(CurrentBBox)):
            xminc, yminc, xmaxc, ymaxc, xmidc, ymidc,  confc, clc, namac, IdPrevC =
CurrentBBox[
                IndexLast]
            pc1 = (int(xminc), int(yminc))
            pc2 = (int(xmaxc), int(ymaxc))
            pcc = (int(xmidc), int(ymidc))
            frame = cv2.rectangle(frame, pc1, pc2, (255, 0, 255), 1)
            frame = cv2.circle(frame, pcc, 2, (255, 255, 255), 1)

        return frame

    def DrawVehicleVector(self,frame):
        if len(self.ListBBox) >= 2:
            CurrentBBox = self.ListBBox[-1]
            PrevBBox = self.ListBBox[-2]
            for IndexLast in range(1, len(CurrentBBox)):
                xminc, yminc, xmaxc, ymaxc, xmidc, ymidc,  confc, clc, namac, IdPrevC =
CurrentBBox[
```

```
                IndexLast]

            if IdPrevC > -1:

                xminm, yminm, xmaxm, ymaxm, xmidm, ymidm = PrevBBox[IdPrevC][0:6]
                p1 = (int(xmidc), int(ymidc))
                p2 = (int(2*xmidc-xmidm), int(2*ymidc-ymidm))

        return frame

    def get_number_plate(self, frame):#vehicle_image
        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        bfilter = cv2.bilateralFilter(gray, 11, 17, 17) #Noise reduction
        edged = cv2.Canny(bfilter, 30, 200) #Edge detection
        keypoints = cv2.findContours(edged.copy(), cv2.RETR_TREE,
cv2.CHAIN_APPROX_SIMPLE)
        contours = imutils.grab_contours(keypoints)
        contours = sorted(contours, key=cv2.contourArea, reverse=True)[:10]
        location = None
        for contour in contours:
            approx = cv2.approxPolyDP(contour, 10, True)
            if len(approx) == 4:
                location = approx
                break
        mask = np.zeros(gray.shape, np.uint8)
        new_image = cv2.drawContours(mask, [location], 0,255, -1)
        new_image = cv2.bitwise_and(frame, frame, mask=mask)
        (x,y) = np.where(mask==255)
        (x1, y1) = (np.min(x), np.min(y))
        (x2, y2) = (np.max(x), np.max(y))
        cropped_image=gray[x1:x2+1, y1:y2+1]
        return cropped_image,gray

    def get_text(self):
        reader = easyocr.Reader(['en'], verbose=False)
        result = reader.readtext(self.number_plate_image)#registration_number
        return result[0][1]


    def displayStartImage(self):
        frame = cv2.imread('playImage.png', cv2.IMREAD_UNCHANGED)
        frame = cv2.resize(frame, (640, 480), interpolation = cv2.INTER_AREA)
        self.photo = PIL.ImageTk.PhotoImage(image = PIL.Image.fromarray(frame))
        self.canvas1.create_image(0, 0, image = self.photo, anchor = tkinter.NW)


    def playVideo(self):
        if self.load_file == True:
```

```
            if self.pause == True:
                self.pause = False
                self.update()
            else:
                self.update()
        return

    def pauseVideo(self):
        self.pause = True

    def stopVideo(self):
        self.databaseConfigure()
        self.initalizeVariables()
        self.displayStartImage()


    def update(self):
        if self.load_file:
            ret, old_frame = self.vid.get_frame()
            if ret:
                frame = old_frame
                b, c, w = frame.shape
                MidLineY = b - b*(self.yHeight)/100
                BBox = self.VehicleDetection(frame)
                self.ListBBox.append(BBox)
                if len(self.ListBBox)>2:
                    self.ListBBox.pop(0)
                self.ListBBox= self.VehicleTracking()
                self.VehicleCounting(frame,MidLineY)
                frame = self.DrawLastBoundingBox(frame)
                frame = self.DrawVehicleVector(frame)
                p1 = (0, int(MidLineY))
                p2 = (c, int(MidLineY))
                frame = cv2.line(frame, p1, p2, (0,255,0), 2)
                frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
                frame = cv2.resize(frame,(640,480),None,0.5,0.5)
                self.photo = PIL.ImageTk.PhotoImage(image = PIL.Image.fromarray(frame))
                self.canvas1.create_image(0, 0, image = self.photo, anchor = tkinter.NW)
            if not self.pause:
                self.window.after(self.delay, self.update)
        else:
            return


    def update_count_gui(self):
        self.car_label_text = Label(self.frame7, text=str(self.leftCounter[2] +
self.rightCounter[2]),bg="#344D67",fg="white",font=self.bText)
        self.car_label_text.grid(row=0,column=1)
```

```
        self.motor_label_text = Label(self.frame8, text=str(self.leftCounter[3] +
self.rightCounter[3]),bg="#344D67",fg="white",font=self.bText)
        self.motor_label_text.grid(row=0,column=1)
        self.bus_label_text = Label(self.frame8, text=str(self.leftCounter[5] +
self.rightCounter[5]),bg="#344D67",fg="white",font=self.bText)
        self.bus_label_text.grid(row=1,column=1)
        self.truck_label_text = Label(self.frame7, text=str(self.leftCounter[7] +
self.rightCounter[7]),bg="#344D67",fg="white",font=self.bText)
        self.truck_label_text.grid(row=1,column=1)


    def open_photo(self):
        file = askopenfile(mode='r', filetypes=[('Image Files', ["*.jpg"])])
        self.filename.set(file.name)
        self.img = PIL.Image.open("+ self.filename.get())
        self.img = self.img.resize((426,240), PIL.Image.Resampling.LANCZOS)
        self.img = PIL.ImageTk.PhotoImage(self.img)
        self.canvas2.create_image(0,0, anchor=tkinter.NW, image=self.img)

    def capture(self):
        new_image = cv2.imread(self.filename.get())
        self.detectedText = self.numberPlateDisplay(new_image)
        self.number_plate.config(text=self.detectedText)
        self.initalizeVariables()

    def close(self):
        self.databaseConfigure()
        self.window.destroy()

    def select_video(self):
        self.leftCounter = {2: 0, 3: 0, 5:0, 7: 0}
        self.rightCounter = {2: 0, 3: 0, 5:0, 7: 0}
        try:
            file = askopenfile(mode='r', filetypes=[("MP4 files", "*.mp4"),("WMV files", "*.wmv"),
("AVI files", "*.avi")])
            self.video_source = file.name
            self.vid = MyVideoCapture(self.video_source)
            self.load_file = True
        except:
            self.video_source = ''
            self.load_file = False



# In[17]:


App(tkinter.Tk(), "Heavy Vehicle Classification and Identification")
```

****