EC5422. Tarea 1: Deteccion de caracteres

Departamento de Electrónica y Circuitos
Universidad Simón Bolívar
Andres Suarez, 12-10925
Jeckson Jaimes, 12-10446

4 de Marzo, 2018

Se requería implementar un algoritmo bajo Python, usando OpenCV, que detectara e realizara el conteo de los carácteres de una imagen compuesta por texto.

Este algoritmo contara de cuatro etapas: obtención de la imagen, optimización, filtrado y procesamiento.

1. Obtención

Se decidio trabajar con dos tipos de imagenes de prueba, una imagen con texto creado por el editor de texto Word y otra obtenida del chat de Whatsapp Web.

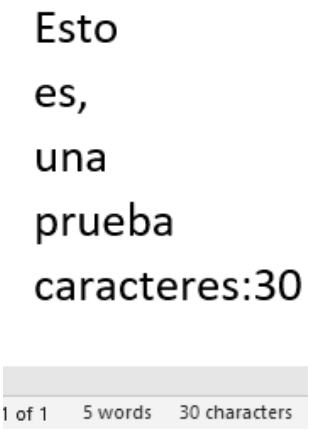


Figura 1: Imagen de prueba 1, texto de Word

Figura 2: Imagen de prueba 2, texto de Whatsapp Web

Estas dos imagenes fueron almacenadas en una variable para poder ser procesadas.

2. Optimizacion

Luego de obtener las imagenes se procedio a transformar las imagenes a la escala de grises, debido a que de este modo el procesamiento se simplifica. Una vez la imagen en blanco y negro se le aplica una

funcion de binarización, de este modo la imagen quedara solo con dos valores blanco(255) y negro(0).

La funcion de binarizacion se conoce como *cv2.threshold*, esta funcion recorre pixel a pixel haciendo una comparacion de valores, si el valor es menor que el threshold este pixel pasa a 0, y en caso contrario pasa a 255. El threshold en nuestro caso fue establecido en 127 ya que es el que nos permitia una mejor binarizacion de la imagen. Obteniendo asi los siguientes resultados:

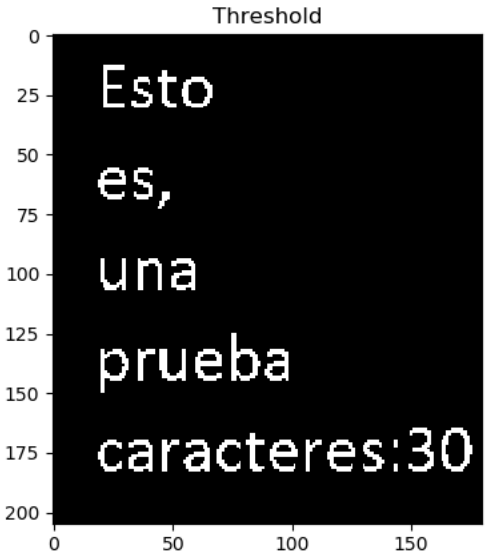


Figura 3: Imagen de prueba 1, Binarizacion

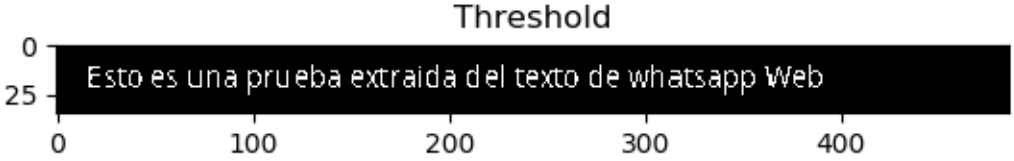


Figura 4: Imagen de prueba 2, Binarizacion

3. Filtrado

Para finalizar se le aplican tres filtros a la imagen para mejorarla y lograr hacer la detección de los caracteres requeridos. Los tres filtros usados:

- Erosion
- Laplaciano
- Dilatacion

3.1. Erosion

La erosion de una imagen no es mas que realizar una convolucion de un arreglo con la imagen, si un pixel de la imagen original sea 1 o 0 sera considerado 1 solo si todos los pixeles de la forma del arreglo son 1, de otra forma sera 0.

Las imagenes fueron erosionadas primeramente por razones que seran explicadas en la siguiente etapa de filtrado.

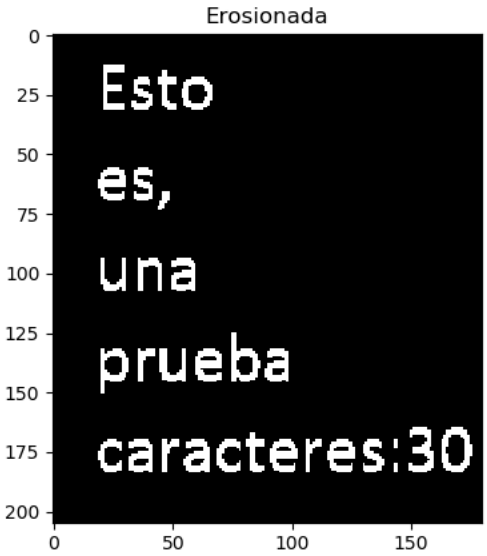


Figura 5: Imagen de prueba 1, Erosion

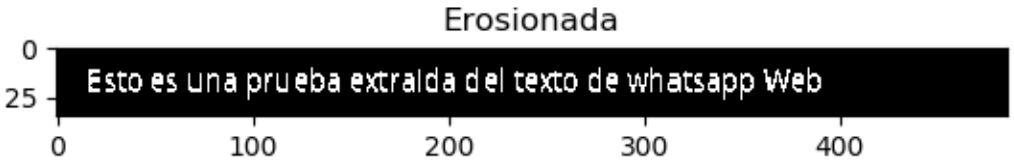


Figura 6: Imagen de prueba 2, Erosion

3.2. Laplaciano

El filtro Laplaciano se encarga de calcular los bordes que existen en las imagenes, este de igual forma es convolucionar la imagen con una matriz de caracteristicas especificas, el cual transforma los valores de los pixeles que se encuentran en los bordes de las letras a un valor máximo. Sin embargo, si la imagen posee letras que se solapan o se encuentran muy pegadas este filtro resulta muy ineficiente, es por esta razón que primero se erosiono la imagen de tal forma que los bordes quedarán mas definidos y con menos ruido.

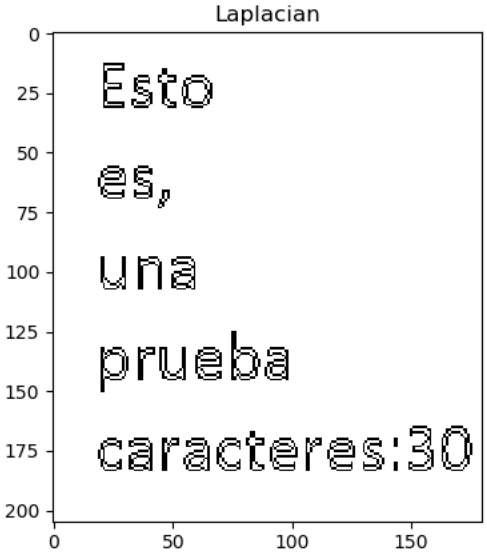


Figura 7: Imagen de prueba 1, Laplaciano

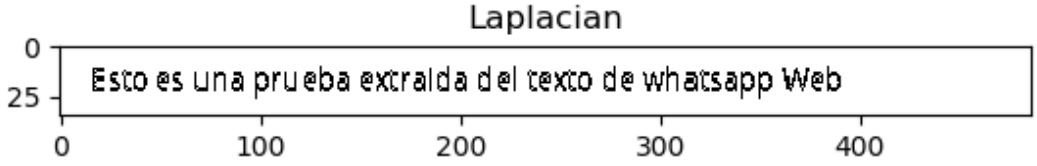


Figura 8: Imagen de prueba 2, Laplaciano

3.3. Dilatación

Por ultimo aun con los filtros aplicados anteriormente existen caracteres como la letra *i*; :, estos resultan un problema a la hora de enumerarlos debido a que cualquier maquina interpretaria dos caracteres, es por esto que se realizo una dilatación de la imagen. La dilatación al contrario de la erosion vuelve los pixeles 1 si algun valor de su arreglo sobrepuesto con la imagen original es 1, si se utiliza un arreglo vertical de longitud de 5-7 pixeles los caracteres especiales se unian formando ahora una sola figura.

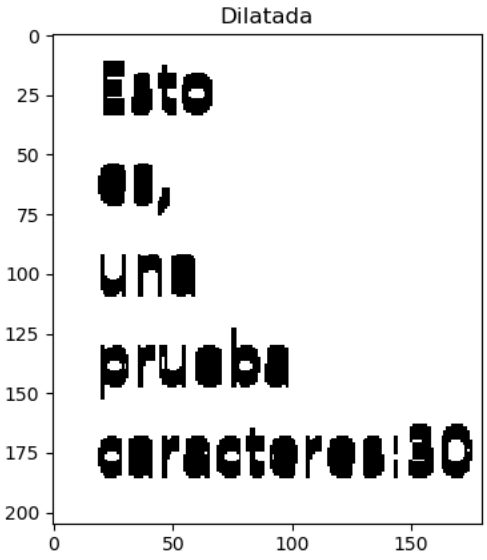
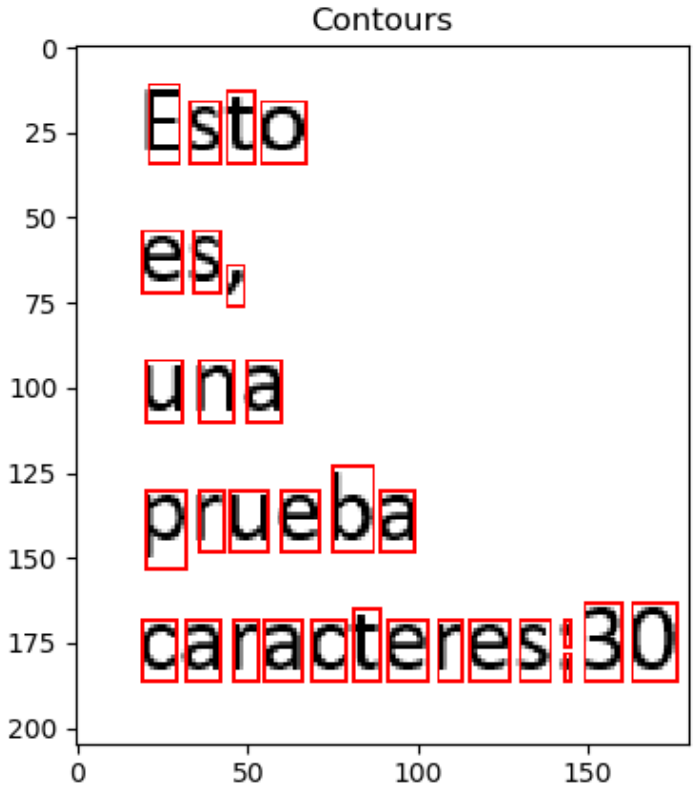


Figura 9: Imagen de prueba 1, Dilatación

Figura 10: Imagen de prueba 2, Dilatación

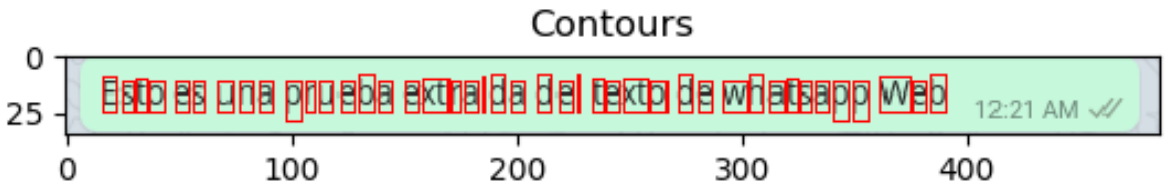
4. Procesamiento

Al tener ya la imagen procesada, se aplica un detector de contornos, que nos devuelve un arreglo con las coordenadas de los vértices de cada letra detectada. El tamaño de este arreglo nos indica la cantidad de caracteres que posee nuestra imagen. Para finalizar, pintamos en la imagen original un recuadro que indica la posición de cada carácter en esta, y observar asi los resultados.




```
PS C:\Users\Andres\Documents\PDI\tarea1> & C:/ProgramData/Anaconda3/envs/pdiclass/python.exe c:\Users\Andres\Documents\PDI\tare
a1\codes\Tarea1.py
El número de carácteres en la imagen es: 30
```

Figura 11: Imagen de prueba 1, Resultado




```
PS C:\Users\Andres\Documents\PDI\tarea1> & C:/ProgramData/Anaconda3/envs/pdiclass/python.exe c:\Users\Andres\Documents\PDI\tare
a1\codes\Tarea1.py
El número de carácteres en la imagen es: 42
```

Figura 12: Imagen de prueba 2, Resultado