Ahmedabad University

PROJECT REPORT

Title: Recognition Of Vehicle License Plate Using Image Processing

Submitted to: Prof. Ashok Ranade

Group-28 | ECE210 Signals and Systems

Contents

Synopsis

ANPR(Automatic Number Plate Recognition) system is an identification and verification system of license plates in vehicles, using image processing and optical character recognition. This system is used on a much larger scale in various areas like traffic control, parking control and tolling on highways. It is especially useful for enforcement of traffic rules and generating fine receipts online.

This project intends to demonstrate the functioning of the ANPR system on a small scale. It involves a simple graphical user interface created using the Tkinter library in python, image processing by means of OpenCV library in python, and data management through sqlite3.

The system accepts the name of the image stored in the computer, as entered by the user. The image will go through various stages of processing like plate detection and text recognition. The system will be able to identify the unique registration number of the given vehicle.

The program will then pass a query through the database containing information about all the vehicles and their respective owners, scanning for the registration number identified. When the owner is identified, his/her details like residential address, phone number etc. will be displayed on the screen.

Processing Techniques

The proposed system focuses upon the image processing techniques in which captured images of vehicle license plates are processed through multiple algorithms to convert the alpha numeric conversion of image into text format and displayed. Image processing steps include major machine learning algorithms like KNN algorithms . It also includes some of the important techniques like character segmentation and character recognition. Image processing in Number Plate Recognition(NPR) system spectral analysis approach is used for acquiring the image, to extract the region of interest , character segmentation using Support Vector Machine(SVM) features techniques and algorithms. Due to the colour of the number plate characters and the background of the number plate, it is difficult to detect the boundary of the number plate from the input car images in the outdoor scene so the gradients of the original image is adopted to detect the candidate number plate regions. Algorithms which are a combination of morphological operation, segmentation and canny edge detector are also used. License plate location algorithms consist of steps like Edge Detection, Morphological operation like dilation and erosion, Smoothing, segmentation of characters and recognition of plate character.Image is pre-processed through different techniques and algorithms and location of the number plate is extracted. Major steps are :

1. RGB(RED GREEN BLUE) To Gray Scale Conversion

Grayscaling is the method of transforming an image to grey shades from other colour spaces, such as RGB, CMYK, HSV, etc. It varies between full white and full black. Python RGB to GrayScale can be done by using functions of the OPENCV module . "`cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)`" function can be used to convert RGB image to GrayScale image.

Fig-1.1 - RGB image(Original Image)

Fig-1.2-Gray Scale image

2. Median Filtering

When images are acquired, a lot of noises are correlated with that picture. It's difficult to remove the noise with Gray processing. So to remove noise from the image "*Median Filters*" are used to make the image free from noises. This is a mandatory step because in License Plate recognition it directly affects the accuracy of the recognition rate of the system.

3. Edge Detection and dilation of BGM (Binary Gradient Masking)

The goal of *Edge Detection* is to dramatically decrease the quantity of data of an image and it maintains the structural properties for more image analysis. Edge Detection locates the sharp discontinuities in an image. This is the most common technique to detect the meaningful discontinuities in intensity values.

Fig 3.1- Image after dilation of BGM and Edge Detection

4. Plate Region Extraction

The fourth step of an algorithm is the extraction of a license plate from an image. We can do it by finding row and column values of that image followed by modifying the image by using `r/3:r` and saving the objects. A dilation is the morphological closing operation followed by an erosion operation in which same structuring is used for both the operations.

Fig-4.1-Image of plate after plage region extraction

5. Morphological Operations

Morphology is a large range of image processing operations that are focused on shapes.Morphological tasks apply an organizing component to an input picture, making an yield picture of a similar size. The most fundamental morphological tasks are dilation and erosion.*Dilation* is performed by adding pixels to the boundaries of objects for all the pixels in the input pixel's area. *Erosion* is just opposite to dilation. The idea of erosion is that it always tries to keep foreground white. The boundaries of the foreground object are eroded by this. In erosion all the pixels near boundaries are discarded according to the size of the kernel which results in decreasing of thickness or the size of the foreground object i.e. white region decreases.

"`erosion = cv2.erode(img,kernel,iterations= 1)`" is the command for erosion `while "dilation=cv2.dilate(img,kernel,iterations =1)`" is the command for dilation. The morphological open operation is an erosion operation followed by a dilation operation for the same structuring element. While morphological close operation is a dilation operation followed by an erosion operation for the same structuring element.

6. Character Segmentation

Character segmentation is an operation that attempts to decompose an image into sub images of individual symbols of a series of characters.In an Optical Character Recognition System, it is one of the decision processes (OCR).The main goal of this process is to segment all characters without losing any features of the characters of an image. In this step the individual characters need to be segmented from each other.Segmentation is one of the most important processes in the number plate recognition, because all further steps rely on it. It partitions the image into some characterised parts so that each part contains one character and can be extracted for further processing. Image Segmentation can be applied in two ways: *Contour based segmentation or Region based segmentation*.Here in this stimulation we have applied *Contour based segmentation* in which is based on gradient based segmentation method. It attempts to find the edges and boundaries directly depending upon their high gradient magnitudes. Character segmentation technique uses *Bounding Box technique.*

Bounding Box technique is used to measure properties of an image region. Once a bounding box is created over each character and number presented on a number plate , each character and number is extracted out for recognition of the number plate.

Fig-6.1- Bounding Box technique for character segmentation.

7. Character Recognition

Character Recognition means the mechanical or electronic translation into machine-editable text of images of typed, hand-written, or printed text. The programme for character recognition processes these scans of character segmented images to distinguish between images in the light and dark zone text and decide what letters are represented. So each character of the license plate is recognised using Character recognition technique.

Number plate recognition is now used to compare each individual character against the complete alphanumeric database using template matching. After successful and accurate template matching , a text file of the output image is displayed.

Fig-7.1&7.2-Data of Characters is used for template matching with processed image in imgThresh

K-NEAREST NEIGHBOR (KNN)-ALGORITHM

K-Nearest Neighbour is one of the simplest algorithms which works on supervised learning processing techniques for machine learning. It is a very simple and easy-to-implement machine learning algorithm that can be used to solve both classification and regression problems.At the training point, the KNN algorithm only stores the dataset and then classifies the data into a group that is very close to the new data. The steps to implement KNN algorithm is as follows:

Step 1:Choose the number K of the neighbors from an image.

Step 2:Calculate the Euclidean distance of the number of neighbours of K.

Step 3:As per the calculated Euclidean distance, take the K nearest neighbours.

Step 4:Count the number of data points between these K neighbours.

Step 5:Assign new data points to the group for which the neighbor's maximum number is calculated.

In python it is easy to implement the KNN algorithm by using inbuilt functions in modules ."`fileName.loadKNNDataTrainKNN()`"is the function implemented in simulation for KNN Algorithm.

```
blnKNNTrainingSuccessful = DetectChars.loadKNNDataAndTrainKNN()         # attempt KNN training
if blnKNNTrainingSuccessful == False:                                   # if KNN training was not successful
    print("\nerror: KNN traning was not successful\n")                  # show error message
    return                                                              # and exit program
# end if
```

Fig-8.1-The above image is the screenshot of the Applied KNN algorithm in the program.

Flowcharts

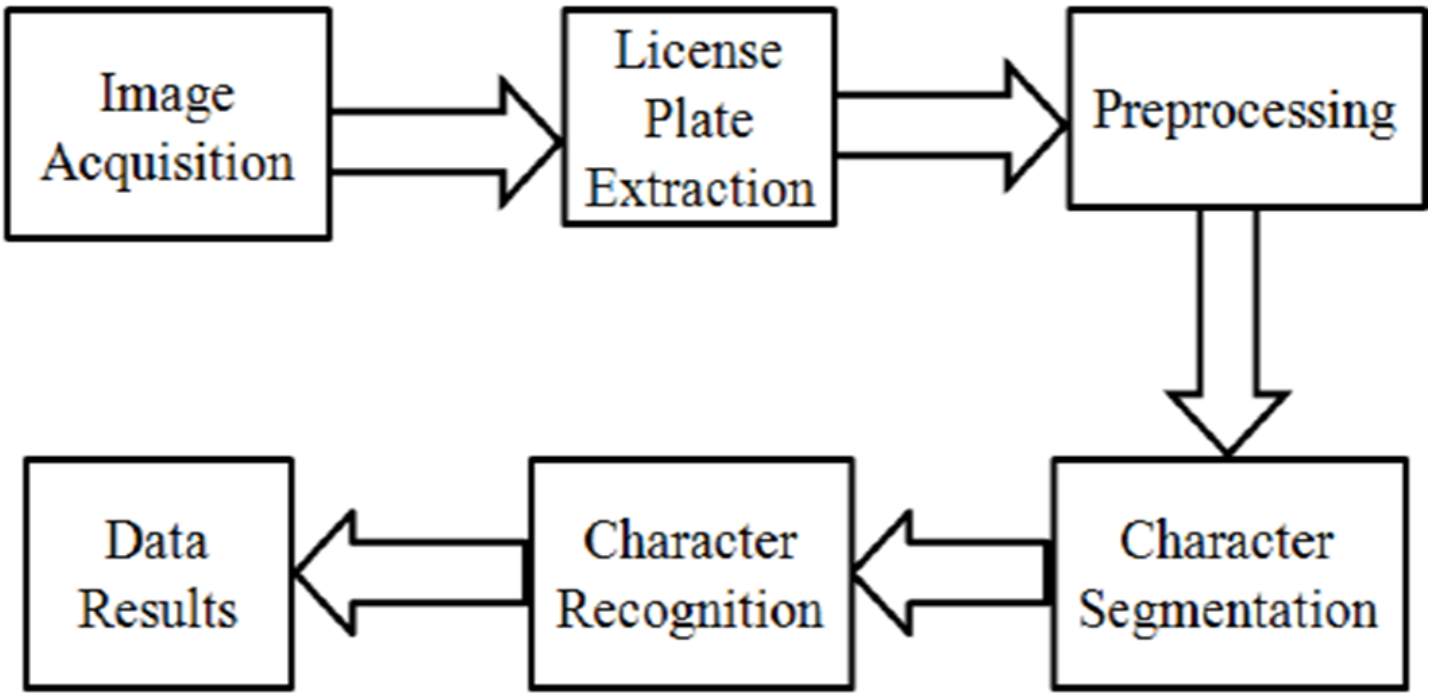

Fig-9.1: Flow chart of the program which includes major steps of the algorithm.

Steps

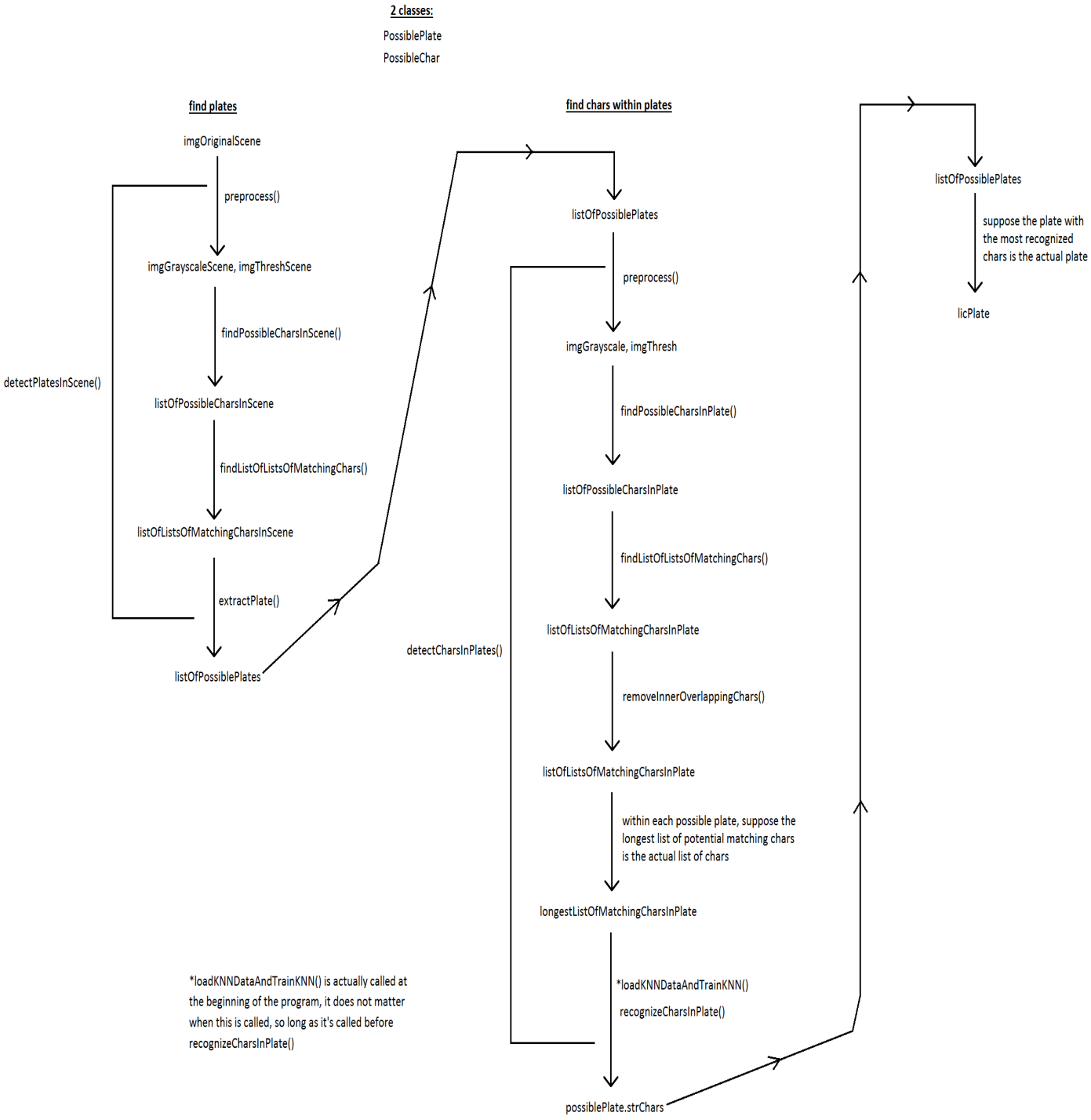

Fig-9.2: Detailed Flowchart of the Algorithm which describes all the processing methods and methodologies of the program of each step.

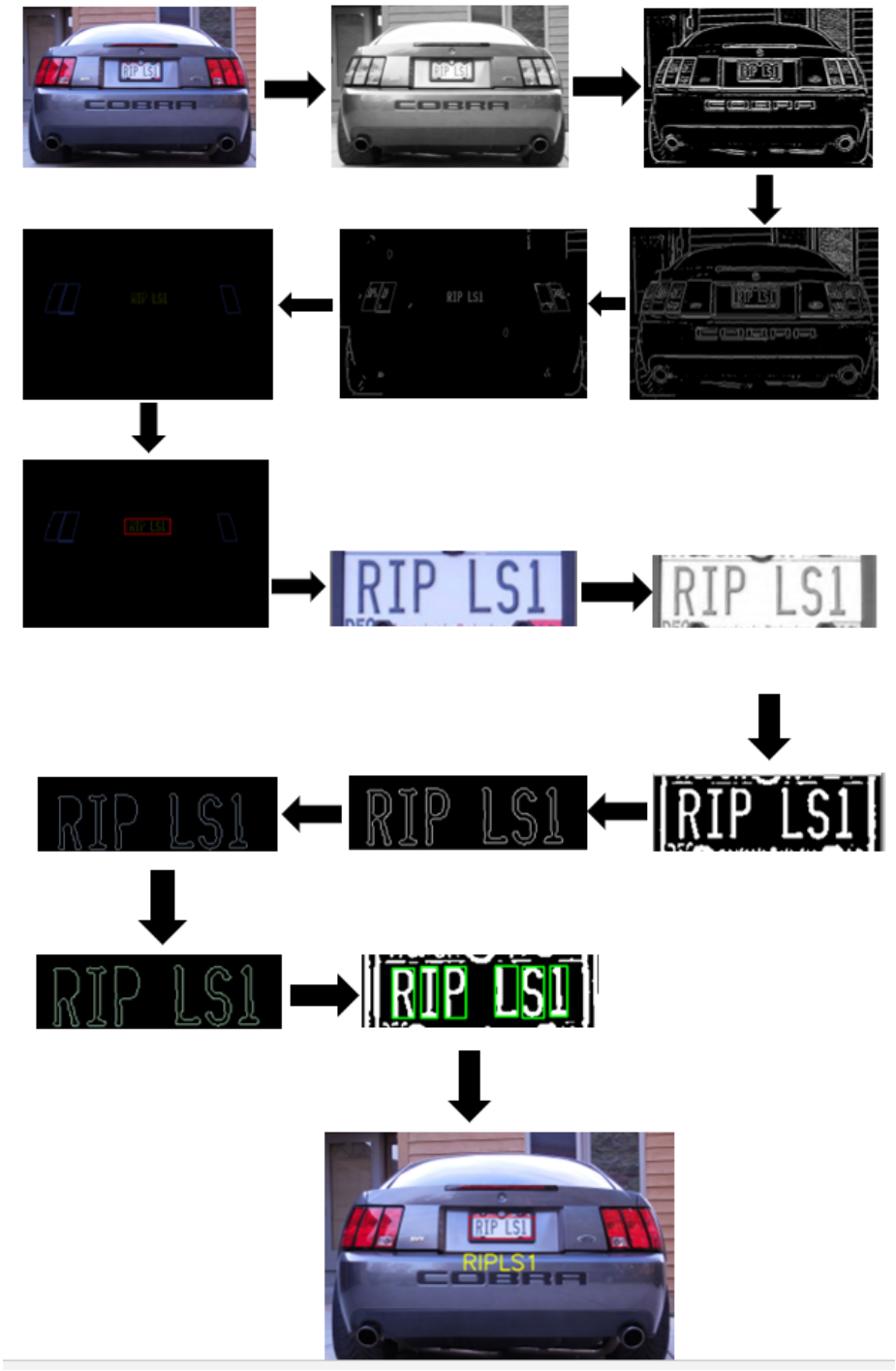

Fig-9.3: Detailed flowchart of Algorithm(Fig:9.2) with images processed at each step.

User-Interface

In order to create the user-interface, *Tkinter* library in python has been used.The standard Python interface to the Tk GUI toolkit is the Tkinter package ('Tk interface'). On most Unix platforms, as well as on Windows systems, both Tk and Tkinter are available.

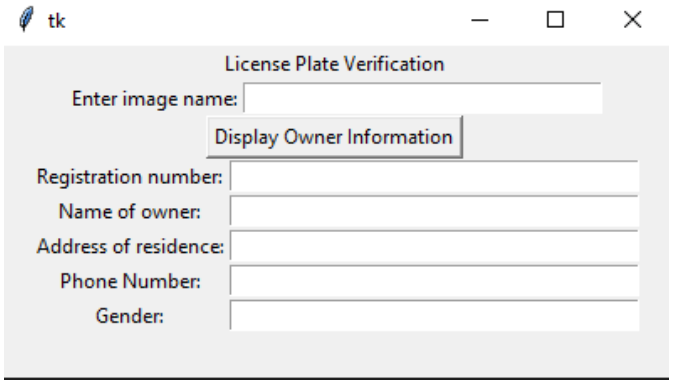

Fig-10.1-User Interface (General window when user runs the program)

The user interface includes a simple window that asks the user to enter the name of the image that corresponds to the license plate being verified.

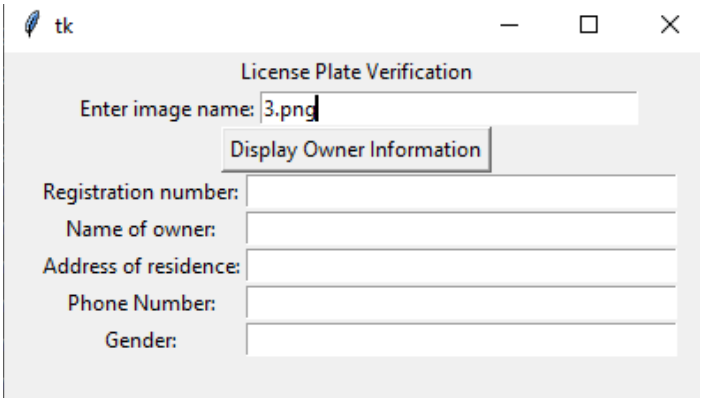

Fig-10.2- User can enter the name of an image in Textbox besides "Enter image name:"

Upon clicking the button “Display Owner Information”, the image will be processed to determine the registration number and other details of the owner that will be displayed.

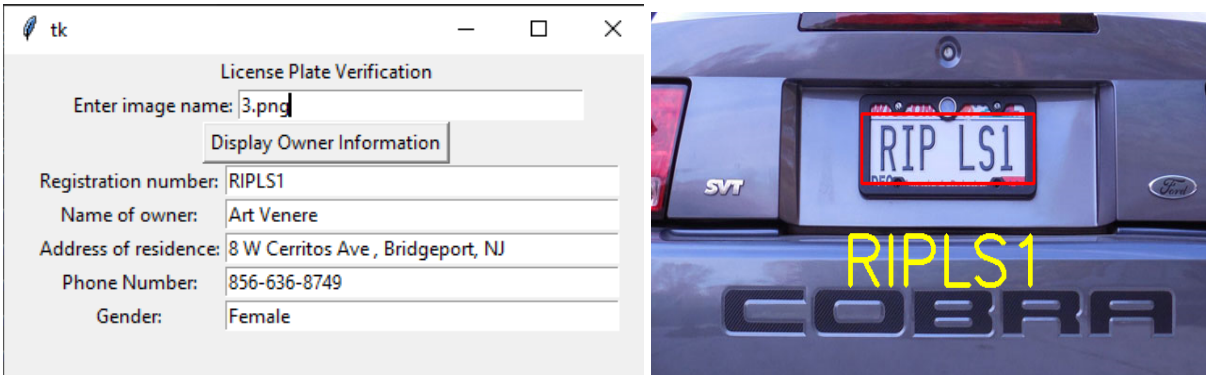

Fig-10.3- When the user clicks the “Display Owner Information” the above information of vehicle owner and image is displayed as the final output.

Limitations And Errors

Automatic licence plate recognition systems still have various problems. As we know that it is difficult to create a number plate recognition system with an accuracy of 100% so there are some limitations and errors in our system also but however we have managed to create a system with an accuracy of about 83%. There is a limitation in the step of character recognition as sometimes it recognises the letter incorrectly and this mostly happens with the characters which are having the same characteristics or with the characters which are difficult to recognise. When the user inputs the name of an image in the window then after all the processing steps for some image it displays the wrong characters in the output window and also in the final image which shows the detected license plate. This kind of error also occurs when the image is blurred or the image is not segmented correctly. Due to this error system does not give the relevant details of the owner in the output window. Following images show the limitation of the program as it recognises one character incorrectly.

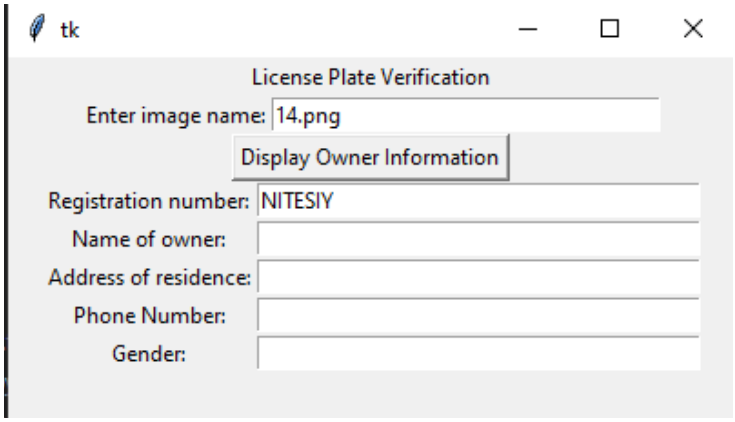

Fig-11.1-The first image is the original image named "14.png" and the second image is the User interface in which "14.png" image is taken as an input. As we can see that the information of the owner is not displayed because the system has recognized the license plate incorrectly . System has recognized "NITESIY" whereas original number plate is "NITESKY" so the system has made error in recognizing the character "K" instead system has recognised it as character "I".

Fig-11.2: The above two images shows that the system is making an error in "CHARACTER RECOGNITION STEP" as the characters in other step are processed correctly but at the last step of character recognition it makes an error in recognising the character "K" of the license plate.

Program

Main.py | DetectChars.py | DetectPlates.py | PossibleChar.py | PossiblePlate.py | Preprocess.py

Main.py > main

```python
import cv2
import numpy as np
import os
import tkinter as tk
import sqlite3

import DetectChars
import DetectPlates
import PossiblePlate



# module level variables ##########################################################################
SCALAR_BLACK = (0.0, 0.0, 0.0)
SCALAR_WHITE = (255.0, 255.0, 255.0)
SCALAR_YELLOW = (0.0, 255.0, 255.0)
SCALAR_GREEN = (0.0, 255.0, 0.0)
SCALAR_RED = (0.0, 0.0, 255.0)

showSteps = False
```

Fig-12.1: Key libraries are imported and module level variables are set.

```python
def main():

    window=tk.Tk()
    window.geometry("400x200")
    labelTitle=tk.Label(text="License Plate Verification")
    labelTitle.pack()
    frame1=tk.Frame()
    tk.Label(frame1,text="Enter image name:").grid(row=0, column=0)
    entry1=tk.Entry(frame1, width=35)
    entry1.grid(row=0, column=1)
    frame1.pack()

    frame2=tk.Frame()
    tk.Label(frame2,text="Registration number:").grid(row=0,column=0)
    textReg=tk.Entry(frame2, width=40)
    textReg.grid(row=0,column=1)
    tk.Label(frame2,text="Name of owner:").grid(row=1,column=0)
    textName=tk.Entry(frame2, width=40)
    textName.grid(row=1,column=1)
    tk.Label(frame2,text="Address of residence:").grid(row=2,column=0)
    textAddress=tk.Entry(frame2, width=40)
    textAddress.grid(row=2,column=1)
    tk.Label(frame2,text="Phone Number:").grid(row=3,column=0)
    textPhone=tk.Entry(frame2, width=40)
    textPhone.grid(row=3,column=1)
    tk.Label(frame2,text="Gender:").grid(row=4,column=0)
    textGender=tk.Entry(frame2, width=40)
    textGender.grid(row=4,column=1)
```

Fig-12.2: Main function begins and tkinter window is established.

```
51
52      def disp():
53          imageName=entry1.get()
54          textReg.delete(0,30)
55          textName.delete(0,30)
56          textAddress.delete(0,30)
57          textGender.delete(0,30)
58          textPhone.delete(0,30)
59
60          blnKNNTrainingSuccessful = DetectChars.loadKNNDataAndTrainKNN()          # attempt KNN training
61          if blnKNNTrainingSuccessful == False:                                    # if KNN training was not successful
62              print("\nerror: KNN traning was not successful\n")                   # show error message
63              return                                                               # and exit program
64          # end if
65
66          #imageName = input("ENTER THE NAME OF IMAGE IN JPEG/PNG FORMAT:")
67
68
69          imgOriginalScene  = cv2.imread(imageName)              # open image
70
71          if imgOriginalScene is None:                            # if image was not read successfully
72              print("\nerror: image not read from file \n\n")  # print error message to std out
73              os.system("pause")                                  # pause so user can see error message
74              return                                              # and exit program
75          # end if
76
77          listOfPossiblePlates = DetectPlates.detectPlatesInScene(imgOriginalScene)           # detect plates
78
79          listOfPossiblePlates = DetectChars.detectCharsInPlates(listOfPossiblePlates)        # detect chars in plates
```

Fig-12.3: disp() function within main() is created and KNN model is trained.

```
80
81          cv2.imshow("imgOriginalScene", imgOriginalScene)           # show scene image
82
83          if len(listOfPossiblePlates) == 0:                         # if no plates were found
84              print("\nno license plates were detected\n")  # inform user no plates were found
85          else:                                                      # else
86                  # if we get in here list of possible plates has at leat one plate
87
88                  # sort the list of possible plates in DESCENDING order (most number of chars to least number of chars)
89              listOfPossiblePlates.sort(key = lambda possiblePlate: len(possiblePlate.strChars), reverse = True)
90
91                  # suppose the plate with the most recognized chars (the first plate in sorted by string length descendi
92              licPlate = listOfPossiblePlates[0]
93
94              cv2.imshow("imgPlate", licPlate.imgPlate)          # show crop of plate and threshold of plate
95              cv2.imshow("imgThresh", licPlate.imgThresh)
96
97              if len(licPlate.strChars) == 0:                    # if no chars were found in the plate
98                  print("\nno characters were detected\n\n")
99                  textEntry.insert(0, "Invalid image/Number plate not found.")          # show message
100                 return                                         # and exit program
101             # end if
```

Fig-12.4: Possible plates are detected and Registration number is identified.

```
            drawRedRectangleAroundPlate(imgOriginalScene, licPlate)            # draw red rectangle around plate

            print("\nlicense plate read from image = " + licPlate.strChars + "\n")  # write license plate text to std out
            print("----------------------------------------")

            conn=sqlite3.connect('vehicleData.db')
            textReg.insert(0, licPlate.strChars)
            cursor=conn.execute("SELECT*FROM vehicleOwner where RegNo is ?",(licPlate.strChars,))

            for row in cursor:
                textName.insert(0, row[1])
                textAddress.insert(0, row[2])
                textPhone.insert(0,row[4])
                textGender.insert(0, row[3])

            writeLicensePlateCharsOnImage(imgOriginalScene, licPlate)           # write license plate text on the image

            cv2.imshow("imgOriginalScene", imgOriginalScene)                # re-show scene image

            cv2.imwrite("imgOriginalScene.png", imgOriginalScene)           # write image out to file

        # end if else

        cv2.waitKey(0)                 # hold windows open until user presses a key

    button1=tk.Button(text="Display Owner Information",command=disp)
    button1.pack()
    frame2.pack()
    window.mainloop()
    return
# end main
```

Fig-12.5: Registration number is displayed in the user window along with other owner details retrieved from the database through 'SELECT' query.

Conclusion

The license plate recognition systems are widely in use across the world. The procedure is difficult to implement due to numerous phases involved in image processing. Furthermore, accuracy of the system depends on factors like quality of image, amount of light in the image, blur, and variation in fonts of the characters. These factors can be taken care of by implementing the model more efficiently. As a result there is still a lot of scope for improvement in the technology. As this technology with 100% accuracy cannot be created because of some steps of Optical Character Recognition (OCR) like character recognition and character segmentation always have some kind of limitations.Though we tried to make a license plate recognition system using image processing with an accuracy of 83-84%.And also this created system is designed in user-friendly way . User has to just type the image name and in result it will display all the relevant information of the vehicle owner.This system can be Implemented in highly restricted areas such as military zones, Parliament, Supreme Court or area around top government offices or in parking areas of offices and shopping malls for vehicle authorisation and identification.

References

1. R. J. D. l. H. H. A. Hegt and N. A. Khan. A high performance license plate recognition system.in Proc. IEEE Int. Conf. System, Man, and Cybernetics, pages 4357{4362, 1998.
2. Nagare A. P. (2011), License Plate Character Recognition System using Neural Network, International Journal of Computer Applications, vol. 25.
3. Kapadia P. S. (2010), Car License Plate Recognition using Template Matching Algorithm, Sacramento, California: California State University.
4. Chang, S. L., Chen, L. S., Chung, Y. C., & Chen, S. W. (2004). Automatic license plate recognition. *IEEE transactions on intelligent transportation systems*, *5*(1), 42-53.