python

Scientific Libraries 1: NumPy

Ing. Juan Camilo Correa Chica

Scientific Libraries 1: NumPy

- Ejercicio de aplicación: License Plate Segmentation

Ejercicio de aplicación

El reconocimiento de placas de automóviles es una de las aplicaciones más difundidas del procesamiento digital de imágenes. En esta ocasión no vamos precisamente a reconocer una placa, simplemente vamos a realizar un paso previo, la segmentación de la placa. La segmentación corresponde al proceso de separar la placa del resto de la imagen del automóvil para que en un paso posterior un algoritmo de reconocimiento de caracteres (OCR) se encargue de descifrar las letras y números de la placa.

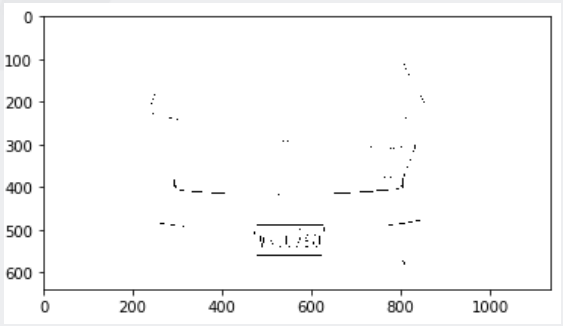
En este ejercicio vamos a tener una fotografía de un automóvil, y luego se aplicarán los conceptos de programación en Python junto con las funciones para manipulación de arreglos que vienen incluidas en la librería Numpy para segmentar la placa.

Para realizar dicho procesamiento se echará mano de la técnica de ventanas deslizantes concéntricas (SCW) y de algunas técnicas empíricas adicionales.

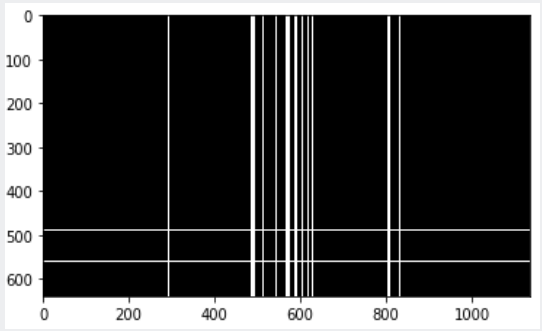
Ejercicio de aplicación

Proceso de segmentación de placas de automóviles:

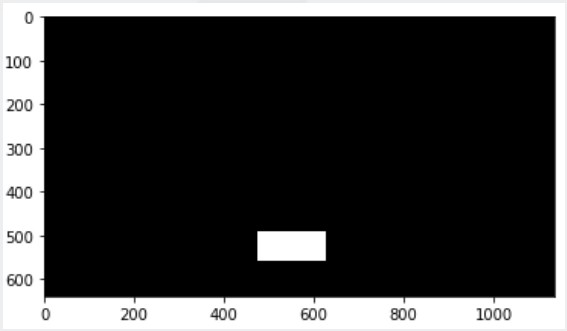
Carga de la imagen original



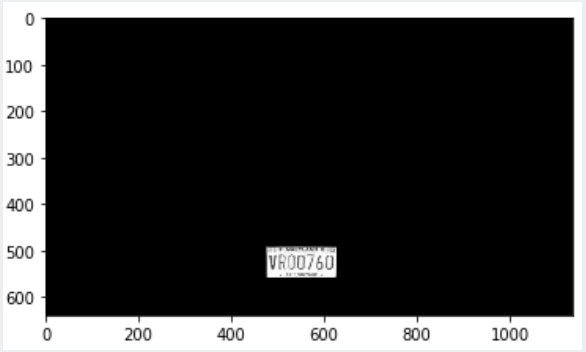
Búsqueda de bordes de posibles zonas rectangulares (SCW)



Identificación de región de interés ROI (técnica empírica)



Máscara binaria para segmentación de la ROI



Segmentación de la placa

Ejercicio de aplicación

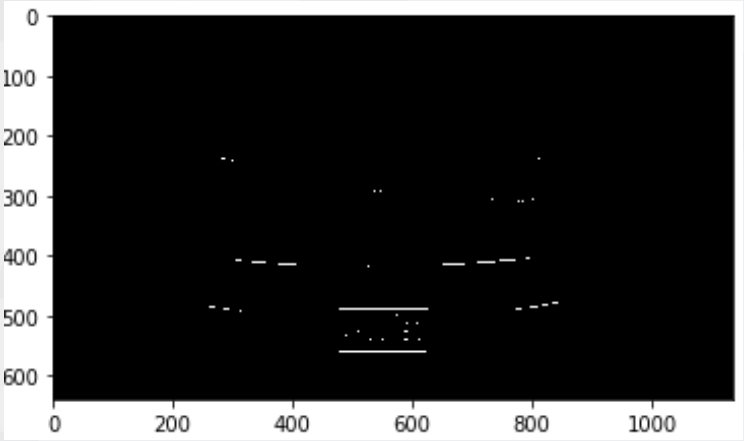
Carga de la imagen: Para el ejercicio se proporciona el archivo de imagen "**lp3.jpg**". Para poderla cargar como un arreglo matricial se utiliza la función "**imread**" de la librería de Python llamada "**pylab**". Para poder procesar la imagen, primero debe pasarla a escala de grises, lo cual se puede lograr eliminando la tercera dimensión del arreglo de la imagen (almacene el arreglo bidimensional en una variable adicional)

Imagen en color RGB

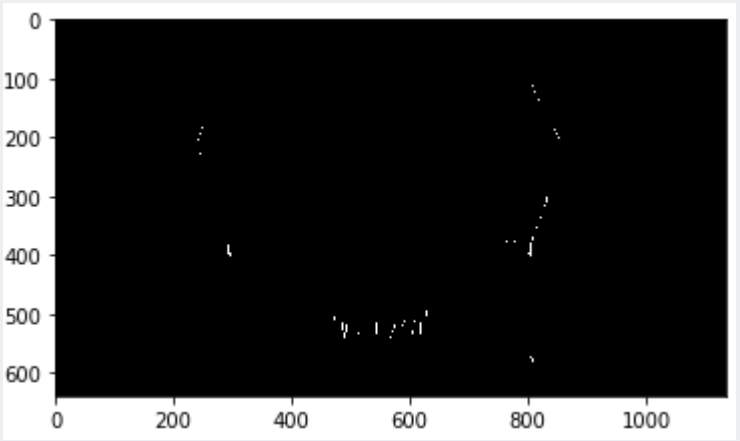
Imagen en escala de grises

Ejercicio de aplicación

Búsqueda de bordes de posibles zonas rectangulares: Para buscar posibles zonas rectangulares se debe aplicar la técnica de ventanas deslizantes concéntricas (SCW), con esta técnica se logra conformar dos arreglos, uno que contiene los bordes verticales, y otro que contiene los bordes horizontales de todas las posibles zonas rectangulares que puedan existir en la imagen. Esto es debido a que la placa por lo general está enmarcada en una zona de forma rectangular.



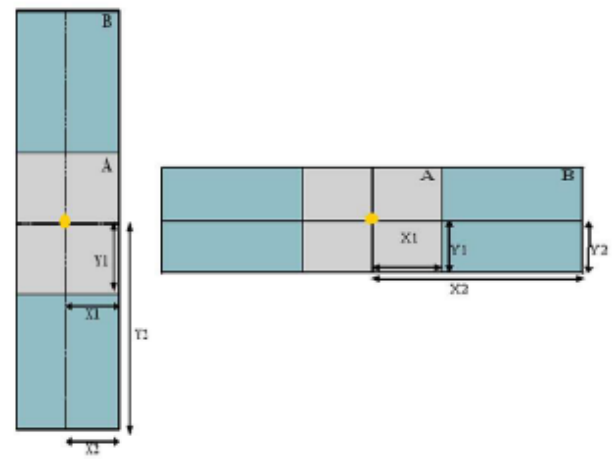
Bordes horizontales de posibles zonas rectangulares

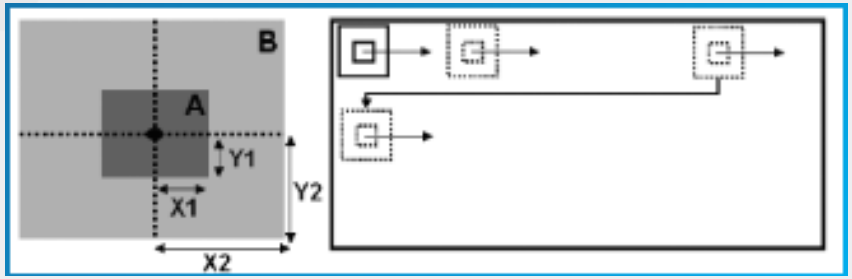


Bordes verticales de posibles zonas rectangulares

Ejercicio de aplicación

Técnica de ventanas deslizantes concéntricas (SCW): Para buscar posibles zonas rectangulares se debe aplicar la técnica de ventanas deslizantes concéntricas que no es más que recorrer la imagen completa pixel a pixel evaluando una característica estadística de los pixeles que rodean al pixel en cuestión. Para hacer esto es necesario crear dos ventanas (arreglos), uno de mayor área que el otro pero ambos centrados en un mismo pixel, las ventanas van recorriendo los pixeles de la imagen y van evaluando la razón entre las medias o desviaciones estándar de los pixeles que están contenidos en las ventanas en ese momento.





Ventanas concéntricas y su desplazamiento por la imagen

Ejercicio de aplicación

Técnica de ventanas deslizantes concéntricas (SCW): El tamaño de las ventanas es un dato que se va consolidando empíricamente conforme se van ejecutando rondas del algoritmo y se va ajustando este parámetro. Se debe escoger un umbral (threshold) según el cual se decide si un pixel hace parte de un borde (vertical u horizontal) de la imagen. De acuerdo al umbral y la razón entre las medidas estadísticas de las ventanas se crea un arreglo binario en el cual se da un valor de 1 a un pixel que se considera como perteneciente a un borde de una zona rectangular, y un valor de cero a los pixeles que no. El umbral también es un parámetro que se va ajustando empíricamente a medida que se ejecutan rondas del algoritmo.

$$I_1(x',y') = \begin{cases} 0, & \text{if } \frac{std_B}{std_A} \leq T \\ 1, & otherwise \end{cases}$$

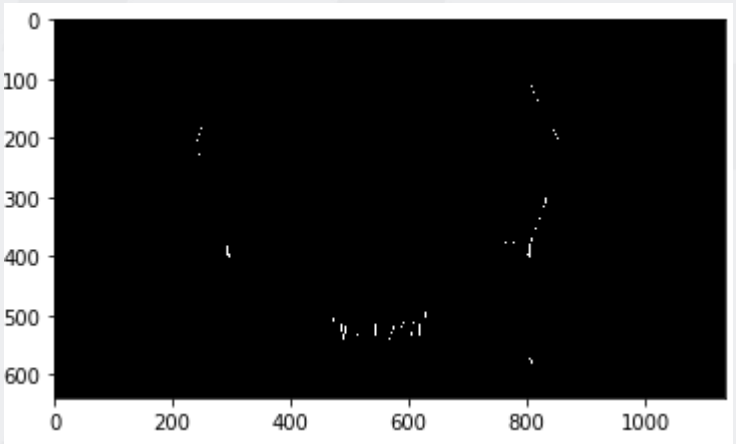
Criterio para los pixeles de un borde (desviación estándar)

$$in\ I_1(x,y) \Rightarrow \begin{cases} I_{AND}(x,y) = 0, & if\ \frac{M_B}{M_A} \leq T \\ I_{AND}(x,y) = 1, & if\ \frac{M_B}{M_A} > T \end{cases}$$

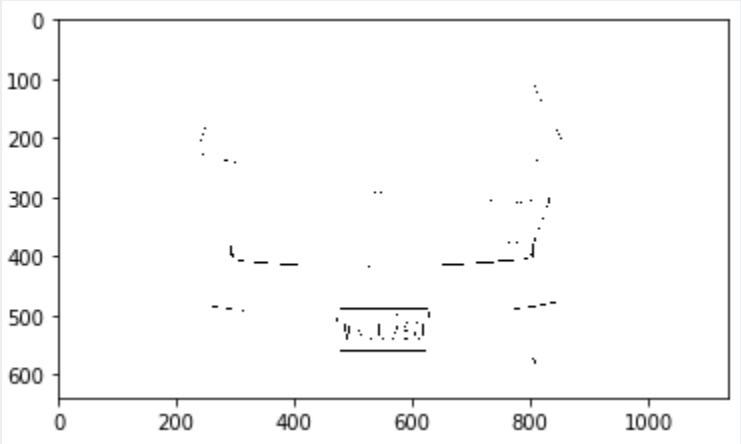
Criterio para los pixeles de un borde (media)

Ejercicio de aplicación

Consolidación de las zonas rectangulares: Para revelar cuales pueden ser las posibles zonas rectangulares presentes en la imagen se deben mezclar los arreglos obtenidos para los bordes verticales y para los bordes horizontales. Como los anteriores son arreglos binarios entonces se puede aplicar una operación lógica "**OR**" para combinar ambos. Luego puede invertir el arreglo resultante para que se pueda observar con mayor facilidad el resultado (operación lógica "**NOT**").



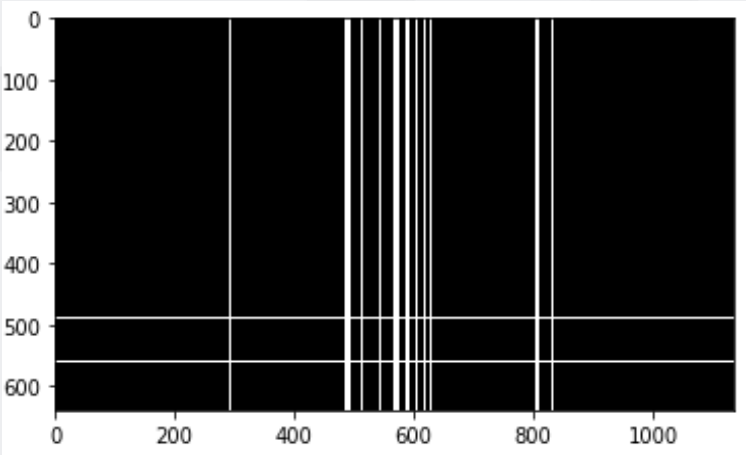
Bordes horizontales y verticales consolidados



Al invertir la imagen binaria

Ejercicio de aplicación

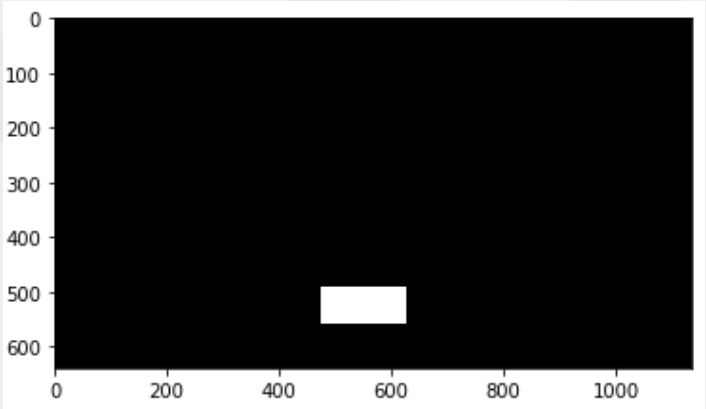
Identificación de la región de interés ROI: De todas las posibles zonas rectangulares se debe identificar la que corresponde a la placa del automóvil, este es un proceso empírico en el que se hace un conteo de el número de pixeles por filas y columnas y se crea un nuevo arreglo en el que se resaltan las filas y las columnas que obedecen a un criterio empírico definido, en este caso el criterio será las filas y columnas que contengan cierta cantidad de pixeles en 0 o en 1 dependiendo de si el arreglo de consolidación de bordes se invirtió o no. El umbral para el número de pixeles horizontales y verticales también es un parámetro que se ajusta de acuerdo a los resultados obtenidos al correr el algoritmo en repetidas ocasiones.



Arreglo en el que se resaltan las filas y columnas que cumplen cierto criterio de conteo de pixeles.

Ejercicio de aplicación

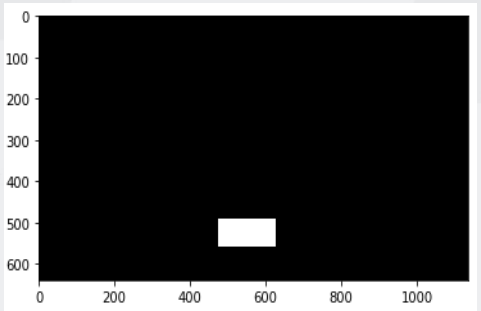
Identificación de la región de interés ROI: Una vez resaltadas las filas y columnas que pudieran encerrar la zona rectangular de interés se den seleccionar los bordes horizontales de la ROI de acuerdo aun parámetro empírico que solo elija las filas que encierren una zona rectangular con una altura similar a la que se esperaría para una placa de automóvil en una fotografía similar. Luego se escogen los borden verticales como los límites de la zona de mayor concentración de líneas verticales. La intersección entre la zona de bordes horizontales y verticales resulta en la zona de interés en la que posiblemente se encuentre la placa. Luego se construye un arreglo binario en el que todas las posiciones son cero a excepción de las posiciones encerradas dentro de los limites hallados para la ROI.

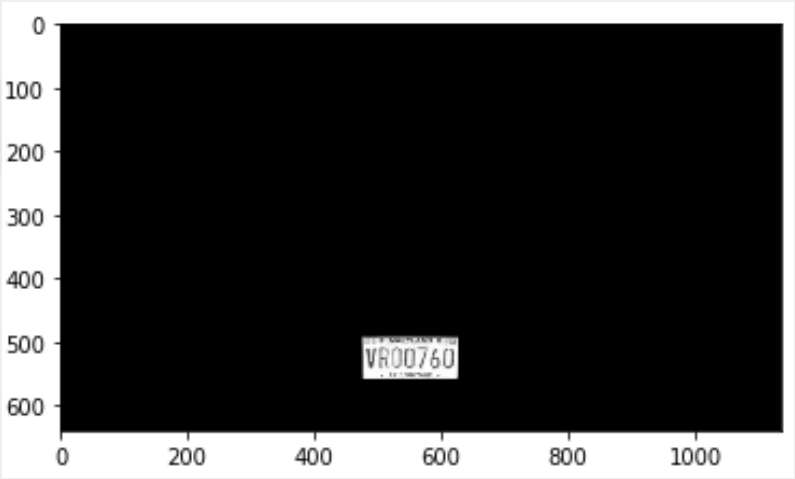


Arreglo que representa a la zona de interés, ROI, encontrada luego de delimitar los bordes horizontales y verticales.

Ejercicio de aplicación

Segmentación de la placa: En el paso anterior se definió una máscara binaria que va permitir recortar la placa de la imagen original, de ese modo se completa el proceso de segmentación. Se debe multiplicar el arreglo de la máscara binaria (ROI) con el arreglo de la imagen en escala de grises, luego se obtiene como resultado una imagen con la placa segmentada.





Segmentación de la placa

Ejercicio de aplicación

Funciones útiles:

Carga de imágenes: pylab.imread('ruta_a_la_imagen')
Agregar plot para imagen: pylab.figure()
Imprimir arreglos como imágenes: pylab.imshow(arreglo, cmap=pylab.cm.gray)
Crear arreglos numpy vacíos: numpy.empty((m,n), dtype=tipo_de_dato)
Obtener dimensiones de un arreglo: m,n = numpy.shape(arreglo)
Media de los elementos de un arreglo: numpy.mean(arreglo)
Desviación estándar de los elementos de un arreglo: numpy.std(arreglo)
Operación lógica OR: numpy.logical_or(arreglo1, arreglo2)
Operación lógica NOT: numpy.logical_not(arreglo)
Multiplicación de arreglos: arreglo1 * arreglo2
Tomar todos los elementos de una fila de un arreglo: var_row = arreglo[row]
Tomar todos los elementos de una columna de un arreglo: var_col = arreglo[: ,col]
Listar todas las filas que cumplen un critero: var_rows = numpy.where (arreglo1[] == criterio)
Acceder al último elemento de una lista: var = lista[-1]
Contar elementos repetidos de una lista: lista.count(elemento)

Ejercicio de aplicación

Artículos útiles:

Deb, Kaushik. Chae, H.K. Jo, KH. Vehicle License Plate Detection Method Based on Sliding Concentric Windows and Histogram. Journal of Computers, vol 4, no 8. August 2009.

Anagnostopoulos, C.N. Anagnostopoulos, I. Loumos, V. and Kayafas, E. A License Plate Recognition Algorithm for Intelligent Transportation System Applications. IEEE Transactions on Intelligent transportation systems, 2006, vol. 7, no 3, p. 377-392.

La Numpy Cheat Sheet que está compartida en el google classroom....

Enlaces útiles

https://www.youtube.com/playlist?list=PLGBbVX_WvN7bMwYe7wWV5TZt1a58jTggB
https://www.datacamp.com/community/tutorials/python-numpy-tutorial
https://www.datacamp.com/courses/intro-to-python-for-data-science/
https://docs.scipy.org/doc/numpy/user/quickstart.html
https://docs.scipy.org/doc/numpy/
http://cs231n.github.io/python-numpy-tutorial/
http://www.labri.fr/perso/nrougier/teaching/numpy.100/
https://www.machinelearningplus.com/python/101-numpy-exercises-python/
http://arogozhnikov.github.io/2015/09/29/NumpyTipsAndTricks1.html
http://arogozhnikov.github.io/2015/09/29/NumpyTipsAndTricks2.html
https://www.dataquest.io/blog/numpy-tutorial-python/
https://www.machinelearningplus.com/python/numpy-tutorial-part1-array-python-examples/
https://www.machinelearningplus.com/python/numpy-tutorial-python-part2/
http://www.scipy-lectures.org/intro/numpy/index.html