VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

November 2018

Number Plate Recognition

(Using Python and Open CV)

A Project Report

Under the Guidance of,

Prof. A.Srivani

By

17BCE0597 - Shashwat Negi

17BCE2152 - Vibhor Sharma

17BCE2143 - Divyanshu Garg

DECLARATION BY THE CANDIDATE

We hereby declare that the project report entitled "Number Plate Recognition" submitted by us to Vellore Institute of Technology, Vellore in partial fulfilment of the requirement for the award of the degree of **B. Tech (CSE)** is a record of J- component of project work carried out by us under the guidance of **Prof. A.Srivani**. We further declare that the work reported in this project has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university.

Place : Vellore Institute of Technology, Vellore.
Date : 29/10/2018

Signature of the faculty Signature of the Candidate

1. Introduction

1.1 Abstract

Automatic number plate recognition (ANPR) is an image processing technology which uses license plate to identify the vehicle. The main goal here is to design an efficient automatic authorized vehicle identification system by using the particular vehicle's number plate. (because of the number being unique)

These systems are generally used on the entrance of highly restricted area like military zones or area around top government offices e.g. Parliament, Supreme Court etc. The developed system first detects the vehicle and then captures the vehicle image. Vehicle number plate region is then extracted using the image segmentation on the captured image. Optical character recognition technique is used for the character recognition. The resulting data is then used to compare with the records on a database so as to come up with the specific information like the vehicle's owner, place of registration, address, etc. The system which we worked on is implemented and simulated in Python, using OpenCV and it's performance is tested on real image. It is observed from the experiment that the developed system successfully detects and recognize the vehicle number plate on real images.

1.2 Background

The software aspect of the system runs on standard home computer hardware and can be linked to other applications or databases. It first uses a series of image manipulation techniques to detect, normalize and enhance the image of the number plate, and then optical character recognition (OCR) to extract the alphanumerics of the license plate. ANPR systems are generally deployed in one of two basic approaches: one allows for the entire process to be performed at the lane location in realtime, and the other transmits all the images from many lanes to a remote computer location and performs the OCR process there at some

later point in time. When done at the lane site, the information captured of the plate alphanumeric, date-time, lane identification, and any other information required is completed in approximately 250 milliseconds. This information can easily be transmitted to a remote computer for further processing if necessary, or stored at the lane for later retrieval. In the other arrangement, there are typically large numbers of PCs used in a server farm to handle high workloads. Often in such systems, there is a requirement to forward images to the remote server, and this can require larger bandwidth transmission media.

2. Overview and Planning

2.1 Proposed Work

In our project, what our system does is that from the image of the car provided, it first searches and finds out the number plate.

Then in the second step, it performs some steps to extract the characters written on the number plate and gives us the final output of the recognition of the characters on the number plate which helps us achieve our project goal.

2.2 Hardware Requirements

- Camera
- Infra-red
- Frame Grabber
- Computer

2.3 Software Requirements

- **Open CV**
- **Python**

3. Literature Survey and Review

3.1 Literature Summary

Automatic Number Plate Recognition is a process where vehicles are identified or recognized using their number plate or license plate. ANPR uses image processing techniques so as to extract the vehicle number plate from digital images [3].

ANPR systems normally comprises of two components: A camera that used in capturing of vehicle number plate images, and software that extracts the number plates from the captured images by using a character recognition tool that allows for pixels to be translated into

numerical readable characters [6]. A license plate recognition system generally works in four main parts namely image acquisition, license plate detection, characters segmentation, and lastly character recognition [7]. Fig 1 shows a typical ANPR process.

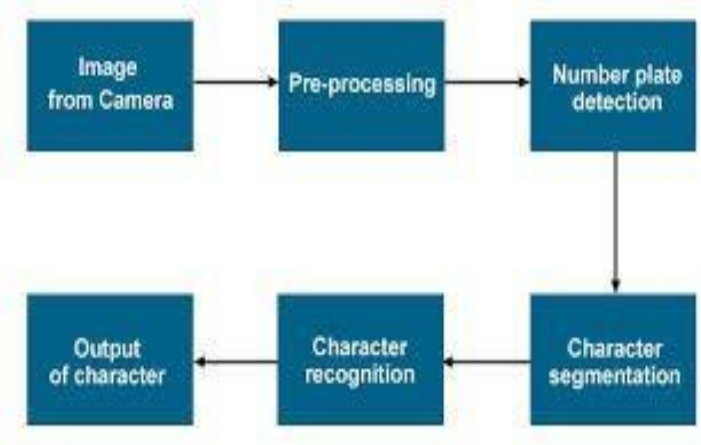

Fig 1: Typical ANPR Process [8]

. *Image Acquisition*

The first step is the image acquisition stage. The image of the vehicle is captured using a photographic camera. The constraint is that the image of the vehicle should be captured in such a way that the selected input image contains rear or front view of the vehicle with the number plate

Number Plate Detection

The next stage that follows is the number plate recognition phase that does several functions such as resizing of the image to a feasible aspect ratio. As well as converting the colored image into a grey scale, image.

Character Segmentation

Character segmentation can be defined as a technique, which partitions images of lines or words into individual characters. It is an operation that seeks to decompose an image of a sequence of character into sub images of individual symbols. Character segmentation is an operation that seeks to decompose an image of a sequence of characters into sub-images of individual symbols.

Character Recognition

Character recognition is process of detecting and recognizing characters from input image and converting it into meaningful text in ASCII (American Standard Code for Information Interchange) or other equivalent machine editable form [14]. Character recognition is the process to classify the input character according to the predefined character class.

Step1: The camera takes a picture of the vehicle containing the number plate (Image acquisition).

Step2: The camera isolates the plate, adjusts the brightness and contrast and segments it into characters (Number plate detection and Character segmentation).

Step3: The pattern of each character is analyzed to convert the picture into text (Character recognition).

4. Methodology

4.1 Method Used

i) Input image

We (the user) input a database of images which consist of license plates which need to be recognized.

It should be kept in mind that the image acquired should be specific to a country or state, as different geographical locations have different standards for the number plates on vehicles.

ii) Training the machine

The machine should be trained using multiple fonts of alphabets and numbers. This is done to make sure that the machine shows no errors to detect the license plate if there is the involvement of a different font.

iii) Plate localization – responsible for finding and isolating the plate on the picture.

In this part we recognize the part of the image consisting of the number plate. That part is then separated from the rest of the image using Image segmentation.

(the recognized plate is squared in red)

(it is then separated using Segmentation (possible plate))

iv) Normalization and Corrosion – adjusts the brightness and contrast of the image.

The image is initially normalized to adjust the brightness and contrast in order to make it suitable for the machine.

The image then undergoes grayscaling and is then eroded multiple times until the machine is clearly able to make out the outline of the characters in the number plate.

(gray scaled)

(eroded)

(further eroding and fine tuning the image (possible characters))

v) Character segmentation and optical character recognition – finds the individual characters on the plates.

After importing the PossiblePlates and PossibleCharacters once we have trained the KNN machine, it is able to make out the characters using the alphanumeric characters in which it was trained.

(finally recognizing the characters)

The complexity of each of these subsections of the program determines the accuracy of the system. During the fourth phase (normalization), our system uses edge detection technique to increase the picture difference between the letters and the plate backing and hence increases accuracy. We also use multiple median filters to reduce the visual noise on the image.

4.2 Applications

- Traffic signals
- Border crossings
- Automobile repossessions
- Recognize customers based on their license plate

5. System Implementation

5.1 Code

```
# Main.py

import cv2
import numpy as np
import os

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
###################################################################################################
def main():

    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

    imgOriginalScene  = cv2.imread("14.png")               # 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

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

```
    # end if else

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

    return
# end main

#################################################################################################
def drawRedRectangleAroundPlate(imgOriginalScene, licPlate):

    p2fRectPoints = cv2.boxPoints(licPlate.rrLocationOfPlateInScene)            # get 4 vertices of rotated
rect

    cv2.line(imgOriginalScene, tuple(p2fRectPoints[0]), tuple(p2fRectPoints[1]), SCALAR_RED, 2)         #
draw 4 red lines
    cv2.line(imgOriginalScene, tuple(p2fRectPoints[1]), tuple(p2fRectPoints[2]), SCALAR_RED, 2)
    cv2.line(imgOriginalScene, tuple(p2fRectPoints[2]), tuple(p2fRectPoints[3]), SCALAR_RED, 2)
    cv2.line(imgOriginalScene, tuple(p2fRectPoints[3]), tuple(p2fRectPoints[0]), SCALAR_RED, 2)
# end function

#################################################################################################
def writeLicensePlateCharsOnImage(imgOriginalScene, licPlate):
    ptCenterOfTextAreaX = 0                             # this will be the center of the area the text will
be written to
    ptCenterOfTextAreaY = 0

    ptLowerLeftTextOriginX = 0                          # this will be the bottom left of the area that the
text will be written to
    ptLowerLeftTextOriginY = 0

    sceneHeight, sceneWidth, sceneNumChannels = imgOriginalScene.shape
    plateHeight, plateWidth, plateNumChannels = licPlate.imgPlate.shape

    intFontFace = cv2.FONT_HERSHEY_SIMPLEX                      # choose a plain jane font
    fltFontScale = float(plateHeight) / 30.0                    # base font scale on height of plate area
    intFontThickness = int(round(fltFontScale * 1.5))           # base font thickness on font scale

    textSize, baseline = cv2.getTextSize(licPlate.strChars, intFontFace, fltFontScale, intFontThickness)
# call getTextSize

            # unpack roatated rect into center point, width and height, and angle
    ( (intPlateCenterX, intPlateCenterY), (intPlateWidth, intPlateHeight), fltCorrectionAngleInDeg ) =
licPlate.rrLocationOfPlateInScene

    intPlateCenterX = int(intPlateCenterX)              # make sure center is an integer
    intPlateCenterY = int(intPlateCenterY)

    ptCenterOfTextAreaX = int(intPlateCenterX)         # the horizontal location of the text area is the
same as the plate

    if intPlateCenterY < (sceneHeight * 0.75):                                                  # if the
license plate is in the upper 3/4 of the image
        ptCenterOfTextAreaY = int(round(intPlateCenterY)) + int(round(plateHeight * 1.6))      # write the
chars in below the plate
    else:                                                                                       # else if
the license plate is in the lower 1/4 of the image
        ptCenterOfTextAreaY = int(round(intPlateCenterY)) - int(round(plateHeight * 1.6))      # write the
chars in above the plate
    # end if

    textSizeWidth, textSizeHeight = textSize                # unpack text size width and height

    ptLowerLeftTextOriginX = int(ptCenterOfTextAreaX - (textSizeWidth / 2))           # calculate the lower
left origin of the text area
    ptLowerLeftTextOriginY = int(ptCenterOfTextAreaY + (textSizeHeight / 2))          # based on the text
area center, width, and height

            # write the text on the image
    cv2.putText(imgOriginalScene, licPlate.strChars, (ptLowerLeftTextOriginX, ptLowerLeftTextOriginY),
intFontFace, fltFontScale, SCALAR_YELLOW, intFontThickness)
# end function

#################################################################################################
if __name__ == "__main__":
    main()
```

5.2 Results and discussion

(recognized number plate correctly shown in the Python IDE)

6. Conclusion

Open CV has thousands of algorithms used for OCR, tracking objects and many more. In Alphanumeric character Detection extraction of features correctly is most important. Accurate detection improves the localization of normal patches on images. Without proper extraction, detection is not possible. Recognition is always performed in pairs using the pattern matching of images belonging to the Training Module. While performing number plate identification it prints the output in the IDE and shows the steps involved.
The Character Detection on Open CV is highly reliable, output is about 90-95% correct.
This System can be implemented in various fields like in parking for surveillance over the parked cars for overtime tickets, or high speeding tickets or for security purposes used by army or police, at entrance gates to verify the vehicle in its database. This can also be implemented for finding a missing/hiding car in an area, if used along with satellite imaging.

A few ANPR software vendors publish accuracy results based on image benchmarks. These results may vary depending on which images the vendor has chosen to include in their test. In 2017, Sighthound reported a 93.6% accuracy on a private image benchmark. In 2017, OpenALPR reported accuracy rates for their commercial software in the range of 95-98% on a public image benchmark. April 2018 research from Brazil's Federal University of Paraná and Federal University of Minas Gerais that compared both systems reported median recognition rate of 93.53% and stated significant improvement from the 81.8% rate obtained in previous works.

7. References

[1] https://en.wikipedia.org/wiki/Automatic_number-plate_recognition

[2] Dehghan, Afshin; Zain Masood, Syed; Shu, Guang; Ortiz, Enrique G. (February 19, 2017). "View Independent Vehicle Make, Model and Color Recognition Using Convolutional Neural Network". Archived from the original on May 30, 2018. Retrieved May 30,2018 – via ResearchGate.

[3] Laroca, Rayson; Severo, Evair; Zanlorensi, Luiz A.; Oliveira, Luiz S.; Resende Goncalves, Gabriel; Robson Schwartz, William; Menotti, David (April 28, 2018). 2018 International Joint Conference on Neural Networks (IJCNN). pp. 1–10. ISBN 978-1-5090-6014-6.