CAR DETECTION USING OPENCV

PROJECT REPORT

Submitted to:

Name: Dr. Altaf Q H Badar

Assisstant Prof

Dept of Electrical Engg

Submitted by:

P.Revanth

22EEB0A34

B.Tech 2nd year

Electrical Engg

ACADEMIC YEAR-2023-24

INTRODUCTION:

The presented Python code is designed for car detection in video files, leveraging computer vision techniques and a graphical user interface (GUI). The primary libraries employed in this project include OpenCV for image processing, Tkinter for GUI development, and Matplotlib for potential visualization (if necessary). The objective is to provide a user-friendly interface for initiating car detection in videos and, optionally, identifying the predominant colors of the detected cars.

CODE OVERVIEW:

Tkinter GUI:

The code begins by creating a Tkinter GUI with three buttons: "Detect Cars", "Detect Cars and Colors" and "Exit." These buttons facilitate the initiation of car detection with or without color analysis.

Car Detection:

The core of the application lies in the '*start_detection'* function, which utilizes OpenCV. It captures frames from a specified video file, converts them to grayscale, and employs a Haar cascade classifier (carx.xml) for car detection. Detected cars are outlined with rectangles, and an increasing count of total cars is displayed on the GUI.

Color Detection:

If the user selects the "Detect Cars and Colors" button, the '*detect_car_colors*' function is invoked for each detected car region. This function categorizes the dominant color in three vertical segments of the car image, utilizing a predefined dictionary of color categories.

FUNCTIONS:

detect_car_colors(car_image):

This function identifies the predominant color in a given car image by splitting it into three vertical segments. The color is determined using a predefined dictionary (color_dict) mapping RGB values to color names.

start_detection(video_path, detect_color=False):

The primary function for car detection in videos. It utilizes Haar cascades for car detection, updates the GUI with the count of detected cars, and optionally displays the colors of the cars if the detect_color parameter is set to True.

find_closest_color(color, color_dict):

A helper function that finds the closest predefined color category for a given RGB color. It calculates the Euclidean distance between the input color and predefined colors in color_dict.

gui:

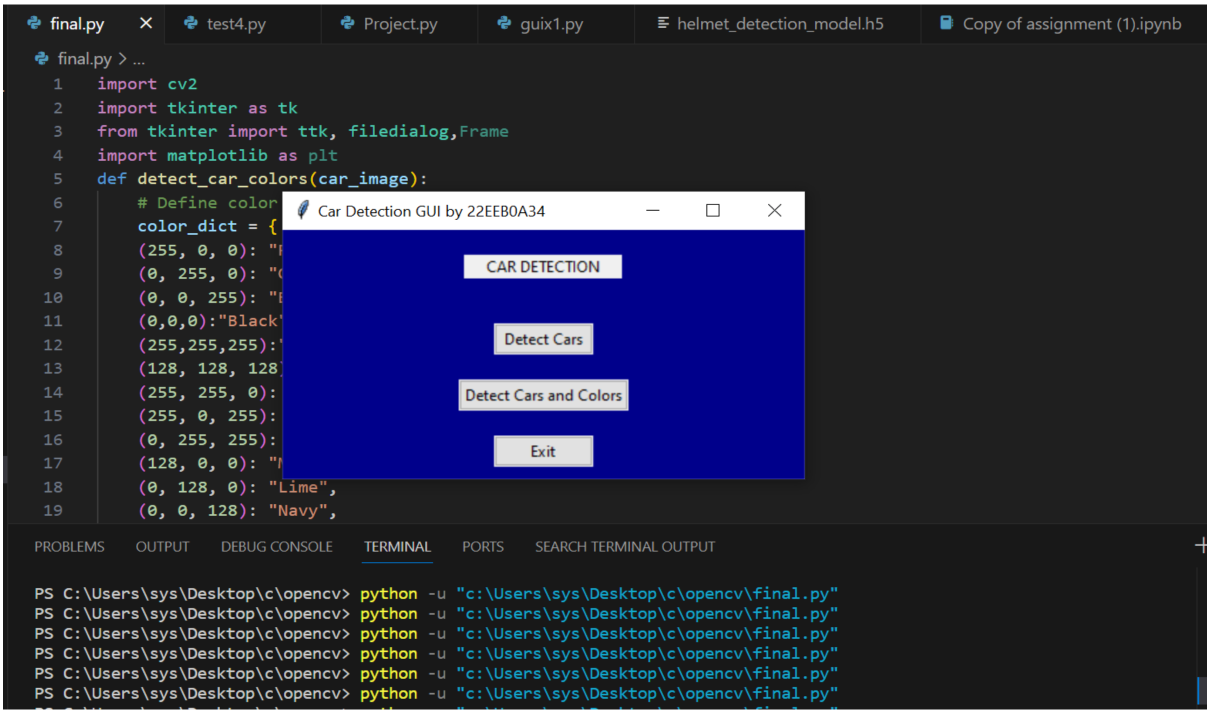
The GUI provides a simple and user-friendly way to interact with the car detection functionality. Users can choose a video file, start the car detection process, and optionally detect the colours of the detected cars. The "Exit" button allows users to close the GUI.

- **buttons**:

  start_button: A button with the label "Detect Cars." When clicked, it calls the on_start_button_click function with the argument detect_color=False.

  start_button_color: Another button with the label "Detect Cars and Colors." When clicked, it calls the on_start_button_click function with the argument detect_color=True.

exit_button: A button with the label "Exit." When clicked, it closes the Tkinter window by calling root.destroy().

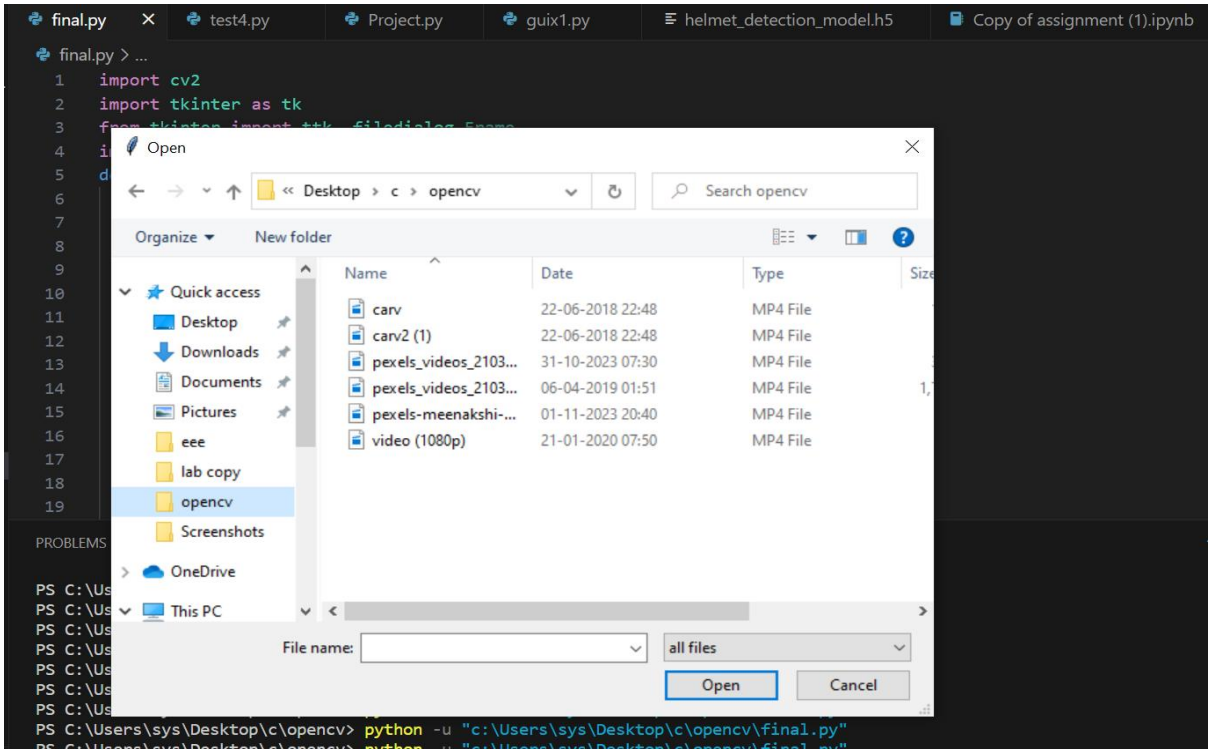


- **button actions**:

on_start_button_click(detect_color): This function is called when either of the "Detect Cars" or "Detect Cars and Colors" buttons is clicked. It opens a file dialog using filedialog.askopenfilename to allow the user to select a video file. If a file is selected, it calls the start_detection function with the selected video path and the detect_color argument.

- **file dialog**:

filedialog.askopenfilename(filetypes=[("all files", "*.mp4")]): Opens a file dialog that allows the user to choose a file with a .mp4 extension. This is used in the on_start_button_click function to get the path of the selected video file.



Haar cascade classifier:

A Haar Cascade Classifier is an object detection method that uses machine learning for identifying objects in images or video. Haar Cascade Classifiers are widely used for real-time object detection in applications like face detection, pedestrian detection, and other generic object detection tasks. OpenCV, a popular computer vision library, provides pre-trained Haar Cascade classifiers for various objects, making it easier for developers to implement object detection in their applications.

- The main tool for recognizing vehicles is the provided.xml file, which can read an image and identify the area where a car or other comparable shape is present.
- the Haar Cascade Classifier is applied to each frame of a video. The cv2.CascadeClassifier('carx.xml') line loads a pre-trained Haar Cascade classifier specifically designed for car detection. The car_cascade.detectMultiScale function is then used to identify regions in the image that likely contain cars.

The portion of frame which contains car image is divided into 3 segments based on pixels. In first segment it will recognize all colors using *'calcHist'* which is an inbuilt function in OpenCV module. The colors present in the specified region are detected and the maximum repeated color is stored in *'max_color'*.

```
color_dict = {
(255, 0, 0): "Red",
(0, 255, 0): "Green",
(0, 0, 255): "Blue",
(0,0,0):"Black",
(255,255,255):"White",
(128, 128, 128): "Gray",
(255, 255, 0): "Yellow",
(255, 0, 255): "Magenta",
(0, 255, 255): "Cyan",
(128, 0, 0): "Maroon",
(0, 128, 0): "Lime",
(0, 0, 128): "Navy",
(255, 128, 0): "Orange",
(128, 255, 0): "Lime Green",
(0, 128, 255): "Sky Blue",
(255, 0, 128): "Rose",
(128, 0, 128): "Purple",
(128, 255, 128): "Pale Green",
(0, 128, 128): "Teal",
(128, 128, 0): "Olive",
}
```

Max_color = (200, 50, 100)

color_dict = {

(255, 0, 0): 'Red',

(0, 255, 0): 'Green',

(0, 0, 255): 'Blue',

(255, 255, 0): 'Yellow',

(255, 0, 255): 'Magenta',

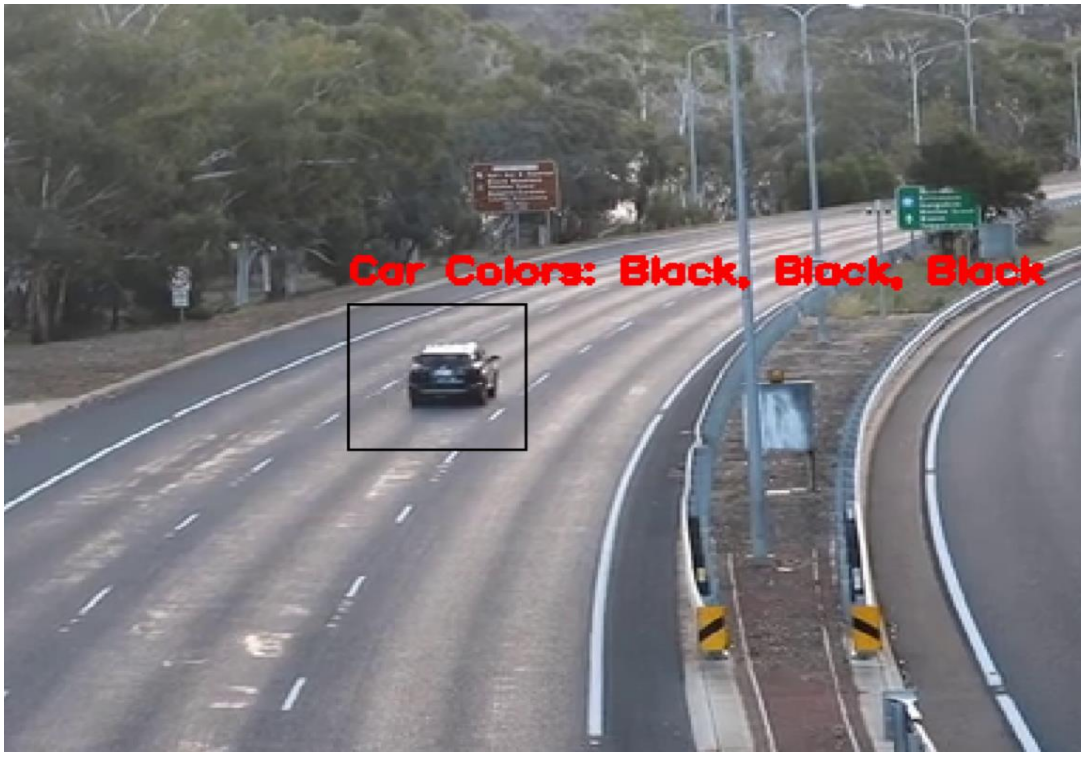
#.....

}

for 1st iteration:$((200-255)^2+(50-0)^2+(100-0)^2)$

This goes on for every color and the color which gives the minimum Euclidean distance is appended in car_colors

- *max_color* is a tuple containing the corresponding RGB intensities of the color with the highest count in the histogram.
- It is then passed to *'find_closest_color'*. In this function two parameters are used: *'min_distance', 'closest_color'*. 1st parameter stores the sum of squares of difference of intensities between *max_color* and the colours in the color_dict and stores the value of the color if the difference is minimum.
- This is done for each segment of the frame and the detected color is printed just above the frame.



In conclusion, the presented program offers a comprehensive solution for car detection in video files, providing users with a user-friendly Graphical User Interface (GUI) built using Tkinter. The integration of OpenCV's Haar Cascade Classifier facilitates efficient and accurate identification of cars within video frames. The flexibility

of the program is enhanced by the option to detect not only the presence of cars but also their dominant colours.

Key Features and Contributions:

1.Haar Cascade Classifier for Car Detection: The core of the program leverages the Haar Cascade Classifier, a powerful machine learning technique for object detection. This ensures reliable identification of cars in various scenarios within video files.

2.Color Detection: The program goes beyond mere car detection by incorporating color identification. Utilizing predefined color categories, the system can provide insights into the dominant colors of the detected cars. This feature adds an additional layer of information, which can be valuable in certain applications such as traffic analysis or vehicle monitoring.

3.User-Friendly GUI: The Tkinter-based GUI offers an intuitive interface for users to interact with the program. The inclusion of buttons such as "Detect Cars" and "Detect Cars and Colors" simplifies the initiation of the detection process, while the "Exit" button allows for the seamless closure of the application.

4.Dynamic Video Processing: The program dynamically processes video frames, displaying real-time information about the total number of detected cars. The inclusion of a pause functionality allows users to control the processing speed, enhancing the program's usability.