CHAPTER 1

INTRODUCTION

No-Parking Vehicle Detection (NPVD) system has been a practical technique in the past decades. One type of intelligent transportation system (ITS) technology is the automatic number plate recognition which can distinguish each vehicle as unique by recognizing the characters of the number plates. Automatic number plate recognition system finds wide varieties of applications to fit itself beyond just controlling access to collect details of vehicle parked in no parking areas. In NPVD, a camera captures the vehicle images and a computer processes them and recognizes the information on the number plate by applying various image processing and optical character recognition techniques. Prior to the character recognition, the number plates must be separated from the background vehicle images. This task is considered as the most crucial step in the ANPR system, which influences the overall accuracy and processing speed of the whole system significantly. Since there are problems such as poor image quality, image perspective distortion, other disturbance characters or reflection on vehicle surface, and the color similarity between the number plate and the background vehicle body, the number plate is often difficult to be located accurately and efficiently.

Generally vehicle number plate recognition is divided into several steps including number plate extraction, image region which contains a number plate, character segmentation, and character recognition. Generally, in order to recognize a vehicle number plate, the region of the number plate should be extracted from a vehicle image. Accurate detection of the plate region is essential process to go over to the step of character recognition.

There are two major methods to extract number plate region,

a) Edge Detection
b) Finding Rectangles in a Vehicle Image

1.1 STATEMENT OF THE PROBLEM

With decreasing costs of high quality surveillance systems, human activity detection and tracking has become increasingly practical. Accordingly, automated systems have been designed for numerous detection tasks, but the task of detecting illegally parked vehicles has been left largely to the human operators of surveillance systems. We propose NPVD for detecting this event in realtime by applying a novel image processing system that can perform the job quite easily and efficiently. After event detection, we extract the number plate (or otherwise called license plate) data for further processing. The proposed program is able to successfully recognize illegally parked vehicles in real-time and impose fine as per the traffic law in india.

CHAPTER 2

SYSTEM ANALYSIS

2.1 Existing system

The problem that countries like India that are still trying to deal with is the uncontrollable traffic rule breaking. Most of the problem fails to find a solution beacause of slow and manual actions that authorities takes when someone breaks the rule. Vehicles that are parked in no parking area often didn’t get any legal actions due to reasons like unability to watchout for local no parking areas and unavailability of that much of human working force. This results in many conditions like traffic jam, public nuisance etc. Thus the traditional present system in inefficient and non reliable.

2.2 Limitation of present system

a) Manual process
b) Time consuming
c) Events may get unreported while authorities are abscent at the time of it’s occurrence
d) More human effforts needed

2.3 Proposed system

NPVD system is proposed for monitoring and imposing fine to vehicles that are parked in non-parking area via identifying vehicle license plate numbers. No additional equipment need to be installed for operating this system. The only requirement of this system is installing special cameras for identifying license numbers on the no-parking area. The images taken by these cameras are subsequently processed in a computer. The cameras used in the system can be deployed under all weather conditions and are equipped with powerful infrared radiation units for identifying vehicle license plates in absolute darkness. The system normally comprises a camera for monitoring the vehicle path, an identification system for recognizing license plate number and are used for further identification of the corresponding owner to impose fines for breaking the traffic rules.

The software program used within the system deliver high precision and provide great processing speeds and fully reliable system. The OpenCV library provide a great image processing engine that ensures powerful and precise processing capabilities. The system prepares reports of vehicle that are parked in terms of time of entry and departure, vehicle license number, duration of each vehicle's stay in the area , amount of fine imposed etc. The user interface of the system is designed for speedy access to system events

2.4 Advantages and features of proposed system

a) The system can detect vehicle immediately
b) Less human intervention needed
c) .Fast, reliable and secure
d) Less paper work needed

2.5 Feasibility Study

A feasibility study is an analysis of how successfully a project can be completed, accounting for factors that affect it such as technical, economic, behavioural, operational factors. When a new project is proposed, it normally goes through the feasibility assessment. Feasibility Study is carried out to determine whether the proposed system is possible to develop with available resources & what should be the cost of consideration.

Various types of feasibilities are,

a) Technical Feasibility
b) Economic Feasibility
c) Operational Feasibility

If the proposed system is not feasible to develop, it is rejected at this very step.

2.5 .1 Technical Feasibility

The proposed system uses the language Python. Based on this criteria, we can strongly say that it is technically feasible, since there will not be much difficulty in getting required resources for the development & maintaining system as well. All the resources needed for the development of the software as well as the maintenance of the same is available in the organization. Here we are utilizing the resources which are already available so it’s very well technically feasible that we can implement flood detection system.

2.5 .2 Economic Feasibility

It is found that the benefit from our system would be more than the cost and time involved in its development. In our system the implementation cost over production is economically feasible. Economic analysis is the most frequently used techniques for evaluating the effectiveness of the proposed system more commonly known as cost/benefit analysis the procedure is to determine the benefits and savings that are expected from a proposed system and compare them with costs.

2.5 .3 Operational Feasibility

The proposed system satisfies operational feasibility in the way that the customers needs are satisfied. The system is adaptable to the customers and acceptable to the common people who use this. Operational feasibility assesses the extent to which the required software performs a series of steps to solve business problems and user requirements. This feasibility is dependent on human resources (software development team) and involves visualizing whether the software will operate after it is developed and be operative once it is installed. Operational feasibility also performs the following tasks:

a) Determines whether the problems anticipated in user requirements are of high priority
b) Determines whether the solution suggested by the software development team is acceptable
c) Analyses whether users will adapt to a new software
d) whether the organization is satisfied by the alternative solutions proposed by the software development team.

CHAPTER 3

SYSTEM SPECIFICATION

3.1 Minimum Software Requirements

1. Operating System	: Windows 8/10
2. Language	: Python
4. IDE	: Spyder3
5. Libraries	: Open CV, Numpy, tKinter

3.2 Minimum Hardware Requirements

1. Processor	: Intel i5 or AMD Ryzen 3
2. RAM	: 3 GB
3. Hard Disk Drive	: 200 GB
4. Peripherals	: Keyboard, Mouse, Monitor, Camera

CHAPTER 4

SYSTEM DESIGN

4.1 Context Level Diagram

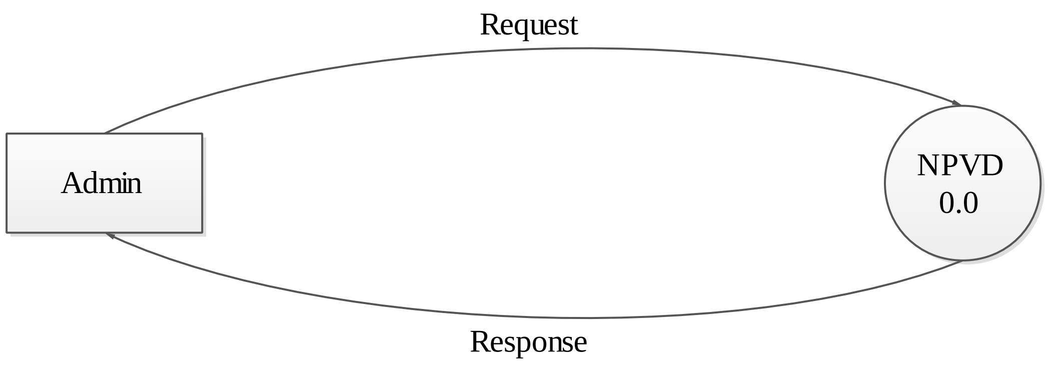


Fig 4.1 Context Level Diagram

4.2 Data Flow Diagram

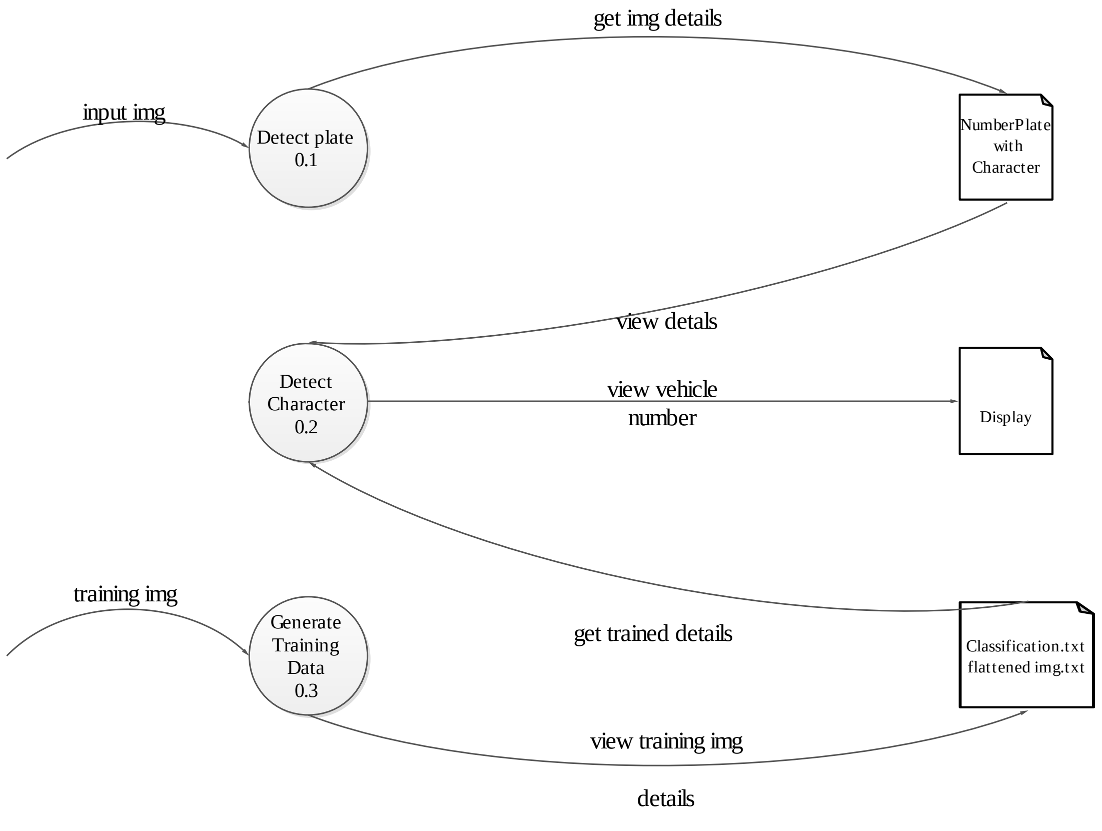


Fig 4.2.1 Level 1 DFD of NPVD

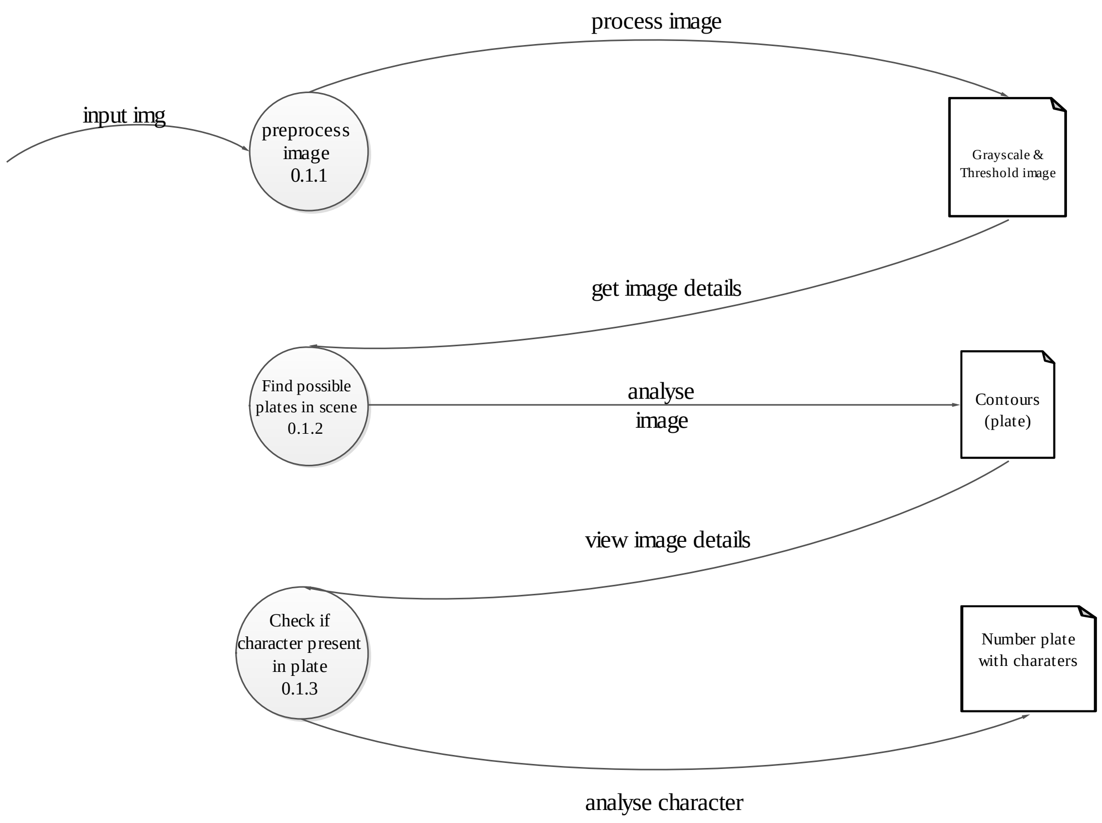

process image
input img
preprocess image 0.1.1
Grayscale & Threshold image
get image details
Find possible plates in scene 0.1.2
analyse image
Contours (plate)
view image details
Check if character present in plate 0.1.3
Number plate with charaters
analyse character

Fig 4.2.2 Level 2 DFD of Detect Plate

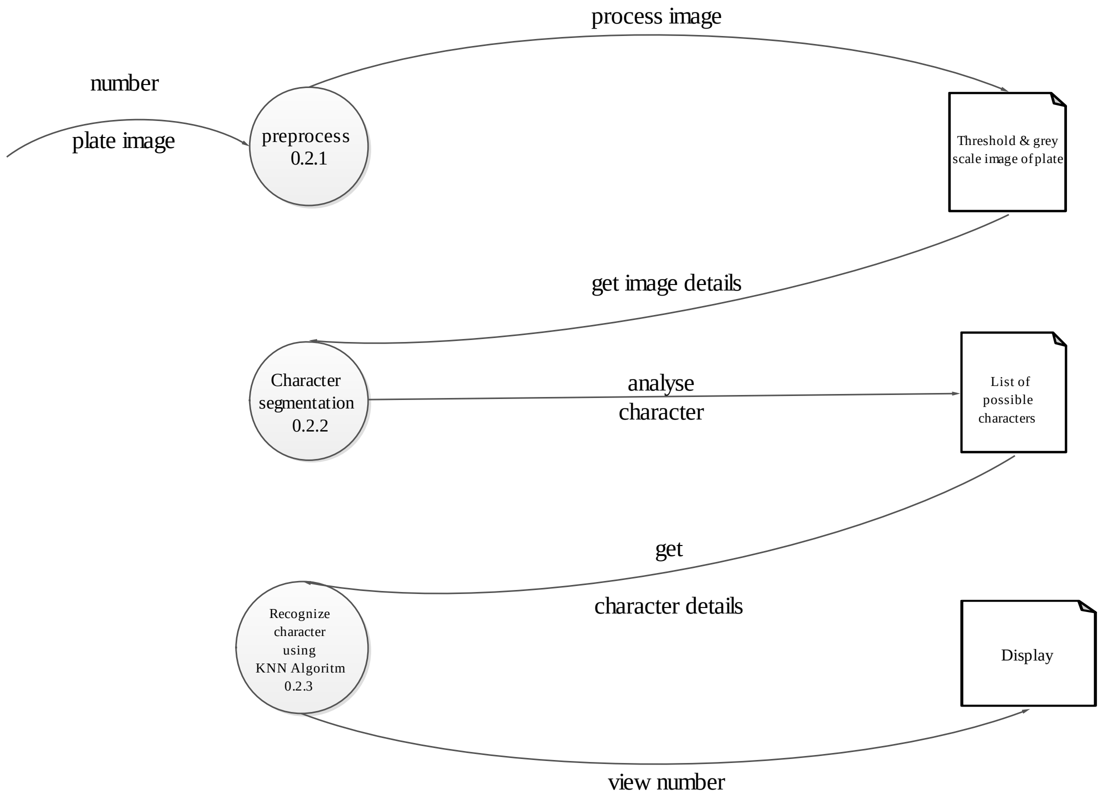


Fig 4.2.3 Level 2 DFD of Detect Character

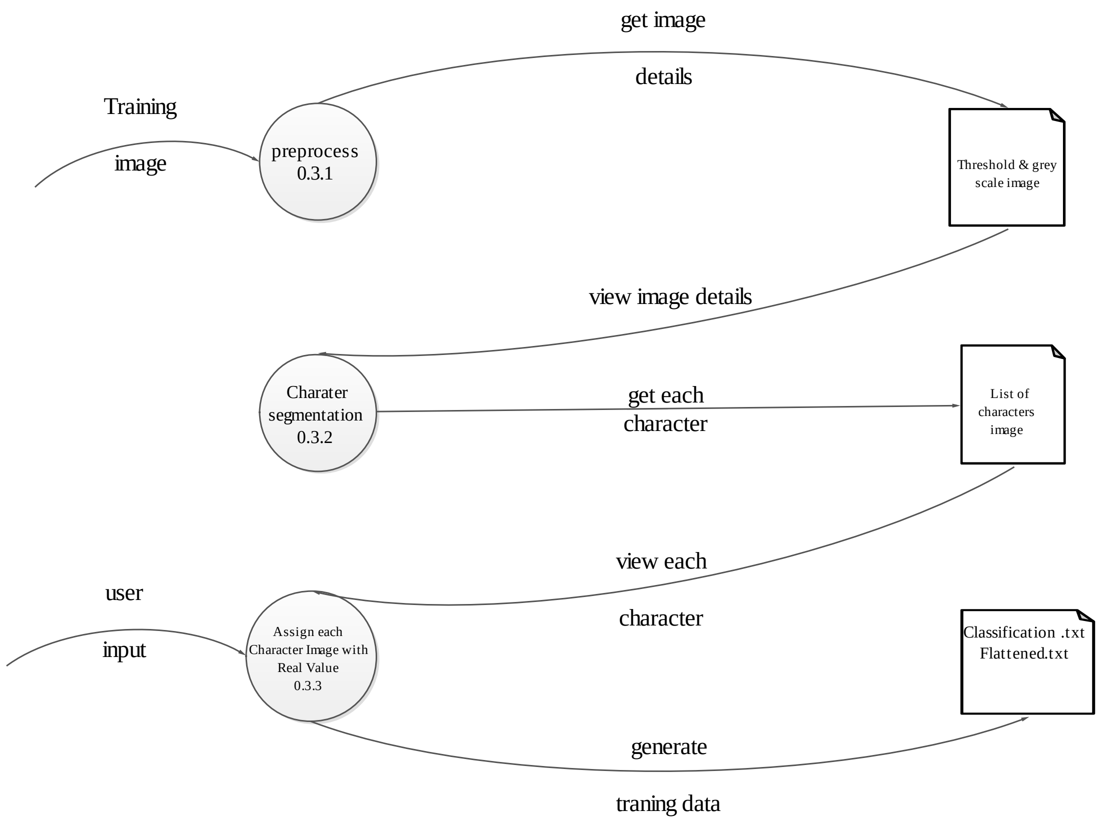


Fig 4.2.4 Level 2 DFD of Generate Training Data

4.3 Design of Each Subsystem

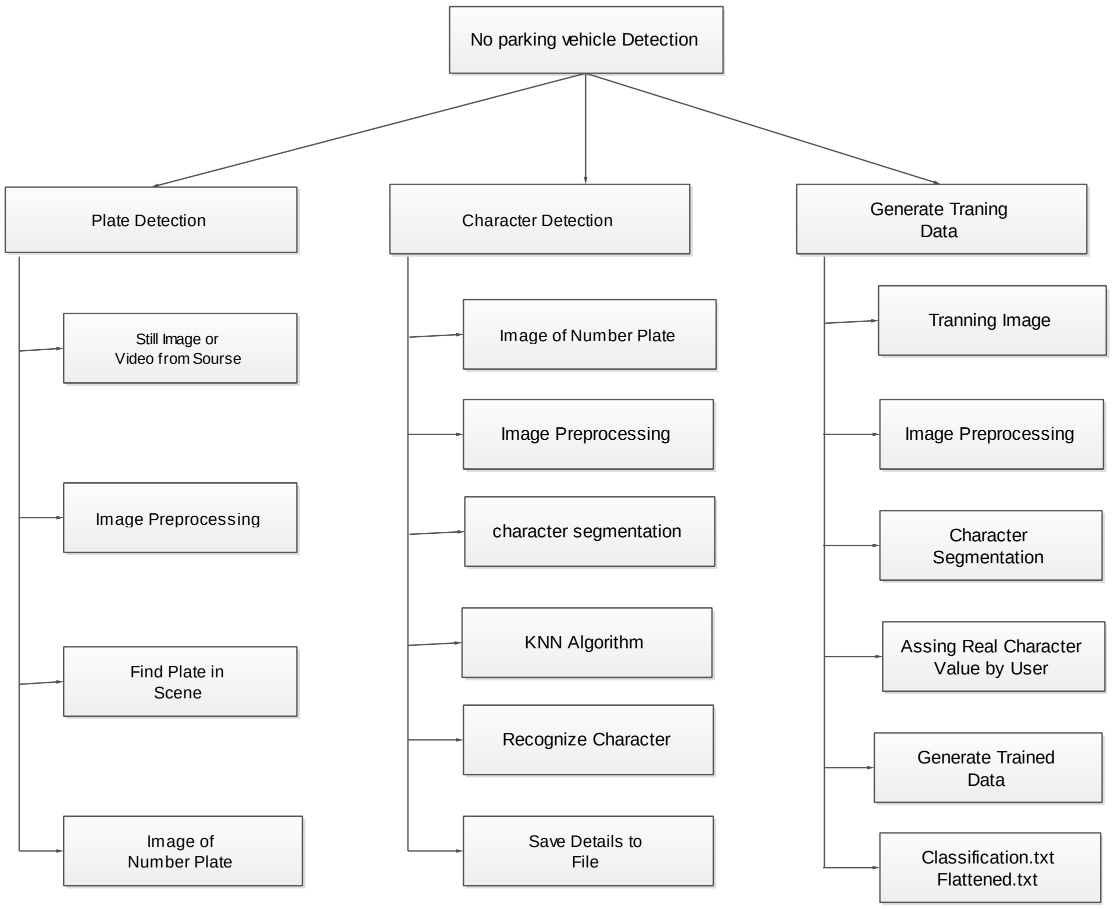


Fig 4.3 Subsystem Design

4.4 UML Diagrams

4.4.1 Use case Diagram

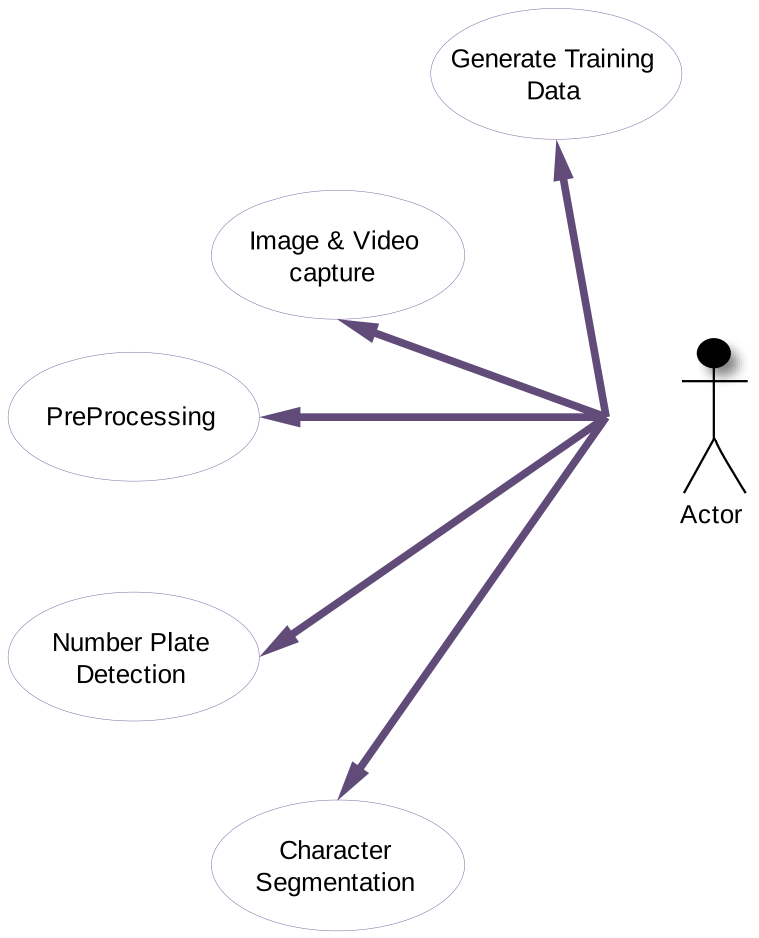


Fig 4.4.1 Use case Diagram

4.4.2 Sequence Diagram

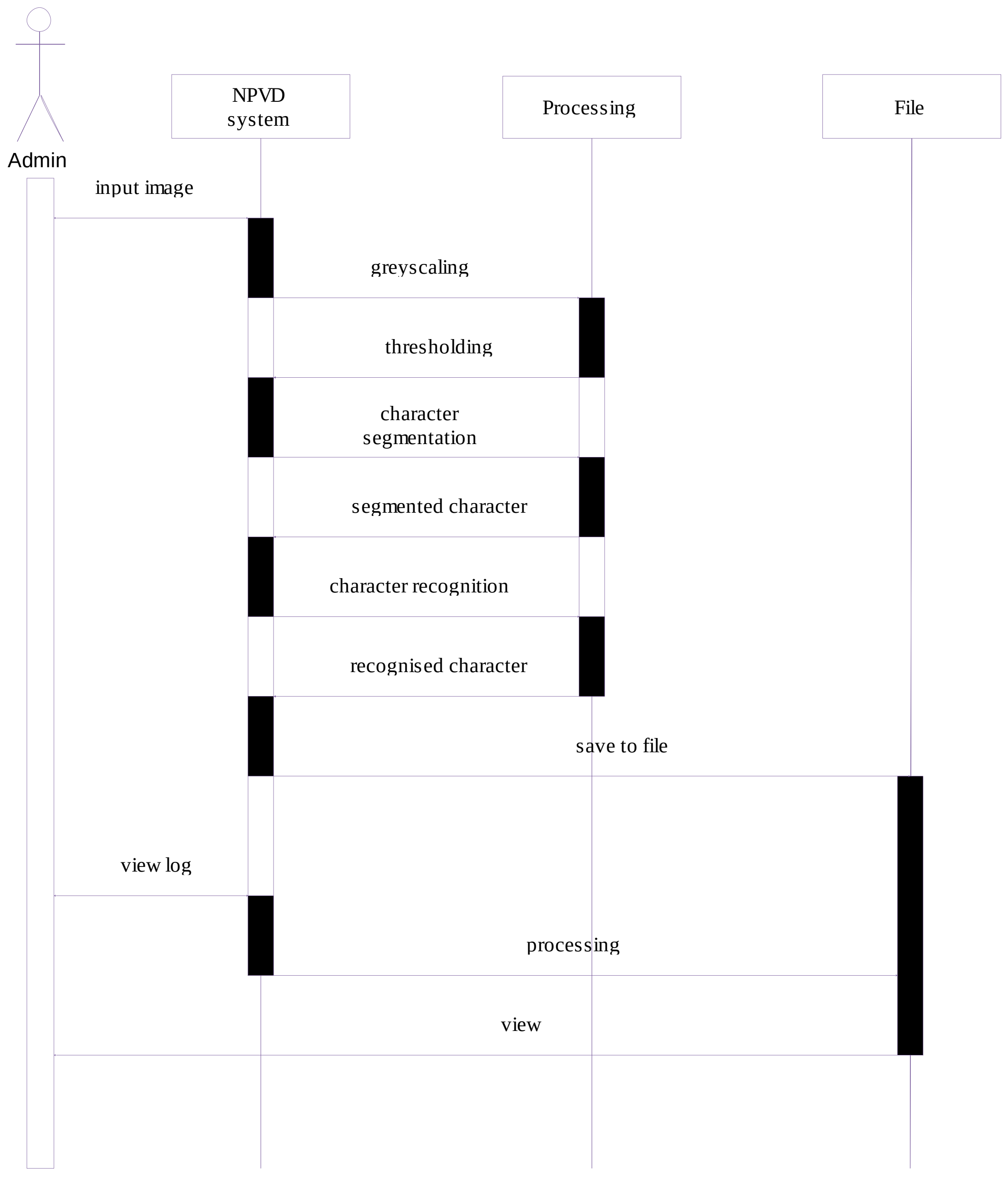


Fig 4.4.2 Sequence Diagram

CHAPTER 5
CODING

5.1 Features of Language

Python Overview

Python is a general-purpose, interpreted, high-level programming language which is widely used nowadays .It is an open source language which was developed by Guido Van Rossum in the late 1980s .Python Software Foundation (PSF), a non-profit organization, holds the intellectual property rights of Python. Python was released in 1991 at Centrum Wiskunde & Informatica (CWI) in the Netherlands as a successor to the ABC language. Rossum named this language after a popular comedy show called 'Monty Python's Flying Circus' (and not after Python-the snake).In the last few years, the popularity of python has increased immensely due to its wide applications. According to most of the tech surveys, Python is in the top ten Most Popular Technologies in 2019.

Python Features

Python is an interpreter-based language, which allows execution of one instruction at a time.

1) Extensive basic data types are supported e.g. numbers (floating point, complex, and unlimited-length long integers), strings (both ASCII and Unicode), lists, and dictionaries.
2) Variables can be strongly typed as well as dynamic typed.
3) Supports object-oriented programming concepts such as class, inheritance, objects, module, namespace etc.
4) Cleaner exception handling support.
5) Supports automatic memory management.

Python Advantages

1) Python provides enhanced readability. For that purpose, uniform indents are used to delimit blocks of statements instead of curly brackets, like in many languages such as C, C++ and Java.
2) Python is free and distributed as open-source software. A large programming community is actively involved in the development and support of Python

libraries for various applications such as web frameworks, mathematical computing and data science.

3) Python is a cross-platform language. It works equally on different OS platforms like Windows, Linux, Mac OSX etc. Hence Python applications can be easily ported across OS platforms.
4) Python supports multiple programming paradigms including imperative, procedural, object-oriented and functional programming styles.

5) Python is an extensible language. Additional functionality (other than what is provided in the core language) can be made available through modules and packages written in other languages (C, C++, Java etc)
6) A standard DB-API for database connectivity has been defined in Python. It can be enabled using any data source (Oracle, MySQL, SQLite etc.) as a backend to the Python program for storage, retrieval and processing of data.
7) Standard distribution of Python contains the Tkinter GUI toolkit, which is the implementation of popular GUI library called Tcl/Tk. An attractive GUI can be constructed using Tkinter. Many other GUI libraries like Qt, GTK, WxWidgets etc. are also ported to Python.
8) Python can be integrated with other popular programming technologies like C, C++, Java, ActiveX and CORBA.

Python Applications

Even though Python started as a general-purpose programming language with no particular application as its focus, over last few years it has emerged as the language of choice for developers in some application areas. Some important applications of Python are summarized below:

Machine Learning

This is another key application area of Python. Python libraries such as Scikit-learn, Tensorflow and NLTK are widely used for the prediction of trends like customer satisfaction, projected values of stocks etc. Some of the real-world applications of machine learning include medical diagnosis,sales prediction, feedback analysis etc.

Image Processing

The OpenCV library is commonly used for face detection and gesture recognition. OpenCV is a C++ library, but has been ported to Python. Because of the rapid development of this feature, Python is a very popular choice from image processing.

Tkinter

Tkinter is the standard GUI library for Python. Python when combined with Tkinter provides a fast and easy way to create GUI applications. Tkinter provides a powerful object-oriented interface to the Tk GUI toolkit.

OpenCV

OpenCV is a cross-platform library using which we can develop real-time computer vision applications. It mainly focuses on image processing, video capture and analysis including features like face detection and object detection. Computer Vision can be defined as a discipline that explains how to reconstruct, interrupt, and understand a 3D scene from its 2D images, in terms of the properties of the structure present in the scene. It deals with modeling and replicating human vision using computer software and hardware.

5.2 Functional Description

# Main.py

```
import cv2
import numpy as np
import os
import re
import DetectChars
import DetectPlates
import PossiblePlate
import csv
from datetime import datetime
import time
# 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
```

```
def main(imgdata):
   print("entered")

   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 if0

   imgOriginalScene  = imgdata              # open image

   if imgOriginalScene is None:                          # if image was not read successfully
      print("\nerror: image not read from file \n\n")  # print error message to std out
      os.system("pause")                               # pause so user can see error message
      return                                    # and exit program
   # end if

   listOfPossiblePlates = DetectPlates.detectPlatesInScene(imgOriginalScene)           #
detect plates

   listOfPossiblePlates = DetectChars.detectCharsInPlates(listOfPossiblePlates)        #
detect chars in plates

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

   if len(listOfPossiblePlates) == 0:                        # if no plates were found
      print("\nno license plates were detected\n")  # inform user no plates were found
   else:                                                # else
            # if we get in here list of possible plates has at leat one plate

            # sort the list of possible plates in DESCENDING order (most number of
chars to least number of chars)
      listOfPossiblePlates.sort(key = lambda possiblePlate: len(possiblePlate.strChars),
reverse = True)

            # suppose the plate with the most recognized chars (the first plate in sorted
by string length descending order) is the actual plate
      licPlate = listOfPossiblePlates[0]
```

```
        cv2.imshow("imgPlate", licPlate.imgPlate)           # show crop of plate and
threshold of plate
        cv2.imshow("imgThresh", licPlate.imgThresh)

        if len(licPlate.strChars) == 0:                     # if no chars were found in the plate
            print("\nno characters were detected\n\n")  # show message
            return                                          # and exit program
        # end if

        #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("----------------------------------------")

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

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


        date=str(datetime.now().strftime('%Y_%m_%d'))
        time=str(datetime.now().strftime('%H_%M_%S'))
        filename = str(datetime.now().strftime('%Y_%m_%d_%H_%M_%S'))
        flag=1
        with open(r"logs/log.txt", newline = "") as file:
           reader = csv.reader(file)
           for col in reader:
              c = 0
              for row in col:

                  if c==2:
                      datechek=row
                  if c==1:
                      numberchek=row
                  c += 1
              if datechek==date and numberchek==licPlate.strChars:
                  flag=0
        #file.close()
        if flag==1:
            file=open(r"logs/log.txt","a+")
```

```
file.write(filename+".png,"+licPlate.strChars+","+date+","+time+","+filename+"\n")
            cv2.imwrite(r"logs/images/{}.png".format(filename), imgOriginalScene)
# write image out to file
        else:
            print("already exist")
    # end if else

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

    return
```

# DetectChars.py

```
import os
import cv2
import numpy as np
import math
import random
import Main
import Preprocess
import PossibleChar

# module level variables

kNearest = cv2.ml.KNearest_create()

        # constants for checkIfPossibleChar, this checks one possible char only (does not
compare to another char)
MIN_PIXEL_WIDTH = 2
MIN_PIXEL_HEIGHT = 8
MIN_ASPECT_RATIO = 0.25
MAX_ASPECT_RATIO = 1.0
MIN_PIXEL_AREA = 80
        # constants for comparing two chars
MIN_DIAG_SIZE_MULTIPLE_AWAY = 0.3
MAX_DIAG_SIZE_MULTIPLE_AWAY = 5.0

MAX_CHANGE_IN_AREA = 0.5

MAX_CHANGE_IN_WIDTH = 0.8
```

```
MAX_CHANGE_IN_HEIGHT = 0.2
MAX_ANGLE_BETWEEN_CHARS = 12.0

        # other constants
MIN_NUMBER_OF_MATCHING_CHARS = 3
RESIZED_CHAR_IMAGE_WIDTH = 20
RESIZED_CHAR_IMAGE_HEIGHT = 30
MIN_CONTOUR_AREA = 100

def loadKNNDataAndTrainKNN():
    allContoursWithData = []                # declare empty lists,
    validContoursWithData = []              # we will fill these shortly

    try:
        npaClassifications = np.loadtxt("classifications.txt", np.float32)                  # read
in training classifications
    except:                                                                 # if file could not be
opened
        print("error, unable to open classifications.txt, exiting program\n")  # show error
message
        os.system("pause")
        return False                                                        # and return False
    # end try

    try:
        npaFlattenedImages = np.loadtxt("flattened_images.txt", np.float32)                 #
read in training images
    except:                                                                 # if file could not be
opened
        print("error, unable to open flattened_images.txt, exiting program\n")  # show
error message
        os.system("pause")
        return False                                                        # and return False
    # end try

    npaClassifications = npaClassifications.reshape((npaClassifications.size, 1))       #
reshape numpy array to 1d, necessary to pass to call to train

    kNearest.setDefaultK(1)                                                    # set default K to 1

    kNearest.train(npaFlattenedImages, cv2.ml.ROW_SAMPLE, npaClassifications)
# train KNN object

    return True                         # if we got here training was successful so return true
```

```
# end function

def detectCharsInPlates(listOfPossiblePlates):
   intPlateCounter = 0
   imgContours = None
   contours = []

   if len(listOfPossiblePlates) == 0:          # if list of possible plates is empty
      return listOfPossiblePlates             # return
   # end if

         # at this point we can be sure the list of possible plates has at least one plate

   for possiblePlate in listOfPossiblePlates:          # for each possible plate, this is a big
for loop that takes up most of the function

      possiblePlate.imgGrayscale, possiblePlate.imgThresh =
Preprocess.preprocess(possiblePlate.imgPlate)     # preprocess to get grayscale and
threshold images

      if Main.showSteps == True: # show steps
         cv2.imshow("5a", possiblePlate.imgPlate)
         cv2.imshow("5b", possiblePlate.imgGrayscale)
         cv2.imshow("5c", possiblePlate.imgThresh)
      # end if # show steps

            # increase size of plate image for easier viewing and char detection
      possiblePlate.imgThresh = cv2.resize(possiblePlate.imgThresh, (0, 0), fx = 1.6,
fy = 1.6)

            # threshold again to eliminate any gray areas
      thresholdValue, possiblePlate.imgThresh =
cv2.threshold(possiblePlate.imgThresh, 0.0, 255.0, cv2.THRESH_BINARY |
cv2.THRESH_OTSU)

      if Main.showSteps == True: # show steps
         cv2.imshow("5d", possiblePlate.imgThresh)
      # end if # show steps

            # find all possible chars in the plate,
            # this function first finds all contours, then only includes contours that could
be chars (without comparison to other chars yet)
      listOfPossibleCharsInPlate =
findPossibleCharsInPlate(possiblePlate.imgGrayscale, possiblePlate.imgThresh)
```

```
        if Main.showSteps == True: # show steps
            height, width, numChannels = possiblePlate.imgPlate.shape
            imgContours = np.zeros((height, width, 3), np.uint8)
            del contours[:]                                    # clear the contours list

            for possibleChar in listOfPossibleCharsInPlate:
                contours.append(possibleChar.contour)
            # end for

            cv2.drawContours(imgContours, contours, -1, Main.SCALAR_WHITE)

            cv2.imshow("6", imgContours)
        # end if # show steps

                # given a list of all possible chars, find groups of matching chars within the
plate
        listOfListsOfMatchingCharsInPlate =
findListOfListsOfMatchingChars(listOfPossibleCharsInPlate)

        if Main.showSteps == True: # show steps
            imgContours = np.zeros((height, width, 3), np.uint8)
            del contours[:]

            for listOfMatchingChars in listOfListsOfMatchingCharsInPlate:
                intRandomBlue = random.randint(0, 255)
                intRandomGreen = random.randint(0, 255)
                intRandomRed = random.randint(0, 255)

                for matchingChar in listOfMatchingChars:
                    contours.append(matchingChar.contour)
                # end for
                cv2.drawContours(imgContours, contours, -1, (intRandomBlue,
intRandomGreen, intRandomRed))
            # end for
            cv2.imshow("7", imgContours)
        # end if # show steps

        if (len(listOfListsOfMatchingCharsInPlate) == 0):                        # if no
groups of matching chars were found in the plate

            if Main.showSteps == True: # show steps
                print("chars found in plate number " + str(
                    intPlateCounter) + " = (none), click on any image and press a key to
continue . . .")
                intPlateCounter = intPlateCounter + 1
                cv2.destroyWindow("8")
                cv2.destroyWindow("9")
                cv2.destroyWindow("10")
                cv2.waitKey(0)
            # end if # show steps

            possiblePlate.strChars = ""
            continue                                        # go back to top of for
loop
        # end if

        for i in range(0, len(listOfListsOfMatchingCharsInPlate)):                  #
within each list of matching chars
            listOfListsOfMatchingCharsInPlate[i].sort(key = lambda matchingChar:
matchingChar.intCenterX)        # sort chars from left to right
            listOfListsOfMatchingCharsInPlate[i] =
removeInnerOverlappingChars(listOfListsOfMatchingCharsInPlate[i])        # and
remove inner overlapping chars
        # end for

        if Main.showSteps == True: # show steps
            imgContours = np.zeros((height, width, 3), np.uint8)

            for listOfMatchingChars in listOfListsOfMatchingCharsInPlate:
                intRandomBlue = random.randint(0, 255)
                intRandomGreen = random.randint(0, 255)
                intRandomRed = random.randint(0, 255)

                del contours[:]

                for matchingChar in listOfMatchingChars:
                    contours.append(matchingChar.contour)
                # end for

                cv2.drawContours(imgContours, contours, -1, (intRandomBlue,
intRandomGreen, intRandomRed))
            # end for
            cv2.imshow("8", imgContours)
        # end if # show steps
```

```
                # within each possible plate, suppose the longest list of potential matching
chars is the actual list of chars
        intLenOfLongestListOfChars = 0
        intIndexOfLongestListOfChars = 0

                # loop through all the vectors of matching chars, get the index of the one
with the most chars
        for i in range(0, len(listOfListsOfMatchingCharsInPlate)):
            if len(listOfListsOfMatchingCharsInPlate[i]) > intLenOfLongestListOfChars:
                intLenOfLongestListOfChars = len(listOfListsOfMatchingCharsInPlate[i])
                intIndexOfLongestListOfChars = i
            # end if
        # end for

                # suppose that the longest list of matching chars within the plate is the
actual list of chars
        longestListOfMatchingCharsInPlate =
listOfListsOfMatchingCharsInPlate[intIndexOfLongestListOfChars]

        if Main.showSteps == True: # show steps
            imgContours = np.zeros((height, width, 3), np.uint8)
            del contours[:]

            for matchingChar in longestListOfMatchingCharsInPlate:
                contours.append(matchingChar.contour)
            # end for

            cv2.drawContours(imgContours, contours, -1, Main.SCALAR_WHITE)

            cv2.imshow("9", imgContours)
        # end if # show steps
        possiblePlate.strChars = recognizeCharsInPlate(possiblePlate.imgThresh,
longestListOfMatchingCharsInPlate)

        if Main.showSteps == True: # show steps
            print("chars found in plate number " + str(
                intPlateCounter) + " = " + possiblePlate.strChars + ", click on any image
and press a key to continue . . .")
            intPlateCounter = intPlateCounter + 1
            cv2.waitKey(0)
        # end if # show steps

    # end of big for loop that takes up most of the function
    if Main.showSteps == True:
        print("\nchar detection complete, click on any image and press a key to continue
. . .\n")
        cv2.waitKey(0)
    # end if

    return listOfPossiblePlates
# end function
def findPossibleCharsInPlate(imgGrayscale, imgThresh):
    listOfPossibleChars = []                    # this will be the return value
    contours = []
    imgThreshCopy = imgThresh.copy()

            # find all contours in plate
    contours, npaHierarchy = cv2.findContours(imgThreshCopy, cv2.RETR_LIST,
cv2.CHAIN_APPROX_SIMPLE)

    for contour in contours:                        # for each contour
        possibleChar = PossibleChar.PossibleChar(contour)

        if checkIfPossibleChar(possibleChar):              # if contour is a possible char,
note this does not compare to other chars (yet) . . .
            listOfPossibleChars.append(possibleChar)       # add to list of possible chars
        # end if
    # end if

    return listOfPossibleChars
# end function
def checkIfPossibleChar(possibleChar):
            # this function is a 'first pass' that does a rough check on a contour to see if it
could be a char,
            # note that we are not (yet) comparing the char to other chars to look for a
group
    if (possibleChar.intBoundingRectArea > MIN_PIXEL_AREA and
        possibleChar.intBoundingRectWidth > MIN_PIXEL_WIDTH and
possibleChar.intBoundingRectHeight > MIN_PIXEL_HEIGHT and
        MIN_ASPECT_RATIO < possibleChar.fltAspectRatio and
possibleChar.fltAspectRatio < MAX_ASPECT_RATIO):
        return True
    else:
        return False
    # end if
# end function
```

```
def findListOfListsOfMatchingChars(listOfPossibleChars):
            # with this function, we start off with all the possible chars in one big list
            # the purpose of this function is to re-arrange the one big list of chars into a
list of lists of matching chars,
            # note that chars that are not found to be in a group of matches do not need to
be considered further
    listOfListsOfMatchingChars = []                  # this will be the return value

    for possibleChar in listOfPossibleChars:                        # for each possible char in
the one big list of chars
        listOfMatchingChars = findListOfMatchingChars(possibleChar,
listOfPossibleChars)        # find all chars in the big list that match the current char

        listOfMatchingChars.append(possibleChar)                # also add the current char
to current possible list of matching chars

        if len(listOfMatchingChars) < MIN_NUMBER_OF_MATCHING_CHARS:     #
if current possible list of matching chars is not long enough to constitute a possible
plate
            continue                            # jump back to the top of the for loop and try again
with next char, note that it's not necessary
                                    # to save the list in any way since it did not have
enough chars to be a possible plate
        # end if

                                    # if we get here, the current list passed test as a "group"
or "cluster" of matching chars
        listOfListsOfMatchingChars.append(listOfMatchingChars)      # so add to our list
of lists of matching chars

        listOfPossibleCharsWithCurrentMatchesRemoved = []

                                    # remove the current list of matching chars from the big
list so we don't use those same chars twice,
                                    # make sure to make a new big list for this since we
don't want to change the original big list
        listOfPossibleCharsWithCurrentMatchesRemoved = list(set(listOfPossibleChars)
- set(listOfMatchingChars))

        recursiveListOfListsOfMatchingChars =
findListOfListsOfMatchingChars(listOfPossibleCharsWithCurrentMatchesRemoved)
# recursive call
```

```
        for recursiveListOfMatchingChars in recursiveListOfListsOfMatchingChars:
# for each list of matching chars found by recursive call
            listOfListsOfMatchingChars.append(recursiveListOfMatchingChars)             #
add to our original list of lists of matching chars
        # end for

        break       # exit for

    # end for

    return listOfListsOfMatchingChars
# end function

def findListOfMatchingChars(possibleChar, listOfChars):
            # the purpose of this function is, given a possible char and a big list of possible
chars,
            # find all chars in the big list that are a match for the single possible char, and
return those matching chars as a list
    listOfMatchingChars = []                # this will be the return value

    for possibleMatchingChar in listOfChars:                # for each char in big list
        if possibleMatchingChar == possibleChar:    # if the char we attempting to find
matches for is the exact same char as the char in the big list we are currently checking
                                                    # then we should not include it in the list of matches
b/c that would end up double including the current char
            continue                                # so do not add to list of matches and jump back
to top of for loop
        # end if
                    # compute stuff to see if chars are a match
        fltDistanceBetweenChars = distanceBetweenChars(possibleChar,
possibleMatchingChar)

        fltAngleBetweenChars = angleBetweenChars(possibleChar,
possibleMatchingChar)

        fltChangeInArea = float(abs(possibleMatchingChar.intBoundingRectArea -
possibleChar.intBoundingRectArea)) / float(possibleChar.intBoundingRectArea)

        fltChangeInWidth = float(abs(possibleMatchingChar.intBoundingRectWidth -
possibleChar.intBoundingRectWidth)) / float(possibleChar.intBoundingRectWidth)
        fltChangeInHeight = float(abs(possibleMatchingChar.intBoundingRectHeight -
possibleChar.intBoundingRectHeight)) / float(possibleChar.intBoundingRectHeight)

                # check if chars match
        if (fltDistanceBetweenChars < (possibleChar.fltDiagonalSize *
MAX_DIAG_SIZE_MULTIPLE_AWAY) and
            fltAngleBetweenChars < MAX_ANGLE_BETWEEN_CHARS and
            fltChangeInArea < MAX_CHANGE_IN_AREA and
            fltChangeInWidth < MAX_CHANGE_IN_WIDTH and
            fltChangeInHeight < MAX_CHANGE_IN_HEIGHT):

            listOfMatchingChars.append(possibleMatchingChar)        # if the chars are a
match, add the current char to list of matching chars
        # end if
    # end for

    return listOfMatchingChars                  # return result
# end function
# use Pythagorean theorem to calculate distance between two chars
def distanceBetweenChars(firstChar, secondChar):
    intX = abs(firstChar.intCenterX - secondChar.intCenterX)
    intY = abs(firstChar.intCenterY - secondChar.intCenterY)

    return math.sqrt((intX ** 2) + (intY ** 2))
# end function
# use basic trigonometry (SOH CAH TOA) to calculate angle between chars
def angleBetweenChars(firstChar, secondChar):
    fltAdj = float(abs(firstChar.intCenterX - secondChar.intCenterX))
    fltOpp = float(abs(firstChar.intCenterY - secondChar.intCenterY))

    if fltAdj != 0.0:                           # check to make sure we do not divide by zero if
the center X positions are equal, float division by zero will cause a crash in Python
        fltAngleInRad = math.atan(fltOpp / fltAdj)      # if adjacent is not zero, calculate
angle
    else:
        fltAngleInRad = 1.5708                          # if adjacent is zero, use this as the
angle, this is to be consistent with the C++ version of this program
    # end if

    fltAngleInDeg = fltAngleInRad * (180.0 / math.pi)       # calculate angle in degrees

    return fltAngleInDeg
# end function
# if we have two chars overlapping or to close to each other to possibly be separate
chars, remove the inner (smaller) char,
# this is to prevent including the same char twice if two contours are found for the
same char,
# for example for the letter 'O' both the inner ring and the outer ring may be found as
contours, but we should only include the char once
def removeInnerOverlappingChars(listOfMatchingChars):
    listOfMatchingCharsWithInnerCharRemoved = list(listOfMatchingChars)
# this will be the return value

    for currentChar in listOfMatchingChars:
        for otherChar in listOfMatchingChars:
            if currentChar != otherChar:        # if current char and other char are not the
same char . . .
                                                # if current char and other char have
center points at almost the same location . . .
                if distanceBetweenChars(currentChar, otherChar) <
(currentChar.fltDiagonalSize * MIN_DIAG_SIZE_MULTIPLE_AWAY):
                                # if we get in here we have found overlapping chars
                                # next we identify which char is smaller, then if that char was not
already removed on a previous pass, remove it
                    if currentChar.intBoundingRectArea < otherChar.intBoundingRectArea:
# if current char is smaller than other char
                        if currentChar in listOfMatchingCharsWithInnerCharRemoved:
# if current char was not already removed on a previous pass . . .
                            listOfMatchingCharsWithInnerCharRemoved.remove(currentChar)
# then remove current char
                        # end if
                    else:                                       # else if other char is
smaller than current char
                        if otherChar in listOfMatchingCharsWithInnerCharRemoved:
# if other char was not already removed on a previous pass . . .
                            listOfMatchingCharsWithInnerCharRemoved.remove(otherChar)
# then remove other char
                        # end if
                    # end if
                # end if
            # end if
        # end for
    # end for

    return listOfMatchingCharsWithInnerCharRemoved
# end function
# this is where we apply the actual char recognition
def recognizeCharsInPlate(imgThresh, listOfMatchingChars):
    strChars = ""               # this will be the return value, the chars in the lic plate

    height, width = imgThresh.shape
    imgThreshColor = np.zeros((height, width, 3), np.uint8)

    listOfMatchingChars.sort(key = lambda matchingChar: matchingChar.intCenterX)
# sort chars from left to right

    cv2.cvtColor(imgThresh, cv2.COLOR_GRAY2BGR, imgThreshColor)
# make color version of threshold image so we can draw contours in color on it

    for currentChar in listOfMatchingChars:                                  # for each char in
plate
        pt1 = (currentChar.intBoundingRectX, currentChar.intBoundingRectY)
        pt2 = ((currentChar.intBoundingRectX + currentChar.intBoundingRectWidth),
(currentChar.intBoundingRectY + currentChar.intBoundingRectHeight))

        cv2.rectangle(imgThreshColor, pt1, pt2, Main.SCALAR_GREEN, 2)          #
draw green box around the char

                # crop char out of threshold image
        imgROI = imgThresh[currentChar.intBoundingRectY :
currentChar.intBoundingRectY + currentChar.intBoundingRectHeight,
                           currentChar.intBoundingRectX : currentChar.intBoundingRectX +
currentChar.intBoundingRectWidth]

        imgROIResized = cv2.resize(imgROI, (RESIZED_CHAR_IMAGE_WIDTH,
RESIZED_CHAR_IMAGE_HEIGHT))          # resize image, this is necessary for
char recognition

        npaROIResized = imgROIResized.reshape((1,
RESIZED_CHAR_IMAGE_WIDTH * RESIZED_CHAR_IMAGE_HEIGHT))        #
flatten image into 1d numpy array

        npaROIResized = np.float32(npaROIResized)             # convert from 1d numpy
array of ints to 1d numpy array of floats

        retval, npaResults, neigh_resp, dists = kNearest.findNearest(npaROIResized, k =
1)             # finally we can call findNearest !!!

        strCurrentChar = str(chr(int(npaResults[0][0])))          # get character from
results

        strChars = strChars + strCurrentChar                   # append current char to full
string
```

```
    # end for

    if Main.showSteps == True: # show steps
        cv2.imshow("10", imgThreshColor)
    # end if # show steps
    return strChars
# end function
```

CHAPTER 6
TESTING

Testing is a set of activities that can be planned in advance and conducted systematically. System testing is that stage of implementation that is aimed for ensuring that the system works accurately and efficiently before live operation commences. System testing is the execution of the program to check logical changes and its impact on the output of the system with the intention of finding error. A successful test is the one that uncovers a yet undiscovered error. Testing is vital to the success of the system

A strategy for software testing integrates software test case design methods into well planned series of steps that result in successful completion of the software. A software testing strategy should be flexible enough to promote a customized testing approach. A strategy for testing should accommodate low level test that are necessary to verify implemented as well as high level tests that validate major system functions against customer requirements.

Levels of Testing

Testing is done in the following levels:

1. Unit Testing
2. Integration Testing
3. Validation Testing
4. Output Testing

Unit Testing

In this level of testing each of the modules were tested one by one individually, before they were integrated. The modules were tested as soon as they were developed. It helped to recognize areas requiring modifications and corrections. In this system, unit testing can be easily accomplished. The project is divided into four modules training data generation, number plate detection, character segmentation and user. The modules includes various activities. In the case of this project, each module was built separately and was tested and verified individually before integrating them to a complete project.

Integration Testing

The modules that are tested individually and confirmed to be working according to specifications are then integrated to form the entire system. The training data generation, number plate detection, character segmentation and user modules are integrated in this testing step and then the entire system is tested. The interface between the modules are thoroughly tested. It is ensured that data and controls are passed properly between modules

Validation Testing

At the end of the integration testing software is completely assembled as a package and interfacing errors have been uncovered and corrected and final series of software validation testing begins

Once the system is integrated and tested to operate properly, it is tested to see if the software developed meets all functional, behavioural and performance requirements. The errors which were uncovered during Integration testing were covered here.

In this system. I have tested the different forms to see whether the inputs given to the forms are stored in the appropriate entries. I also tested the home page with image/video inputs, to see whether the inputs are sent to the system and are validated properly. Hence, I found that the validation testing had carried out successfully.

Output Testing

Here I have tested the system to determine whether it produce the required output or not and to ensure the correctness of the output and its format. In my project testing is implemented in each of the modules. The following test cases are performed to do the testing.

1. Proper opening and closing of window.

2. Appropriate menu bars displayed in appropriate section.

3. A proper working of system is checked for multiple input modes.

Like this all other processes are tested and hence come under the conclusion that there arise no errors. It also found that there arise no modification and corrections in any areas of the project.

CHAPTER 7

IMPLEMENTATION

Implementation is the stage of the project where theoretical design is turned into a working system. If the implementation is not carefully planned and controlled, it can cause chaos and confusion. Proper implementation is essential to provide a reliable system to meet organization requirements. Successful implementation may not guarantee improvement in the organization using the system, but proper installation will prevent it. The process of putting the developed system in actual use is called system implementation. The system can only be implemented after thorough testing is done and if it is found to be working according to the specifications.

The implementation stage involves following tasks:

1. Careful Planning
2. Investigation of system and constraints.
3. Design of methods to achieve the changeover.
4. Training of the staff in the changeover phase.
5. Evaluation of the changeover method.

In order to have the system running of a specific machine, the following components are needed.

1) Python
2) A preferable operating system like windows 8 or windows 10
3) Spyder3
4) High Resolution Camera

Parallel run is done and both the computerized and manual systems are executed in parallel manual result can be compared with the result of computerized system. For the case of demonstration of the success of this system, it was implemented with successfully running; manual systems results are verified.

CHAPTER 8

SECURITY, BACKUP AND RECOVERY MECHANISMS

Security is an important consideration in desktop application. The first step in securing our application is deciding where we need security and what will be needed to protect.

Security concepts:

1. Authentication
2. Authorization

Authentication

This is the process of determining users identify and forcing users to prove they are who they claim to be usually this involves entering username and password in login page.

Authorization

Once the user is authenticated, authorization is the process of determining whether the user has sufficient permission to perform a given action or not, such as viewing a page or retrieving information from the database.

8.1 User Manual

The user manual provides the detailed description regarding the usage of the software.

The main user tips are:

1. Open file and select OpenImage to open image from local storage.
2. Open file and select OpenVideo to open video from local storage.
3. Open file and select LiveStream to live stream from external camera device.
4. Open file and select Exit to exit from system.
5. Open logs to view all log details of detected vehicles.
6. Use ClearAll to clear them all.

Log in for the first time, please follow the steps below:

1. Click On file , and select Train Data.
2. Input character from keyboard according to the image shown in the display.
3. After Successful training , use the system as usual.

CHAPTER 9

CONCLUSION

Since our system No parking vehicle detection utilises advanced computational features like image processing techniques for identifying the vehicle number plate from images , it reduces the human workload and increases accuracy and provides faster response. The system uses series of image processing techniques and after identification, stores the vehicle details to the database. It is widely believed that the price to be paid for improved performance have sharply increased computational costs. It have shown that this is not necessarily so. Finely sampled pyramids may be obtained inexpensively by extrapolation from coarsely sampled ones. This insight decreases computational cost substantially and help to implement our project at a low cost.

CHAPTER 10
FUTURE ENHANCEMENT

Our project has a very vast scope in the future. The model may be expanded to use in areas like Parking management system or Tolling System. In future mobile apps can be implemented to access information about the logs of the system or real time monitoring of the situation from a remote device. This enhancement will make it a better utility for mobile users as well in future

APPENDIX

Input and Output Forms

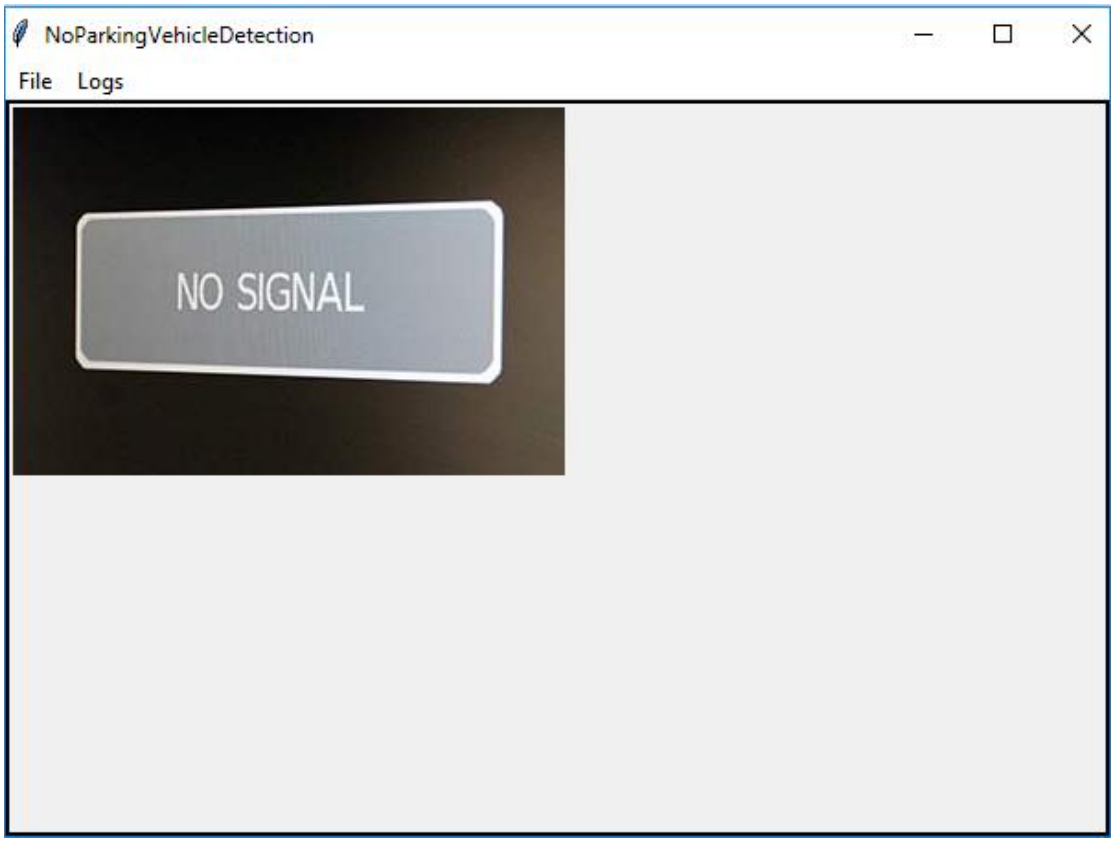


Fig 1: NPVD Home Page

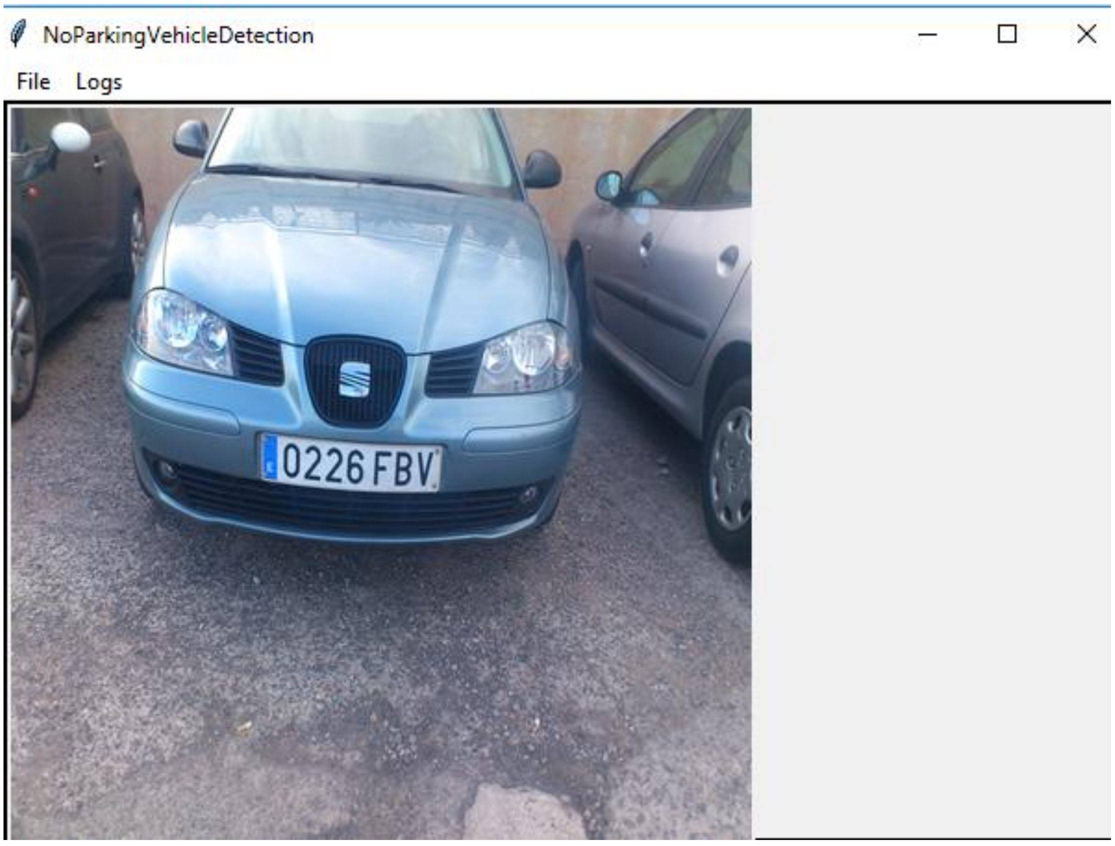
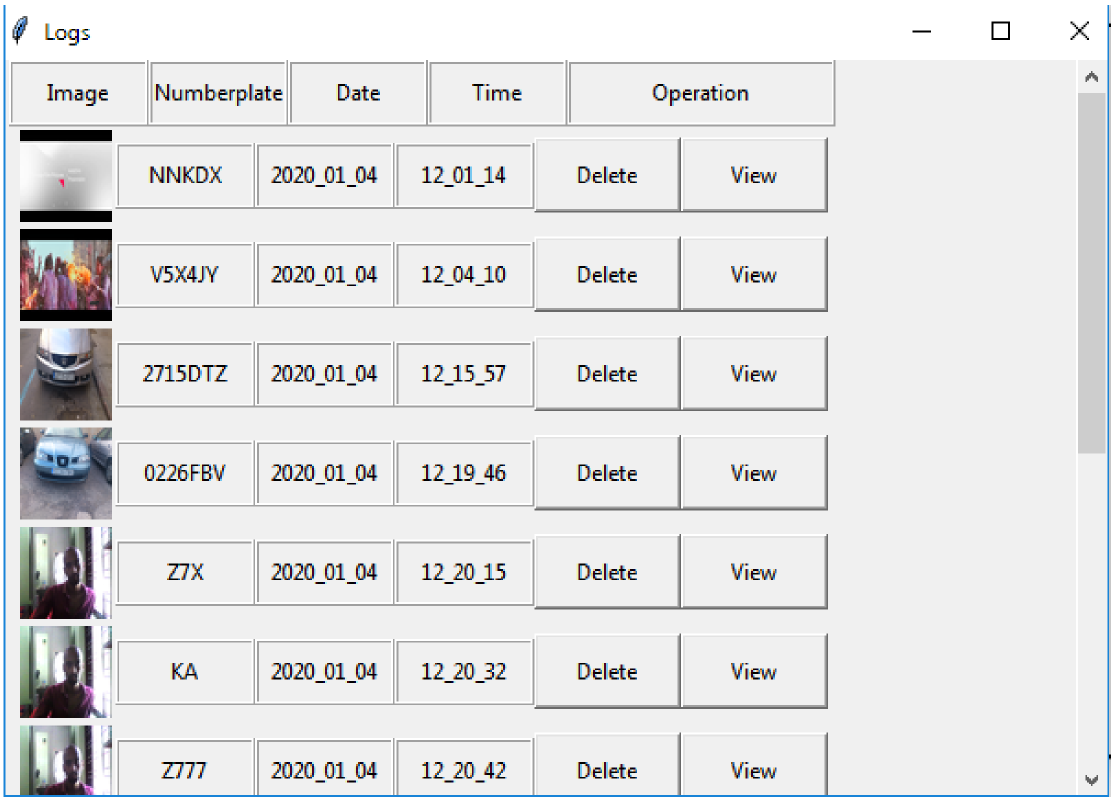


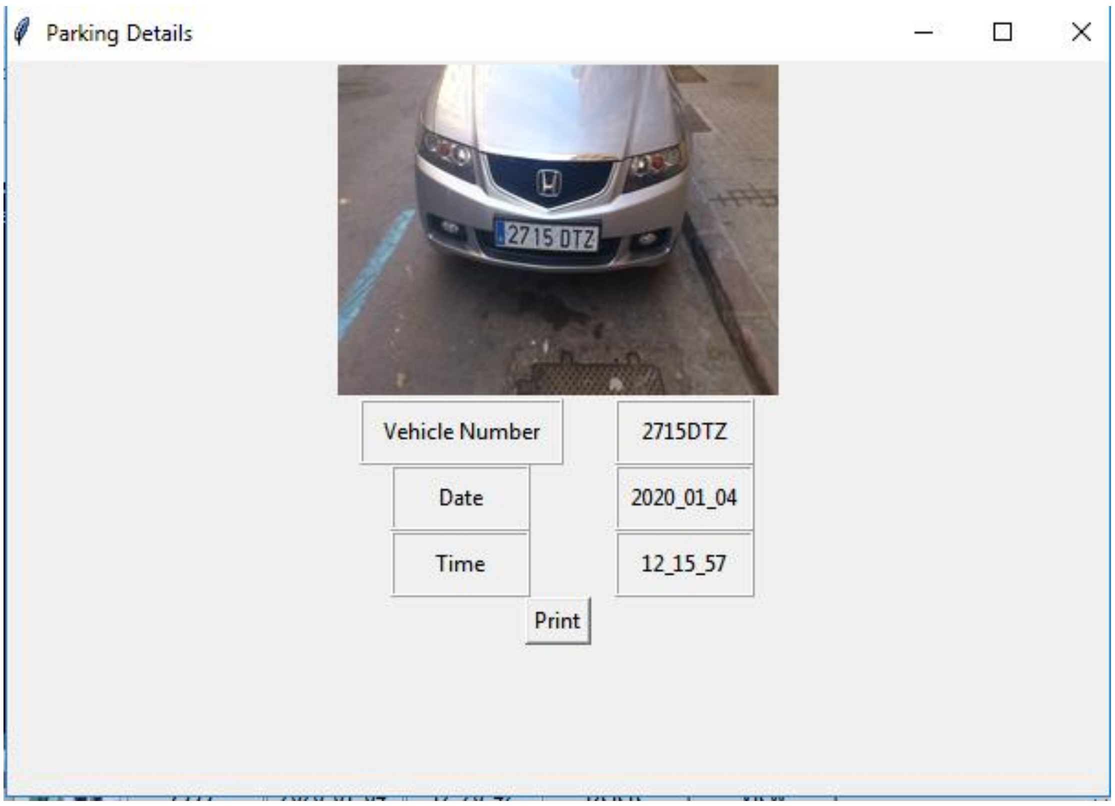
Fig 2: NPVD Live Stream Page



Logs

Image	Numberplate	Date	Time	Operation	
	NNKDX	2020_01_04	12_01_14	Delete	View
	V5X4JY	2020_01_04	12_04_10	Delete	View
	2715DTZ	2020_01_04	12_15_57	Delete	View
	0226FBV	2020_01_04	12_19_46	Delete	View
	Z7X	2020_01_04	12_20_15	Delete	View
	KA	2020_01_04	12_20_32	Delete	View
	Z777	2020_01_04	12_20_42	Delete	View

Fig 3: NPVD View Log Page


Parking Details

Vehicle Number	2715DTZ
Date	2020_01_04
Time	12_15_57

Fig 4: NPVD Vehicle Details

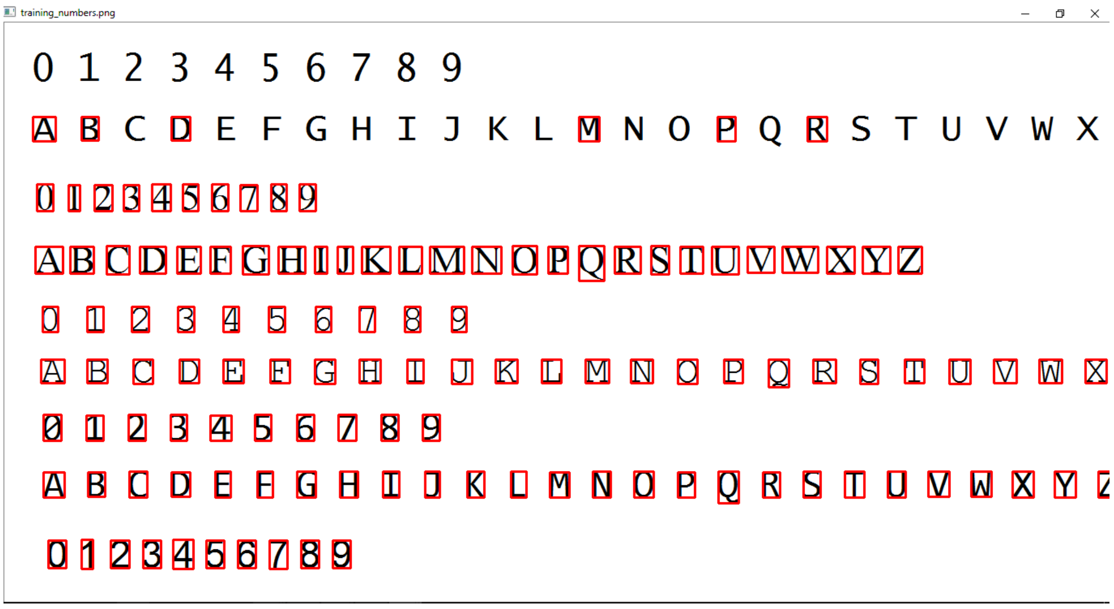

training_numbers.png
0 1 2 3 4 5 6 7 8 9
A B C D E F G H I J K L M N O P Q R S T U V W X
0 1 2 3 4 5 6 7 8 9
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
0 1 2 3 4 5 6 7 8 9
A B C D E F G H I J K L M N O P Q R S T U V W X
0 1 2 3 4 5 6 7 8 9
A B C D E F G H I J K L M N O P Q R S T U V W X Y
0 1 2 3 4 5 6 7 8 9

Fig 5: NPVD Training Data

BIBLIOGRAPHY

Reference

1. J-Y.Bouguet. Pyramidal Implementation of the Lucas-Kanade Feature Tracker. OpenCV Documentation, Microprocessor Research Lab, Intel Corporation, 1999.

2. R.C. Gonzalez and R.E.Wood. Digital Image Processing. Prentice Hall, 1993.

3. G.Ritter, J.Wilson. Computer Vision. Algorithms in Image Algebra. CRC Press, 1996.

Internet sites

1. OpenCV Tutorial, https://www.tutorialspoint.com/opencv/index.html

2. Python Tkinter Tutorial, https://www.python-course.eu/python_tkinter.php

3. Video Tutorial on Number Plate Recognition, https://www.youtube.com/watch?v=fJcl6Gw1D8k