INTRODUCTION

As the number of vehicles has been increased with considerable amount during the recent years, more and more attention is to be paid on advanced, efficient and accurate intelligent transportation systems (ITSs). One of the important techniques used in ITS is Automobile Number Plate Detection System. ANPD is a computer vision technology that recognizes the vehicle's number plate without direct human intervention. This system extracts the characters of the number plate from the captured images of the vehicle, which can be further used for various purpose. Hence, it can be used in traffic management applications like automatic gate control for authorized/non-authorized vehicles, entrance admissions in toll systems, monitoring of traffic violations, borders crossing control and premises where high security is needed, such as the Parliament building, etc.

REALISATION OF THE PROBLEM

Nowadays, prime focus is on security for which Surveillance cameras are installed everywhere to record the ongoing activities. These recording are used whenever any suspicious activity or any accident occurs. Main purpose of these camera is to record daily 24/7 videos which can be used as an evidence for any crime. These recordings are mainly used by police for their investigation purpose. These cameras are installed at many places such as street corners, at house entrance, parking's, tolls etc. Now there is a need to extend this feature of surveillance camera to start extracting images from videos and identifying license plate of vehicle involved in crime, so that these extracted license plate number can be directly used for finding the owner of the vehicle.

OBJECTIVES

This system will be used to scan digital images of number-plates of vehicles and convert the displayed digits and alphabets into text on a computer. It can be used to identify vehicles by their number-plates without human intervention. Real-life situations where such technology can be employed include, but are not limited to, toll collection, traffic management, general surveillance, and keeping track of movement of vehicles in premises where security is paramount.

PROBLEM FORMULATION

Problem Definition-

In present scenario prime focus is on security for which Surveillance cameras are installed everywhere to record the ongoing activities. These recording are used whenever any suspicious activity or any accident occurs. Main purpose of these camera is to record daily 24/7 videos which can be used as an evidence for any crime. These recordings are mainly used by police for their investigation purpose. These cameras are installed at many places such as street corners, at house entrance, parking's, tolls etc.

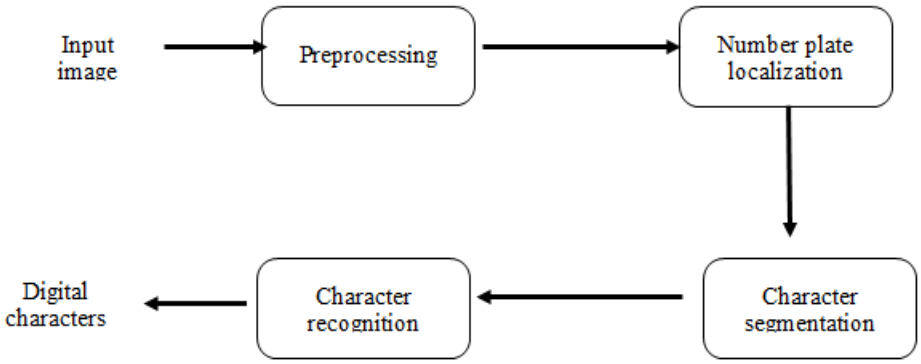
Now there is a need to extend this feature of surveillance camera to start extracting images from videos and identifying license plate of vehicle involved in crime, so that these extracted license plate number can be directly used for finding the owner of the vehicle.

Proposed System-

The proposed system will scan a digital image of a vehicle number plate and extract the digits and alphabets from that image. The input shall be an image of a vehicle number plate stored on the computer, and the output will be that number plate converted into text.

The proposed system consists of following steps-

- Preprocessing
- Localization of number plate
- Character Segmentation
- Character Recognition



REQUIREMENT SPECIFICATION

Performance Requirement-

Performance of the application will highly depend upon the performance of hardware and software components of the computer. Coming to timing relationships, the load time for user interface screens should not take longer than a seconds.

Safety Requirement-

- Application will not cause any harm to human users.
- Application will be restored in case of any emergency

Security Requirement-

Application is designed in such a way that it cannot use file of some other application.

Different modules work in co-ordination with other modules, they can't work independently.

FEASIBILITY STUDY

The prime focus of the feasibility is evaluating the practicality of the proposed system keeping in mind a number of factors. The following factors are taken into account before deciding in favour of the new system. It is of three types:

- Economic Feasibility
- Technical Feasibility
- Operational feasibility

Economic Feasibility-

A project is considered economically feasible when the benefits are greater than the cost of undertaking the project. This project is going to be economically feasible as it is going to reduce the manual work done on paper thereby reducing the cost incurred in paper.

Technical feasibility-

The technical feasibility study describes the details of how you will deliver a product or service i.e. what technology and expertise are required.

Keeping in view the above fact, nowadays all organizations are automating the repetitive and monotonous works done by humans. The key process areas of the current system are nicely amenable to automation and hence the technical feasibility is proved beyond doubt.

Operational Feasibility-

The present system has automated most of the manual tasks. Therefore, the proposed system will increase the operational efficiency of the transport system. This system is a detection system, which needs small amount of resources. Department of transportation or security can meet the conditions both in hardware and software aspects; therefore, this system is feasible in operation.

SYSTEM ANALYSIS AND DESIGN

Description Of Module-

- **USER**

  When user login to the system then following activities can be performed-

  1. Can upload image.
  2. Get License Number of the vehicle.

SYSTEM IMPLEMENTATION

Hardware Specifications-

- Processor: Intel Core I3 or higher power
- RAM: 2 GB or more
- Hard disk: Around 1 GB or more
- Keyboard: Normal or Multimedia
- Mouse: Compatible Mouse

Software Specifications-

- Operating System: This application can run on Linux, Windows and Mac or any other OS in which "OpenCV" (a python library) is preinstalled.

RESULT AND DISCUSSION

Future Scope Of the Project-

"AUTOMOBILE NUMBER PLATE DETECTION SYSTEM" is an application which has a wide range of scope. This project was developed to fulfil the requirements, however there are lots of scope to improve the performance of the system in the area of user interface, data set, etc.

Therefore, there are many features for future enhancement of this project. One of the future enhancements of the project are as follows-

- It may be integrated with other software's and applications to provide additional functionalities and features. For instance, the extracted text may be sent to another application that has a database of number-plates, and it may be matched with that database for vehicle identification or tracking.

Importance of Work -

The modern technology has proved that machine is much efficient and reliable than man. The computer has made great revolution in the world and we can say that it is the best creation of the man in the 20^{th} century. The computer gets the popularity due to its fast processing of data, reliability, and storage capacity. In case of number plate detection, the system eliminates the human errors by automatically detecting the number without human intervention.

Advantages And Special Features Of The System-

Today, the world is running on technology. Everything is running on technology as from shopping to e-learning, everything goes digital. Similarly, Security and surveillance are the sectors that needed the intervention of advanced technology.

This removes the hindrance of wasting and efforts in detecting the number of the vehicle which may even be predicted wrongly due to human errors.

Hence the system is used in traffic management applications like automatic gate control for authorized/non-authorized vehicles, entrance admissions in toll systems, monitoring of traffic violations, borders crossing control and premises where high security is needed, such as the Parliament building, etc.

Limitations -

No software in the world can be called perfect as it has some or the other flaw or lacking, that is new version keeps on pouring. Similarly, this system also possess some limitations such as this system will not work for images in dark.

CONCLUSION

The project "Automobile Number Plate Detection System" is completed, satisfying the required design specifications. The system provides a user-friendly interface. The software is developed with modular approach. All modules in the system have been tested with valid data and invalid data and everything work successfully. Thus, the system has fulfilled all the objectives identified and is able to replace the existing system. The constraints are met and overcome successfully. The system is designed as like it was decided in the design phase. This software has a user-friendly screen that enables the user to use without any inconvenience.

The software is a stand-alone, self-contained program and is not a part of a product family.

REFERENCE/BIBLIOGRAPHY

1. Belal R. Mohamed, Hala M Abd El Kader, Hanaa M. Rafaat, Mohamed S. Sharaf, "AUTOMATIC NUMBER PLATE RECOGNITION", International Journal of Scientific and Research Publications, 2013.
2. Horge Jorgensen, "Automatic License Plate Recognition using Deep Learning Techniques",2017.

APPENDIX

Data flow diagram-

A Data Flow Diagram (DFD) is a diagram that describes the flow of data and the processes that change data throughout a system. It's a structured analysis and design tool that can be used for flowcharting in place of or in association with information oriented and process oriented system flowcharts. When analysts prepare the Data Flow Diagram, they specify the user needs at a level of detail that virtually determines the information flow into and out of the system and the required data resources. This network is constructed by using a set of symbols that do not imply physical implementations. The Data Flow Diagram reviews the current physical system, prepares input and output specification, specifies the implementation plan etc.

Four basic symbols are used to construct data flow diagrams. They are symbols that represent data source, data flows, and data transformations and data storage. The points at which data are transformed are represented by enclosed figures, usually circles, which are called nodes.

The Data flow Diagram shows the flow of data. It is generally made of symbols given below:-

A square shows the Entity.

A Circle shows the Process

An open Ended Rectangle shows the data store.

An arrow shows the data flow.

The DFD can be up to several levels. The 0 level DFD states the flow of data in the system as seen from the outward in each module.

The first level DFD show more detail, about the single process of 0 levels DFD

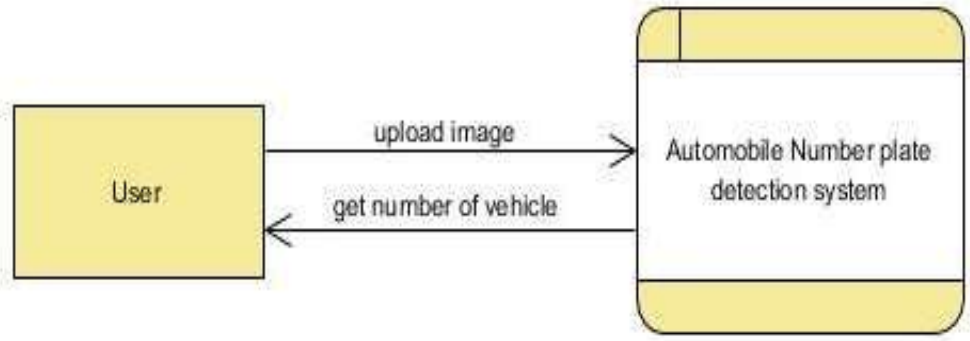


FIG: ZERO LEVEL DFD OF ANPD

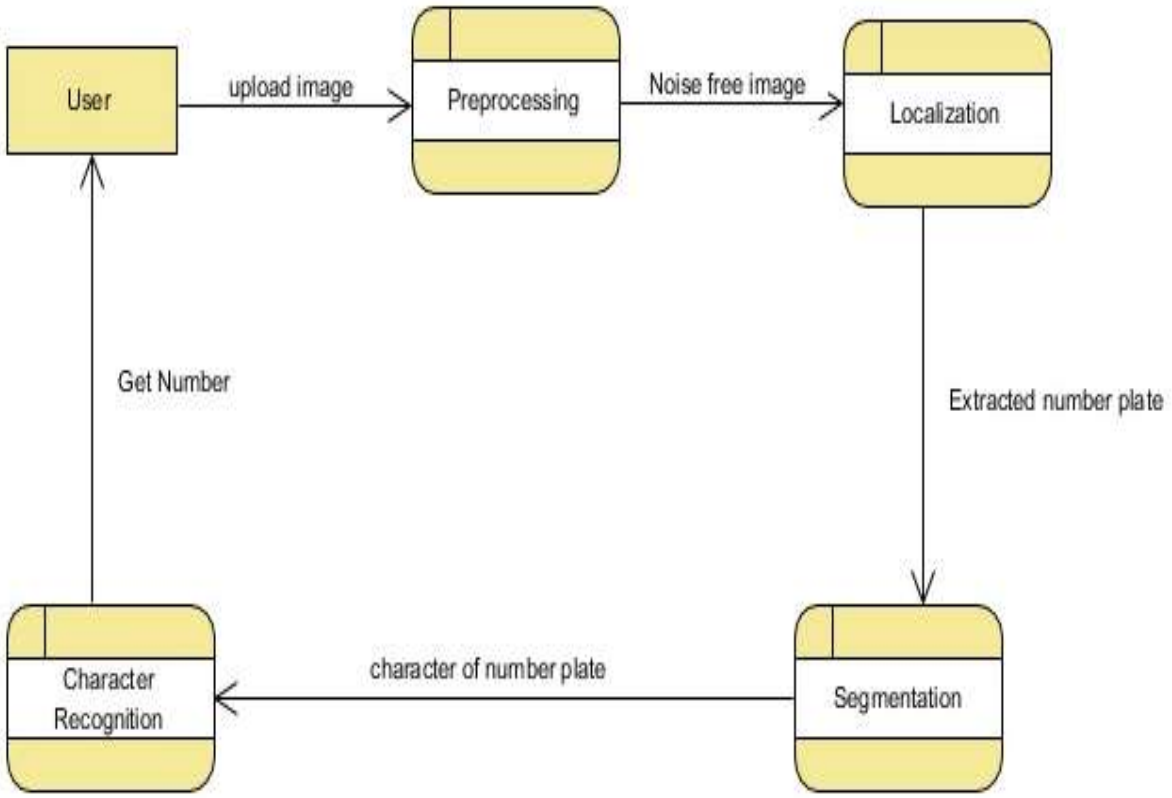


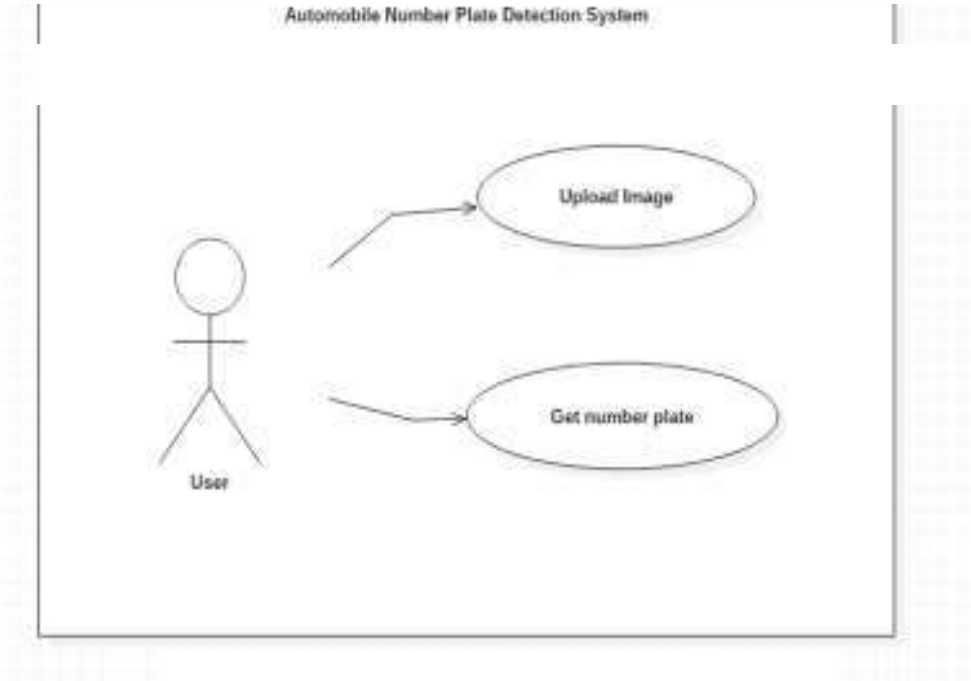
FIG: ONE LEVEL DFD ANPD

USE CASE:

As the most known diagram type of the behavioural UML diagrams, Use case diagrams give a graphic overview of the actors involved in a system, different functions needed by those actors and how these different functions interact.

It's a great starting point for any project discussion because you can easily identify the main actors involved and the main processes of the system.

USE CASE:



SAMPLE CODING

```
import cv2
import numpy as np
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.neural_network import MLPClassifier
import matplotlib.pyplot as plt

from tkinter import *
from PIL import ImageTk,Image
from tkinter import filedialog
```

```
root=Tk()
root.geometry("1000x1000")
#root.minsize(600,500)
labelq = Label(root, text= "ANDS SOFTWARE" , font='Helvetica 17 ',borderwidth=5, relief="groove")
labelq.pack()

labelq.place(x=0,y=0)
root.configure(background="#FFE4B5")
def open():

    global my_image
    global img

    img=root.filename=filedialog.askopenfilename(title="select a file",filetype=(("jpeg","*.jpg"),("All
Files","*.*")))
    my_label=Label(root,text=root.filename).pack()
    my_image=ImageTk.PhotoImage(Image.open(root.filename))
    my_image_label=Label(image=my_image).pack()

    print(img)
    imput=img
    img = cv2.imread(imput,0)
    img = cv2.bilateralFilter(img,5,75,75)

    clahe = cv2.createCLAHE(clipLimit=2.0, tileGridSize=(8,8))########  for images in analysis.txt
    img = clahe.apply(img)

    _,img_dilation = cv2.threshold(img,120,255,cv2.THRESH_BINARY)
    kernel=np.ones((3,3), np.uint8)####
    img_dilation= cv2.erode(img_dilation, kernel, iterations=1)
```

```
laplacian=cv2.Sobel(img_dilation,cv2.CV_64F,1,0,ksize=-1)



height, width=laplacian.shape
count=0
temp=0
i=(height//2)-50
j=100
x1=0
x2=0
set_height=50      ###set ht and width of travesing rectangle
set_width=215
while i<height:
    #print "i=",i
    if height-i<set_height:
        #print "i ",i
        break
    j=100
    while j<width:
        if width-j<set_width+100:
            #print j
            break
        count=0
        j1=j
        i1=i
        count=np.count_nonzero(laplacian[i1:i1+set_height//4,j1:j1+set_width])
        if count<temp//4:
            j+=4
```

```
            continue
        #count=count+np.count_nonzero(laplacian[i1+set_height/4:i1+set_height,j1:j1+set_width])
        count=np.count_nonzero(laplacian[i1:i1+set_height,j1:j1+set_width])
        if temp<count:
            #print "temp=",temp," "
            temp=count
            x1,x2=i,j
        j+=4  ##############################
    i+=3   ################################
#print "number plate found ---coordinates---",x1,"--",x2,"--",temp###x1 x2 coordinates of top left of number plate


c_img=img_dilation[x1:x1+set_height, x2:x2+set_width]  ###
#output_file_name=str(j1111)+'.jpg'
#cv2.imwrite(output_file_name,c_img)
#string=str(counter_i)+'x'#### image is in the form of (number alphabet)(eg 29y, 23z)
img = c_img#cv2.imread(string+".jpg",0)
#org = cv2.imread(string+".jpg")
#image_cropped = cv2.imread(string+".jpg")
#print "working on->"+string
height, width = img.shape
org=np.zeros((height,width,1),np.uint8)
i=0
j=0
while i<height:
    while j<width:
        org[i][j]=img[i][j]
        j+=1
    i+=1
kernel = np.ones((3,3), np.uint8)####
```

```
img = cv2.dilate(img, kernel, iterations=1)

img = cv2.erode(img, kernel, iterations=1)

sobelx = cv2.Sobel(img,cv2.CV_8U,1,0,ksize=-1)
#cv2.imshow('sobelx',sobelx)

sobely = cv2.Sobel(img,cv2.CV_8U,0,1,ksize=-1)

#cv2.imshow('y',sobely)
_,sobelx = cv2.threshold(sobelx,128, 255, cv2.THRESH_BINARY)
_,sobely = cv2.threshold(sobely,128, 255, cv2.THRESH_BINARY)

i=0
count=0
ar=[]
xar=[]

i=0
j=0
while i<width:
    j=0
    count=0
    while j<height:
        if sobelx[j][i]==255:
            count+=1
        j+=1
    ar.append(count)
    i+=1
```

```
prev_val_stored=0
first_val_stored=0
count=0
i=0
j=0
k=0
tall=0
first_point=0
first_val=0
prev_val=0
val=0
after=49
last_point=0
first_point_var=0
while i<height:
    j=0
    count=0
    while j<width:
        if sobelx[i][j]==255:
            count+=1
        j+=1
    xar.append(count)
    val=count
    if i==0:
        first_val=count
        prev_val=count
        first_point_var=i
    else:
        if val<prev_val:
```

```
            temp=prev_val-first_val
            if temp>tall:
                tall=temp
                prev_val_stored=prev_val
                first_val_stored=first_val
                last_point=i-1
                first_point=first_point_var
            k=0
        if val>prev_val:
            if k==0:
                k=1
                first_val=prev_val
                first_point_var=i-1
        prev_val=val
    i+=1
#print "first point->",first_point," start->",first_val_stored,"last point->",last_point," end-
>",prev_val_stored
xx=first_point
yy=last_point


prev_val_stored=0
first_val_stored=0
count=0
i=0
j=0
k=0
tall=0
first_point=0
first_val=0
prev_val=0
```

```
val=0
after=49
last_point=0
first_point_var=0

while i<height:
    j=0
    count=0
    while j<width:
        if sobelx[i][j]==255:
            count+=1
        j+=1
    #xar.append(count)
    val=count
    if i==0:
        first_val=count
        prev_val=count
        first_point_var=i
    else:
        if val<prev_val:
            temp=prev_val-first_val
            if temp>tall and first_point_var!=xx and yy!=i-1:
                tall=temp
                prev_val_stored=prev_val
                first_val_stored=first_val
                last_point=i-1
                first_point=first_point_var
            k=0
        if val>prev_val:
            if k==0:
```

```
                k=1
                first_val=prev_val
                first_point_var=i-1
            prev_val=val
        i+=1
    #print "first point->",first_point," start->",first_val_stored,"last point->",last_point," end-
>",prev_val_stored


    prev_val_stored=0
    first_val_stored=0
    count=0
    i=height-1
    j=0
    k=1
    tall=0
    first_point1=0
    first_val=0
    prev_val=0
    val=0
    after=49
    last_point1=0
    first_point_var=0
    while i>=0:
        j=0
        count=0
        while j<width:
            if sobelx[i][j]==255:
                count+=1
            j+=1
        #xar.append(count)
```

```
        val=count
        if i==height-1:
            first_val=count
            prev_val=count
            first_point_var=i
        else:
            if val<prev_val:
                temp=prev_val-first_val
                if temp>tall:
                    tall=temp
                    prev_val_stored=prev_val
                    first_val_stored=first_val
                    if i!=0:
                        last_point1=i+1
                    else:
                        last_point1=0
                    first_point1=first_point_var
                k=0
            if val>prev_val:
                if k==0:
                    k=1
                    first_val=prev_val
                    first_point_var=i+1
            prev_val=val
        i-=1
    #print "first point->",first_point1," start->",first_val_stored,"last point->",last_point1," end-
>",prev_val_stored
    xx1=first_point1
    yy1=last_point1
```

```
prev_val_stored=0
first_val_stored=0
count=0
i=height-1
j=0
k=1
tall=0
first_point1=0
first_val=0
prev_val=0
val=0
after=49
last_point1=0
first_point_var=0
while i>=0:
    j=0
    count=0
    while j<width:
        if sobelx[i][j]==255:
            count+=1
        j+=1
    #xar.append(count)
    val=count
    if i==height-1:
        first_val=count
        prev_val=count
        first_point_var=i
    else:
        if val<prev_val:
            temp=prev_val-first_val
```

```
            if temp>tall and first_point_var!=xx1 and yy1!=i+1:
                tall=temp
                prev_val_stored=prev_val
                first_val_stored=first_val
                if i!=0:
                    last_point1=i+1
                else:
                    last_point1=0
                first_point1=first_point_var
            k=0
        if val>prev_val:
            if k==0:
                k=1
                first_val=prev_val
                first_point_var=i+1
        prev_val=val
    i-=1
lp1=last_point1
lp=last_point
fp1=first_point1
fp=first_point
starting1=0
ending1=0


if yy1-yy<last_point1-last_point and ((yy>lp and yy>lp1 and yy1>lp and yy1>lp1)or (yy<lp and
yy<lp1 and yy1<lp and yy1<lp1)):
    if (lp+fp)//2<lp-3:
        var=lp-(lp+fp)//2
    else:
```

```
        var=3
    if (lp1+fp1)//2<lp1+3:
        var1=(lp1+fp1)//2-lp1
    else:
        var1=3
    starting=(last_point-var)
    ending=(last_point1+var1)
    starting1=(((fp+lp)//2)+fp)//2#(last_point-var)
    ending1=(((fp1+lp1)//2)+fp1)//2#(last_point1+var1
else:
    if (xx+yy)//2<yy-3:
        var=yy-(xx+yy)//2
    else:
        var=3
    if (xx1+yy1)//2<yy1+3:
        var1=(xx1+yy1)//2-yy1
    else:
        var1=3
    starting=(yy-var)
    ending=(yy1+var1)
    starting1=(((xx+yy)//2)+xx)//2#(last_point-var)
    ending1=(((xx1+yy1)//2)+xx1)//2#(last_point1+var1


if starting>=ending:###no image sorry brahhh
    #j1111=j1111+1
    #continue
    x=1

if starting<0:
```

```
    starting=0
if ending>=height:
    ending=height-1
#print starting,"--->",ending



heightss, widthss = img.shape
c_img=img[starting:ending,0:0+widthss]  ###

img=img[starting1:ending1,0:0+widthss]
heightss, widthss = img.shape

img = cv2.cvtColor(img,cv2.COLOR_GRAY2RGB)
#cv2.imshow('img',img)



kernel = np.ones((3,3), np.uint8)
c_img = cv2.dilate(c_img, kernel, iterations=1)
_,c_img = cv2.threshold(c_img,128, 255, cv2.THRESH_BINARY)
#cv2.imshow('c_img',c_img)

heightz,widthz= c_img.shape
#print heightz,"--->",widthz
xary=[]
count1=0
i=0
j=0



while i<widthz:
```

```
        j=0
        count1=0
        while j<heightz:
            if c_img[j][i]==0:
                count1+=1
            j+=1
        xary.append(count1)
        i+=1


#print "first point->",first_point," start->",first_val_stored,"last point->",last_point," end-
>",prev_val_stored
    #cv2.imshow('here it is',c_img)
#print xary[0]," ",xary[1]," ",xary[2]
    varia_returns=0
    i=0
    j=0
    k=0
    z=0
    ct=0
    seg_img=[]
    variable=0
    lit=[]



    org=img
    while i<widthz:
        #print "s"
        if (xary[i]<3 and k==0):
            k=1
            z=i
```

```
        while xary[z]<3:
            z+=1
        i=z
        if i!=0:
            j=i-1
        else:
            j=i
        #continue
    elif xary[i]<3 and k==1:

        variable=0
        ia=i
        ja=j
        if j>=2:
            ja=j-2
        if i<=(width-1)-2:
            ia=i+2

        lit.append(org[0:heightss,ja:ia])
        #cv2.imwrite(k1111+str(varia)+'.jpg',org[0:heightss,ja:ia]);
        #varia+=1
        ct+=1
        z=i
        while z<widthz and xary[z]<3:
            z+=1
        i=z
        j=i-1
        continue
    i+=1
j=0
```

```
i=0

while i<len(lit):
    lit[i]=cv2.cvtColor(lit[i],cv2.COLOR_BGR2GRAY)
    #cv2.imshow('haha',lit[i])
    #cv2.waitKey(0)
    #cv2.destroyAllWindows()
    i+=1

train=[]
i=1
j=0
while j<=9:
    i=1
    while i<=27:
        k=str(j)+chr(96+i)+".jpg"
        name=''.join(k)
        img=cv2.imread(name,0)
        img=cv2.resize(img, (20, 20))
        a,img=cv2.threshold(img,128,255,cv2.THRESH_BINARY)
        train.append(img)
        i+=1
    j+=1

i=1
while i<=29:
    k='a'+str(i)+".jpg"
    name=''.join(k)
    img=cv2.imread(name,0)
    img=cv2.resize(img, (20, 20))
```

```
    a,img=cv2.threshold(img,128,255,cv2.THRESH_BINARY)
    train.append(img)
    i+=1


i=1
while i<=23:
    k='b'+str(i)+".jpg"
    name=''.join(k)
    img=cv2.imread(name,0)
    img=cv2.resize(img, (20, 20))
    a,img=cv2.threshold(img,128,255,cv2.THRESH_BINARY)
    train.append(img)
    i+=1

i=1
while i<=26:
    k='c'+str(i)+".jpg"
    name=''.join(k)
    img=cv2.imread(name,0)
    img=cv2.resize(img, (20, 20))
    a,img=cv2.threshold(img,128,255,cv2.THRESH_BINARY)
    train.append(img)
    i+=1

i=1
while i<=16:
    k='d'+str(i)+".jpg"
    name=''.join(k)
    img=cv2.imread(name,0)
```

```
    img=cv2.resize(img, (20, 20))
    a,img=cv2.threshold(img,128,255,cv2.THRESH_BINARY)
    train.append(img)
    i+=1

i=1
while i<=16:
    k='e'+str(i)+".jpg"
    name=''.join(k)
    img=cv2.imread(name,0)
    img=cv2.resize(img, (20, 20))
    a,img=cv2.threshold(img,128,255,cv2.THRESH_BINARY)
    train.append(img)
    i+=1

i=1
while i<=13:
    k='f'+str(i)+".jpg"
    name=''.join(k)
    img=cv2.imread(name,0)
    img=cv2.resize(img, (20, 20))
    a,img=cv2.threshold(img,128,255,cv2.THRESH_BINARY)
    train.append(img)
    i+=1

i=1
while i<=41:
    k='g'+str(i)+".jpg"
    name=''.join(k)
    img=cv2.imread(name,0)
```

```
    img=cv2.resize(img, (20, 20))
    a,img=cv2.threshold(img,128,255,cv2.THRESH_BINARY)
    train.append(img)
    i+=1

i=1
while i<=15:
    k='h'+str(i)+".jpg"
    name=''.join(k)
    img=cv2.imread(name,0)
    img=cv2.resize(img, (20, 20))
    a,img=cv2.threshold(img,128,255,cv2.THRESH_BINARY)
    train.append(img)
    i+=1

i=1
while i<=17:
    k='i'+str(i)+".jpg"
    name=''.join(k)
    img=cv2.imread(name,0)
    img=cv2.resize(img, (20, 20))
    a,img=cv2.threshold(img,128,255,cv2.THRESH_BINARY)
    train.append(img)
    i+=1

i=1
while i<=9:
    k='j'+str(i)+".jpg"
    name=''.join(k)
    img=cv2.imread(name,0)
```

```
    img=cv2.resize(img, (20, 20))
    a,img=cv2.threshold(img,128,255,cv2.THRESH_BINARY)
    train.append(img)
    i+=1

i=1
while i<=26:
    k='k'+str(i)+".jpg"
    name=''.join(k)
    img=cv2.imread(name,0)
    img=cv2.resize(img, (20, 20))
    a,img=cv2.threshold(img,128,255,cv2.THRESH_BINARY)
    train.append(img)
    i+=1

i=1
while i<=8:
    k='l'+str(i)+".jpg"
    name=''.join(k)
    img=cv2.imread(name,0)
    img=cv2.resize(img, (20, 20))
    a,img=cv2.threshold(img,128,255,cv2.THRESH_BINARY)
    train.append(img)
    i+=1

i=1
while i<=17:
    k='m'+str(i)+".jpg"
    name=''.join(k)
    img=cv2.imread(name,0)
```

```
    img=cv2.resize(img, (20, 20))
    a,img=cv2.threshold(img,128,255,cv2.THRESH_BINARY)
    train.append(img)
    i+=1

i=1
while i<=13:
    k='n'+str(i)+".jpg"
    name=''.join(k)
    img=cv2.imread(name,0)
    img=cv2.resize(img, (20, 20))
    a,img=cv2.threshold(img,128,255,cv2.THRESH_BINARY)
    train.append(img)
    i+=1

i=1
while i<=12:
    k='p'+str(i)+".jpg"
    name=''.join(k)
    img=cv2.imread(name,0)
    img=cv2.resize(img, (20, 20))
    a,img=cv2.threshold(img,128,255,cv2.THRESH_BINARY)
    train.append(img)
    i+=1

i=1
while i<=20:
    k='r'+str(i)+".jpg"
    name=''.join(k)
    img=cv2.imread(name,0)
```

```
    img=cv2.resize(img, (20, 20))
    a,img=cv2.threshold(img,128,255,cv2.THRESH_BINARY)
    train.append(img)
    i+=1

i=1
while i<=23:
    k='s'+str(i)+".jpg"
    name=''.join(k)
    img=cv2.imread(name,0)
    img=cv2.resize(img, (20, 20))
    a,img=cv2.threshold(img,128,255,cv2.THRESH_BINARY)
    train.append(img)
    i+=1

i=1
while i<=14:
    k='t'+str(i)+".jpg"
    name=''.join(k)
    img=cv2.imread(name,0)
    img=cv2.resize(img, (20, 20))
    a,img=cv2.threshold(img,128,255,cv2.THRESH_BINARY)
    train.append(img)
    i+=1

i=1
while i<=16:
    k='u'+str(i)+".jpg"
    name=''.join(k)
    img=cv2.imread(name,0)
```

```
    img=cv2.resize(img, (20, 20))
    a,img=cv2.threshold(img,128,255,cv2.THRESH_BINARY)
    train.append(img)
    i+=1

i=1
while i<=17:
    k='v'+str(i)+".jpg"
    name=''.join(k)
    img=cv2.imread(name,0)
    img=cv2.resize(img, (20, 20))
    a,img=cv2.threshold(img,128,255,cv2.THRESH_BINARY)
    train.append(img)
    i+=1

i=1
while i<=28:
    k='z'+str(i)+".jpg"
    name=''.join(k)
    img=cv2.imread(name,0)
    img=cv2.resize(img, (20, 20))
    a,img=cv2.threshold(img,128,255,cv2.THRESH_BINARY)
    train.append(img)
    i+=1


train=np.array(train)
train=train.reshape(-1,400).astype(np.float32)

l=np.arange(10)
```

```
    label1=np.repeat(l,27)


label2=(31,31,31,31,31,31,31,31,31,31,31,31,31,31,31,31,31,31,31,31,31,31,31,31,31,31,31,31,32,32,
32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,33,33,33,33,33,33,33,33,33,33,33,33,
33,33,33,33,33,33,33,33,33,33,33,33,33,33,34,34,34,34,34,34,34,34,34,34,34,34,34,34,34,34,35,35,35,35,35,
35,35,35,35,35,35,35,35,35,35,35,36,36,36,36,36,36,36,36,36,36,36,36,36,36,37,37,37,37,37,37,37,37,37,37,
37,37,37,37,37,37,37,37,37,37,37,37,37,37,37,37,37,37,37,37,37,37,37,37,37,37,37,37,37,37,37,38,38,38,
38,38,38,38,38,38,38,38,38,38,38,38,39,39,39,39,39,39,39,39,39,39,39,39,39,39,39,39,39,39,40,40,40,40,40,
40,40,40,40,41,41,41,41,41,41,41,41,41,41,41,41,41,41,41,41,41,41,41,41,41,41,41,41,41,42,42,42,42,
42,42,42,42,43,43,43,43,43,43,43,43,43,43,43,43,43,43,43,43,43,43,43,44,44,44,44,44,44,44,44,44,44,44,44,
45,45,45,45,45,45,45,45,45,45,45,45,46,46,46,46,46,46,46,46,46,46,46,46,46,46,46,46,46,46,46,46,47,47,
47,47,47,47,47,47,47,47,47,47,47,47,47,47,47,47,47,47,47,47,47,47,48,48,48,48,48,48,48,48,48,48,48,48,48,
48,49,49,49,49,49,49,49,49,49,49,49,49,49,49,49,49,50,50,50,50,50,50,50,50,50,50,50,50,50,50,50,50,
51,51,51,51,51,51,51,51,51,51,51,51,51,51,51,51,51,51,51,51,51,51,51,51,51,51,51,51,51,)

    label2=np.array(label2)


    labels=np.concatenate((label1,label2),axis=0)

    labels=labels[:,np.newaxis]


    if 1==1:


        knn=cv2.ml.KNearest_create()

        knn.train(train, cv2.ml.ROW_SAMPLE, np.ravel(labels))


        forest = RandomForestClassifier(n_estimators = 130)#


        print("training Random Forest model")

        forest = forest.fit( train, np.ravel(labels))#



    outp=[]


    for test in lit:

        ht,wd=test.shape
```

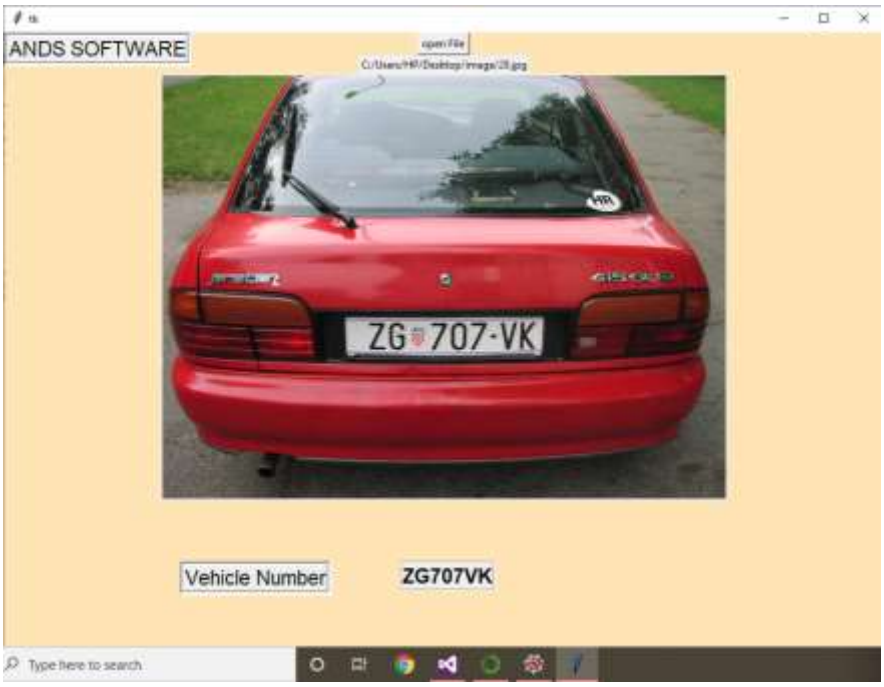
```
if ht<15:
    continue

a,test=cv2.threshold(test,128,255,cv2.THRESH_BINARY)
i=0
j=0
count=0
while i<ht:
    j=0
    while j<wd:
        if test[i][j]==0:
            count+=1
        j+=1
    i+=1

if count<95:
    continue

test=cv2.resize(test, (20, 20))
a,test=cv2.threshold(test,128,255,cv2.THRESH_BINARY)

test=np.array(test)
test=test.reshape(-1,400).astype(np.float32)

result = forest.predict(test)
ret,result,neighbours,dist=knn.findNearest(test,k=3)
```

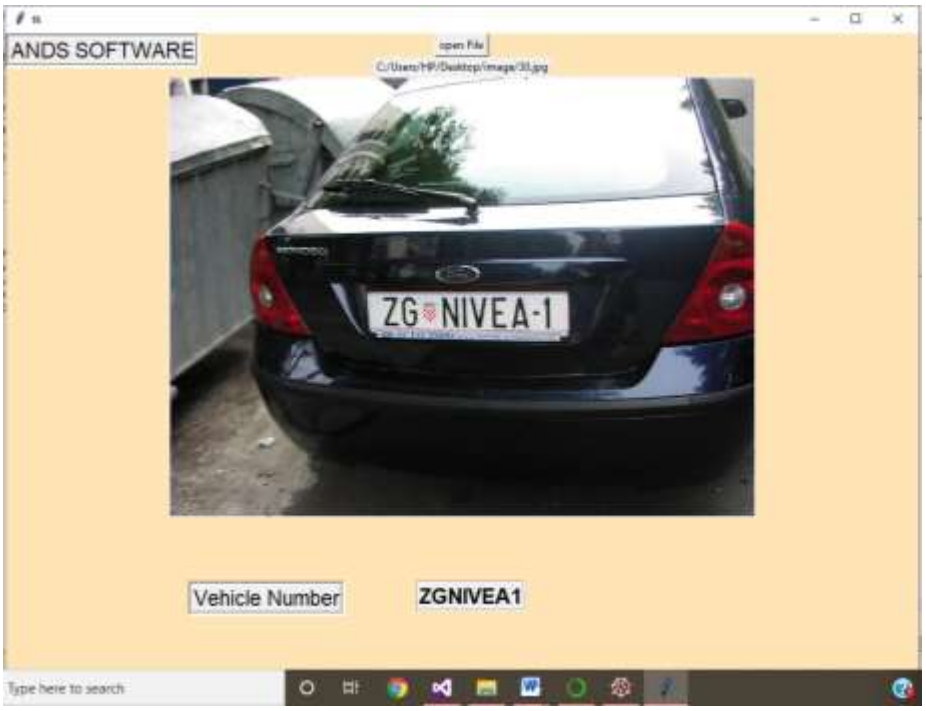
```
        if (dist[0][0]>5500000 and dist[0][1]>5500000) or (dist[0][0]>5500000 and dist[0][2]>5500000) or
(dist[0][1]>5500000 and dist[0][2]>5500000):
            continue
        else:
            if result[0][0]==31.0:
                outp.append('A')
            elif result[0][0]==32.0:
                outp.append('B')
            elif result[0][0]==33.0:
                outp.append('C')
            elif result[0][0]==34.0:
                outp.append('D')
            elif result[0][0]==35.0:
                outp.append('E')
            elif result[0][0]==36.0:
                outp.append('F')
            elif result[0][0]==37.0:
                outp.append('G')
            elif result[0][0]==38.0:
                outp.append('H')
            elif result[0][0]==39.0:
                outp.append('I')
            elif result[0][0]==40.0:
                outp.append('J')
            elif result[0][0]==41.0:
                outp.append('K')
            elif result[0][0]==42.0:
                outp.append('L')
            elif result[0][0]==43.0:
                outp.append('M')
```

```
        elif result[0][0]==44.0:
            outp.append('N')
        elif result[0][0]==45.0:
            outp.append('P')
        elif result[0][0]==46.0:
            outp.append('R')
        elif result[0][0]==47.0:
            outp.append('S')
        elif result[0][0]==48.0:
            outp.append('T')
        elif result[0][0]==49.0:
            outp.append('U')
        elif result[0][0]==50.0:
            outp.append('V')
        elif result[0][0]==51.0:
            outp.append('Z')
        else:
            outp.append(int(result[0][0]))

#print("RF  "+outp)
out2= ''.join(map(str,outp))

#global outstring

outstring = out2

print("NN  "+out2)

label = Label(root, text= outstring, borderwidth=2, font='Helvetica 17 bold',relief="groove")
label.pack()
```

```
    label.place(x=450,y=600)

    labelw = Label(root, text= "Vehicle Number" , font='Helvetica 17 ',borderwidth=5, relief="groove")

    labelw.pack()

    labelw.place(x=200,y=600)


my_btn=Button(root,text="open File",command=open).pack()

root.mainloop()
```

OUTPUT SCREEN




ANDS SOFTWARE
open File
ZG NIVEA-1
Vehicle Number
ZGNIVEA1
Type here to search