License Plate Recognition

Christopher Medina Rodríguez
Universidad Autónoma de Nuevo León
Facultad de Ingeniería Mecánica y Eléctrica

January – June 2015
e-mail: chris.medrdz@gmail.com

Final Project Computer Vision
Advisor teacher: Dr. Satu Elisa Schaeffer

Abstract

License Plate Recognition is a Project development in Python which aims to achieve the plate localization of a car for further treatment and of the same number plate detection.

The Optical Character Recognition (**OCR**) is one of text automatic identification methods most used and this project conducted OCR routines for segmentation and obtaining plaque characteristics analyzed. The purpose of this project is to obtain a correct location in the plate region car, for further analysis of characters and identify the car registration.

Is worth mentioning that has not yet been perfected on the characters recognition from the plate image, and will continue developing for release in the next system version.

Keywords: license plate recognition, object character recognition, Tesseract, Python, OpenCV.

1. Introduction

In 1976 it was created the concept of Automatic Number Plate Recognition (ANPR) by the Police Scientific Development Branch in United Kingdom and from this many countries have adopted this technology in order to bring better electronic control of security cameras, traffic vigilance on roads, variation of traffic, etc.

The plate recognition systems have become for many countries as a prerequisite for the rapid license plates identification for different situations tool for the human eye is often difficult to process quickly.

There are many elements in the environment that will complicated the optimum plate recognition, but which must be taken into consideration to solve this task. To mention some of these factors I want to emphasize the following:

- Weather conditions
- Lighting conditions
- Incorrect plate location
- Speed moving vehicle
- Poor quality and / or cameras range
- Damage and imperfections in the metal plate

2. Design and Project Description

The language that was developed was Python, using libraries and support packages such as **OpenCV** and **Python Tesseract** currently developed by Google and initially by Hewlett Packard.

In Figure 1 we can observe a common methodology plate recognition system, which I based as a starting point for the development of each system subroutines.

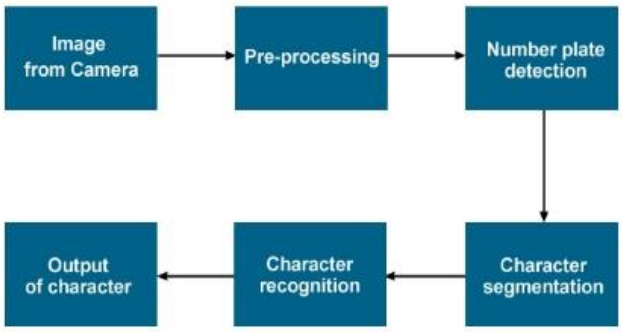

Figure 1. Number Plate Recognition Process (ANPR)

In Figure 2 show some methods in which the input image is subjected to the plate location, processing and image processing, locating the plate region, and detecting the shape plate to reduce the number of false positives in the image.

In all the process of plate recognition is carried out different image processing routines to aptly locate the plate region and extract the characters from the image detected as plaque.

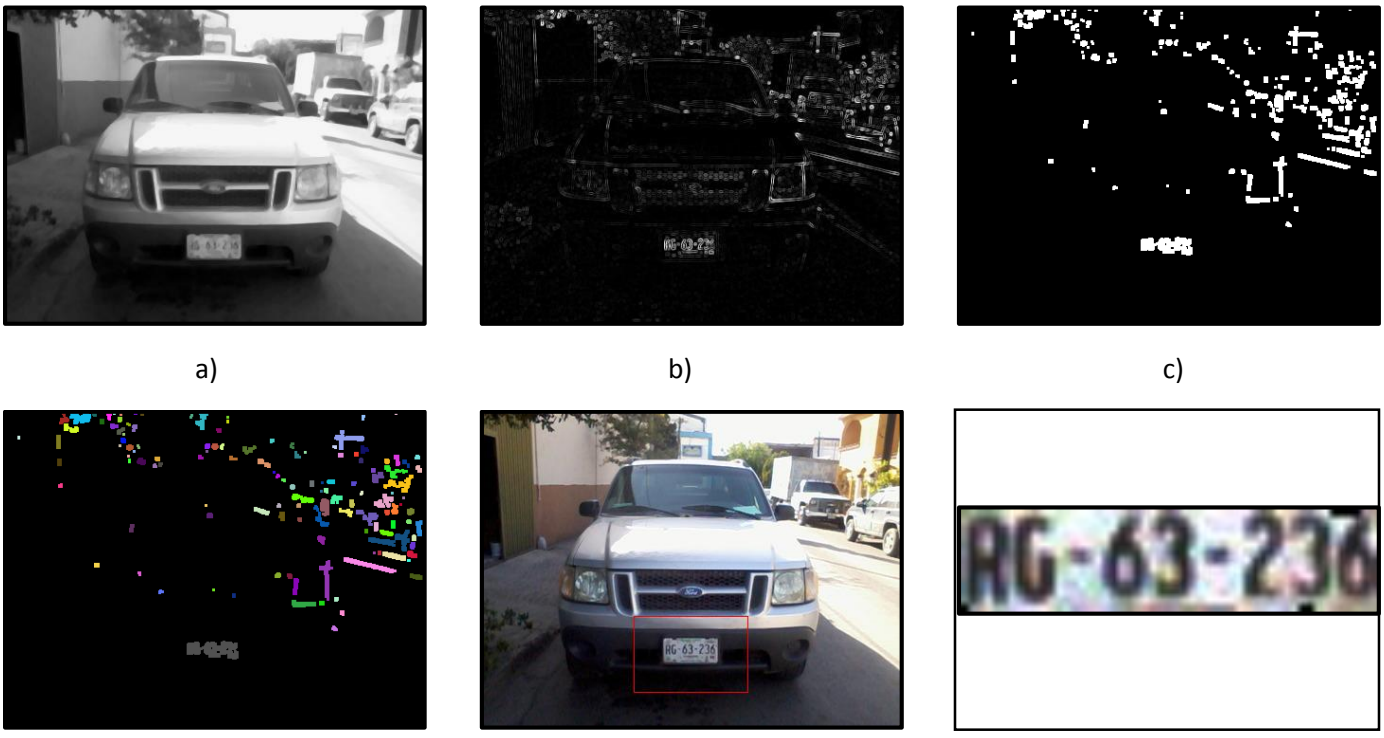

Figure 2. a) converting the image to grayscale and obtaining your threshold, b) Applying a median filter to remove noise in the input image, c) expansion and binarization of pixels, d) bfs method to find the plate form, e) detecting the plate region, f) detecting the license number plate

3. Project Construction

3.1 Image Processing (ProcessImage.py)

The input image is initially processed to improve its quality and prepare for next steps. As a starting point the image is converted to grayscale, then delete the existing noise using a filter difference. The image edges are found by the method of discrete convolution with 2D Sobel operators which masks, the image is normalized and finally it binarized to proceed to the next step which is the plate region location.

3.2 Plate Location (DetectRegion.py)

In this process, a BFS shape detection method is applied by to rule out false negative image and detect a possible plate region in the car. This script has three routines: DetectShapes, FramePlate and GetRegion; where the pixels are grouped into the first most votes and are identified with a gray color, which is the plate region in the second routine simply is marked with a red box around the plate area in the original image, and in the last routine the region is cut to subsequently send the last step.

3.3 Detection and Plate Form Reconstruction (plates.py)

```
cont = 0
contours, hier = cv2.findContours(gray,cv2.RETR_LIST,cv2.CHAIN_APPROX_SIMPLE)
for cnt in contours:
  epsilon = 0.05*cv2.arcLength(cnt,True)
  approx = cv2.approxPolyDP(cnt,epsilon,True)

  if validate(approx): # Si el Area del contorno está entre 100 y 5000 pixeles
    cv2.drawContours(imgPlate,[cnt],0,0,0)
    x,y,w,h = cv2.boundingRect(approx)
    # Se recorta la imagen con las dimensiones de la forma encontrada
    PlateCrop = imgPlate[y:y+h,x:x+w]
    cont += 1

if cont == 0:
  PlateCrop = imgPlate
```

In this snippet code it identified possible shapes of which will be validated by means of two factors: 1) aspect ratio [1: 2] y 2) plate area.

Then, is made a new cut with the letters plate area to send previously processed to Tesseract for OCR recognition.

3.3 Character Recognition (OCR.py)

Finally, is obtained of plate letters zone and applied various filters using OpenCV libraries to normalize image and equalize by histogram and for last send the final image license plate recognition.

This script has two routines: PrepareImage and Recognize; in the first is scaled the image previously processed to a base height of 100px to facilitate recognition of digits and the second is sent GetUTF8Text draw () function previously set using a whitelist which was prepared with capital letters and a script to facilitate the search for characters.

```
def PrepareImage(img_path):
    im2 = Image.open(img_path)

    # Se escala la imagen a 100px de Altura para una mejor deteccion de los caracteres
    baseheight = 100
    hpercent = (baseheight / float(im2.size[1]))
    wsize = int((float(im2.size[0]) * float(hpercent)))
    escaled_plate = im2.resize((wsize, baseheight), Image.ANTIALIAS)
    str_path = 'output/15. Placa Escalada100px.png'
    escaled_plate.save(str_path)

    display = cv2.imread(str_path)
    height, width, depth = display.shape
    channel = 1
    display = cv2.cvtColor(display, cv2.COLOR_BGR2GRAY)
    thresh = 10
    kernel = np.ones((2,2),np.uint8)
    erosion_iters = 1
    display = cv2.erode(display,kernel, iterations = erosion_iters)


    imageRec = cv.CreateImageHeader((width,height), cv.IPL_DEPTH_8U, channel)
    cv.SetData(imageRec, display.tostring(),display.dtype.itemsize * channel * (width))

    return imageRec

def Recognize(image):
    tesseract.SetCvImage(image,api)
    full_text = ""
    full_text = api.GetUTF8Text()
    return str(full_text[0:9])
```

4. Implementation

It user interface has three buttons, which can be navigated images contains in the project folder */input* and initiate the process of plate recognition.

It also has two spaces where the detected plate displays and other recognized car license plate.

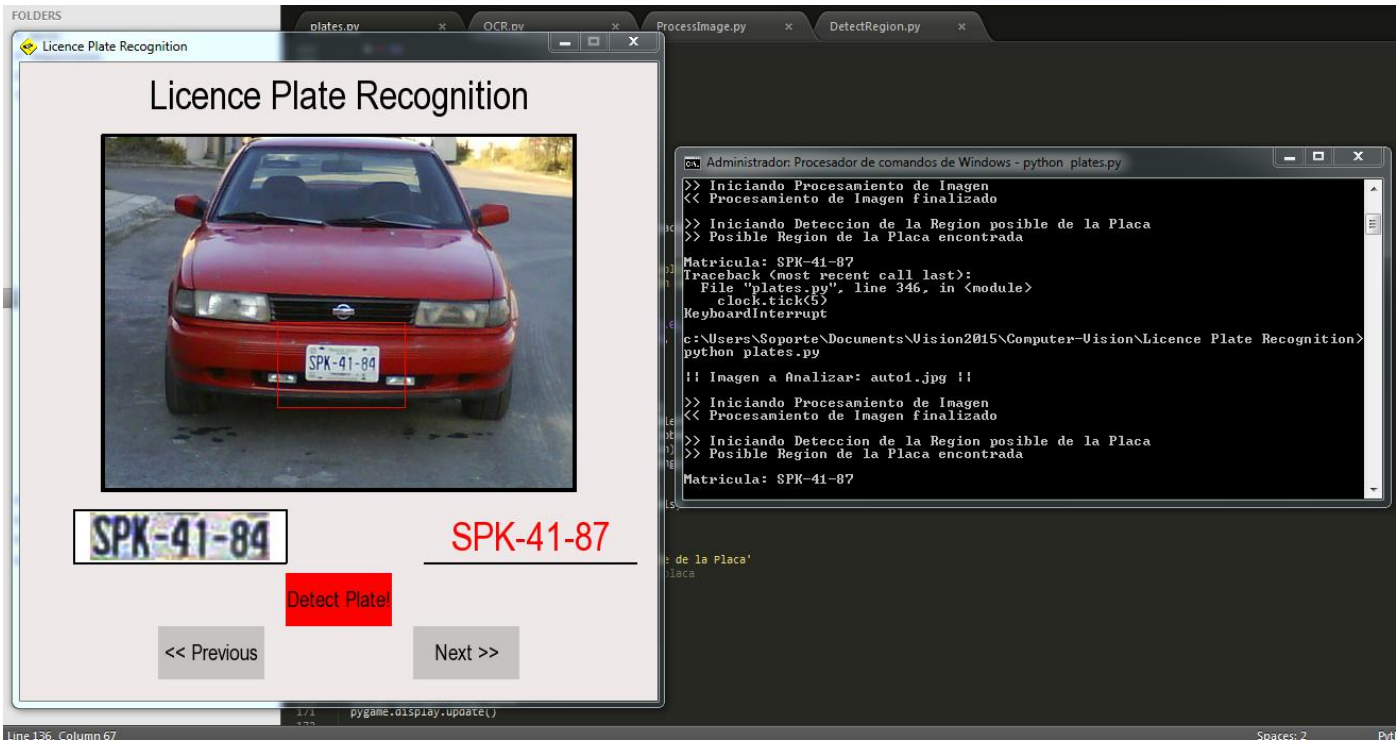

Figure 3. Project User Interface and execution console.

5. Conclusions

The method proposed for solution of this project has the main advantage, the detection of plate region so as to eliminate as many false positive figures in the picture. The final results in the obtaining of effective plate images were in cars not so far away from the camera. Opportunity areas were detected when crop the plate letters before send to OCR, so it turned out complicated processing.

It will seek to improve the of plate letters segmentation for better character recognition, so the angle correction and different sizes of license plates of greater distance with the camera. I believe that this first version is met for the purpose of detecting the location and license number plate through basic techniques of computer vision.

6. References

[1] Automatic Number Plate Recognition in Shogun, *http://nbviewer.ipython.org/gist/kislayabhi/89b985e5b78a6f56029a*

[2] ALPR using Python and OpenCV, http://sajjad.in/content/ALPR_paper.pdf

[3] Image Database: Mastering OpenCV with Practical Computer Vision Projects, *http://www.zemris.fer.hr/projects/LicensePlates/english/images.html*

[4] TESSERACT Manual Page, *http://tesseract-ocr.googlecode.com/svn/trunk/doc/tesseract.1.html*

[5] Histogram Equalization, Fredrik Lundh, *http://effbot.org/zone/pil-histogram-equalization.htm*