# License Plate Recognition

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

Enero – Junio 2015
e-mail: chris.medrdz@gmail.com


**Proyecto Final Visión Computacional**
Maestro Asesor: Dra. Satu Elisa Schaeffer

Resumen

**License Plate Recognition** es un proyecto desarrollado en lenguaje de programación Python el cual tiene como objetivo lograr una localización de la Placa de un automóvil para su posterior tratamiento y detección del número de matrícula de dicho vehículo.

El Reconocimiento Óptico de Caracteres (**OCR**, por sus siglas en inglés), es uno de los métodos de identificación automática de textos más utilizados y en éste proyecto se llevan a cabo rutinas de OCR para la segmentación y obtención de características de la placa a analizar.

El propósito de éste proyecto es lograr una correcta localización de la región de la placa de un automóvil, para su posterior análisis de caracteres e identificar la matrícula del automóvil.

Cabe mencionar que aún no se ha perfeccionado la parte del reconocimiento de los caracteres de la imagen de la placa, y seguirá en desarrollo para su liberación en la siguiente versión del sistema.

*Palabras clave: license plate recognition, object character recognition, Tesseract, Python, OpenCV.*

## 1. Introducción

En 1976 fue creado el concepto de **Automatic Number Plate Recognition (ANPR)** por el Departamento Policiaco de Desarrollo Científico del Reino Unido y a partir de aquí numerosos países han adoptado ésta tecnología para así llevar un mejor control electrónico de cámaras de seguridad, vigilancia vial en carreteras, variación del tráfico, etc.

Los sistemas de reconocimiento de placas se han convertido para muchos países como una herramienta indispensable para la rápida identificación de matrículas para diversas situaciones que para el ojo humano es muchas veces difícil de procesar rápidamente.

Existen muchos elementos en el ambiente que harán complicado el reconocimiento óptimo de placas, pero los cuales se tienen que tener en cuenta para resolver ésta tarea. Por mencionar algunos de estos factores de quiero destacar los siguientes:

- Las condiciones del clima
- Condiciones de iluminación
- Incorrecta ubicación de la placa
- Velocidad del vehículo en movimiento
- Poca calidad y/o alcance de las cámaras
- Daños e imperfecciones en el metal de la placa.

## 2. Diseño y Descripción del Proyecto

El lenguaje en que fue desarrollado fue Python, utilizando librerías y paquetes de apoyo como lo son **OpenCV y Python Tesseract**, desarrollado actualmente por Google e inicialmente por la compañía Hewlett Packard.

En la *Figura 1* podemos observar la metodología común de un sistema de reconocimiento de placas, en la cual me he basado como punto de partida para el desarrollo de cada una de las subrutinas del sistema.

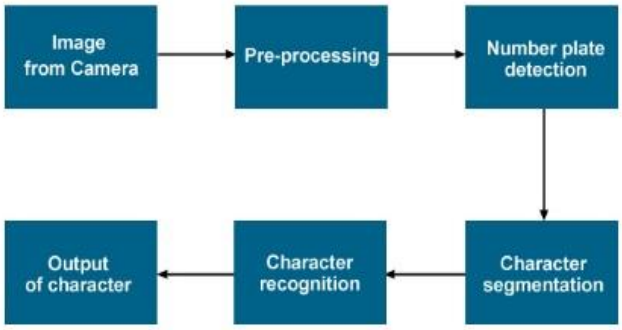


*Figura 1. Proceso de Reconocimiento de Placas (ANPR)*

En la *Figura 2* se muestran algunos de los métodos en los que la imagen de entrada se somete para la localización de la placa, el procesamiento y tratamiento de la imagen, la localización de la región de la placa, y la detección de la forma de la placa para reducir el número de falsos positivos en la imagen.

En todo el proceso de reconocimiento de la placa se llevan a cabo diversas rutinas de tratamiento de imágenes para poder localizar acertadamente la región de la placa y extraer así los caracteres de la imagen detectada como placa.

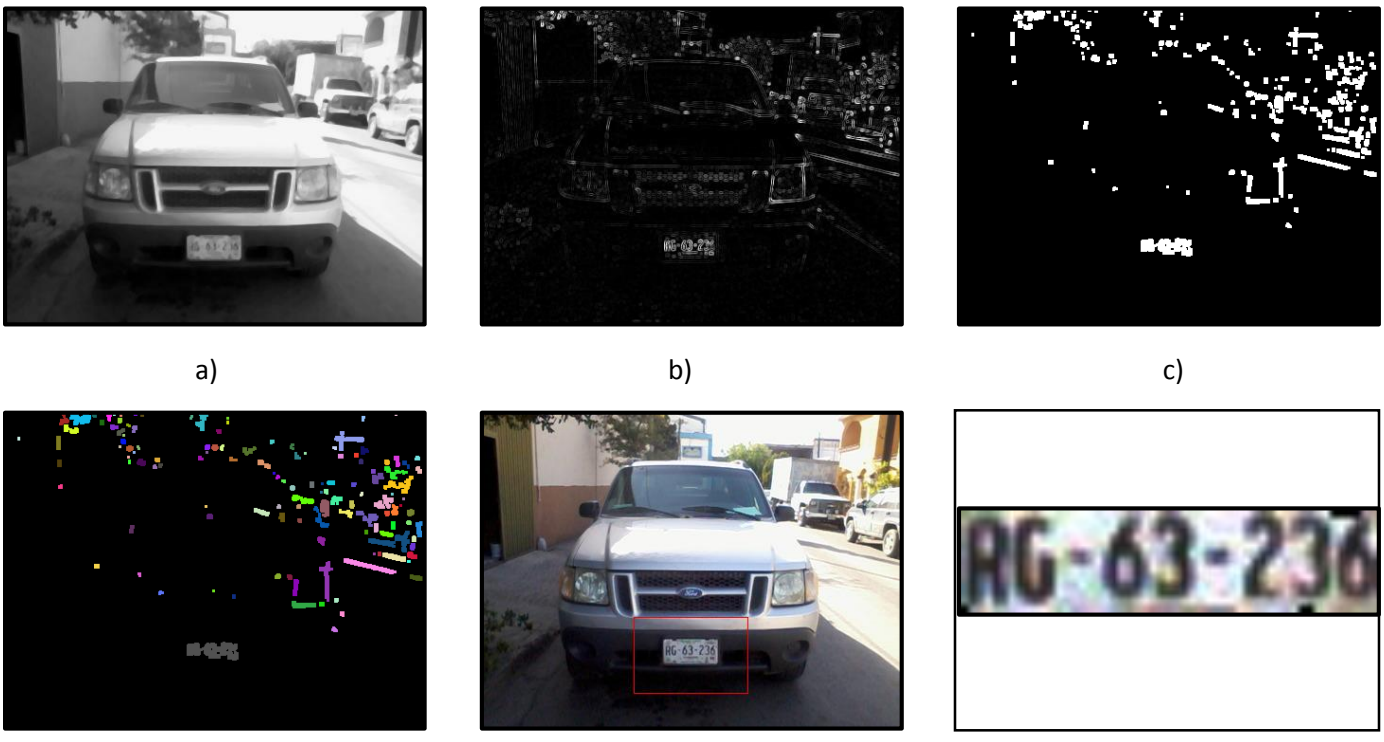


*Figura 2. a) Conversión de la imagen a escala de grises y obtención de su umbral, b) Aplicación de un filtro medio para eliminar ruido en la imagen de entrada, c) dilatación y binarización de pixeles, d) método bfs para encontrar la forma de la placa, e) detección de la región de la placa, f) detección de la zona de texto de la placa.*

## 3. Construcción del Proyecto

*3.1 Procesamiento Imagen (ProcessImage.py)*

*La imagen de entrada se procesa inicialmente para mejorar su calidad y prepararla para próximas etapas. Como punto inicial se convierte la imagen a escala de grises, para después eliminar el ruido existente mediante un filtro diferencia. Se encuentran los bordes de la imagen mediante el método de convolución discreta 2D con operadores de Sobel para las máscaras, se Normaliza la imagen y se Binariza finalmente para poder continuar con el siguiente paso que es la localización de la región de la placa.*

*3.2 Localización de Placa (DetectRegion.py)*

*En éste proceso, se aplica un método de detección de formas mediante BFS para así descartar falsos negativos en la imagen y detectar una posible región de la placa en el automóvil. Éste Script cuenta con tres rutinas: DetectShapes, FramePlate* y *GetRegion*; donde en la primera se agrupan los pixeles con mayor votos y se identifican con un color gris, los cuales será la región de la placa, en la segunda rutina simplemente se marca con un recuadro rojo la zona de la placa en la imagen original, y en la última rutina se corta la región para enviarla posteriormente al último paso.

*3.3 Detección y Reconstrucción de Forma de la Placa (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
```

Mediante éste fragmento de código se identifican los posibles bordes de la placa la cual se validará mediante dos factores: 1) proporción de aspecto [1:2] y 2) área de la placa.

Posteriormente se hace un nuevo recorte solamente con el área de las letras de la placa para poder enviarlo, previamente procesado, a Tesseract para su reconocimiento OCR.

*3.3 Reconocimiento de Caracteres (OCR.py)*

Por último se obtiene el recorte de la zona de las letras de la placa y se aplican mediante librerías de OpenCV diversos filtros para normalizar la imagen y equalizarla mediante histograma y enviar la imagen final a Tesseract para su reconocimiento de matrícula.

Éste Script cuenta con dos rutinas: PrepareImage y Recognize; en la primera se escala la imagen previamente procesada a una altura base de 100px para poder facilitar el reconocimiento de los dígitos y en la segunda se manda llamar la función *GetUTF8Text()* previamente configurada mediante un ***whitelist*** el cuál se preparó con letras mayúsculas y un guion para facilitar la búsqueda de caracteres.

```
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. Implementación

La interfaz de usuario cuenta con tres botones, los cuáles se puede navegar con las imágenes contendias en el folder */input* del proyecto e iniciar con el proceso de reconocimiento de placas.

Además cuenta con dos espacios en donde se mostrará la placa detectada y en otro la matrícula del automóvil reconocida.

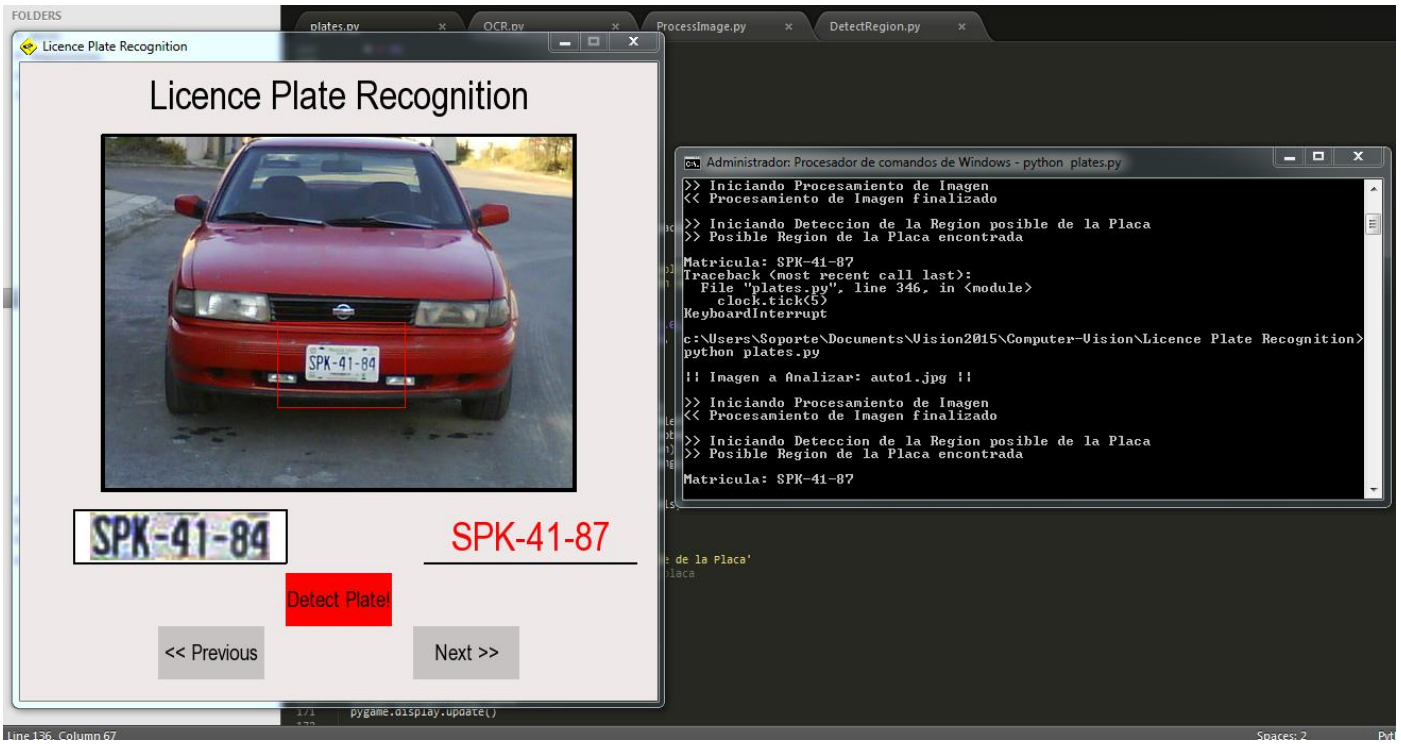


*Figura 3. Interfaz de Usuario del Proyecto y consola de ejecución.*

## 5. Conclusiones

El método propuesto para la solución de éste proyecto tiene como ventaja principal, la detección de la región de la placa para así eliminar el mayor número de falsos positivos de figuras en la imagen. Los resultados finales en la obtención de la placa fueron efectivos en imágenes con autos no tan alejados de la cámara. Se detectaron áreas de oportunidad en el recorte de las letras de la placa antes de enviar a OCR, por lo que resulto complicado el procesamiento.

Se buscará mejorar la segmentación de las letras de la placa para un mejor reconocimiento de los caracteres, así también la corrección del ángulo y diferentes tamaños de placas de autos de mayor lejanía con la cámara. Considero que para ésta primera versión se cumple el objetivo de la localización y detección del número de matrícula mediante técnicas básicas de visión computacional.

## 6. Referencias

[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*