

Procesamiento de Imágenes

Trabajo práctico nº 1

Augusto Rabbia

Manuel Spreutels

Octubre, 2023

Introducción

El trabajo realizado consiste de dos programas realizados en Python 3.11.4. Estos programas fueron realizados con el objetivo de resolver dos problemas:

El primero se trata de encontrar detalles ocultos en una imagen utilizando ecualización de histograma de forma local.

El segundo es validación de los campos de un formulario, siempre que tanto los formularios como sus campos cumplan con ciertas restricciones que se detallarán más adelante.

Estos problemas serán resueltos por medio de algunas de las funciones que provee la librería de procesamiento de imágenes OpenCV y la representación matricial interna de las imágenes.

Entorno de trabajo

El trabajo fue realizado sobre Python, en un entorno de desarrollo local con Visual Studio Code.

Versiones del software:

- Python: 3.11.4
- Pip: 23.1.2
- Numpy: 1.24.3
- Matplotlib: 3.7.1
- OpenCV: 4.8.1.78

Desarrollo

● Ecualización local:

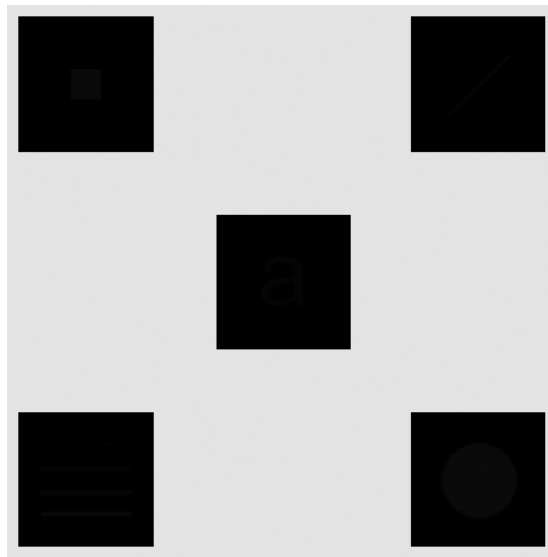
Este programa obtiene detalles escondidos de la imagen proporcionada siguiendo los siguientes pasos:

1. Creamos una copia de la imagen, la cual tendrá un borde espejado obtenido con **cv2.copyMakeBorder()**, con el parámetro de borde **cv2.BORDER_REPLICATE**.
2. Dada una máscara de cierto tamaño, la centramos sobre cada píxel de la imagen con bordes.

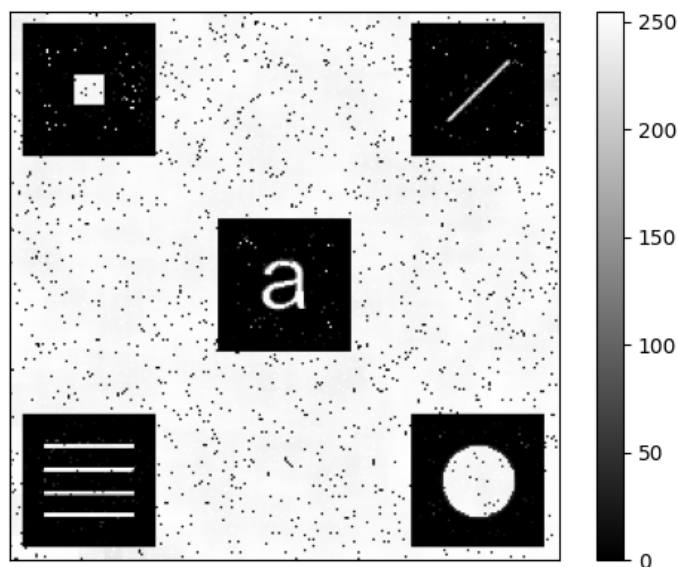
3. En cada pixel, realizamos una ecualización de histograma en su ventana de procesamiento con **cv2.equalizeHist()**.
4. Tomamos el valor que resulta en el píxel sobre el que se centra la ventana.
5. Insertamos ese pixel en una nueva copia de la imagen.

Una vez hecho este procedimiento, obtendremos una nueva imagen con sus detalles previamente ocultos descubiertos parcialmente.

Dado el siguiente input:



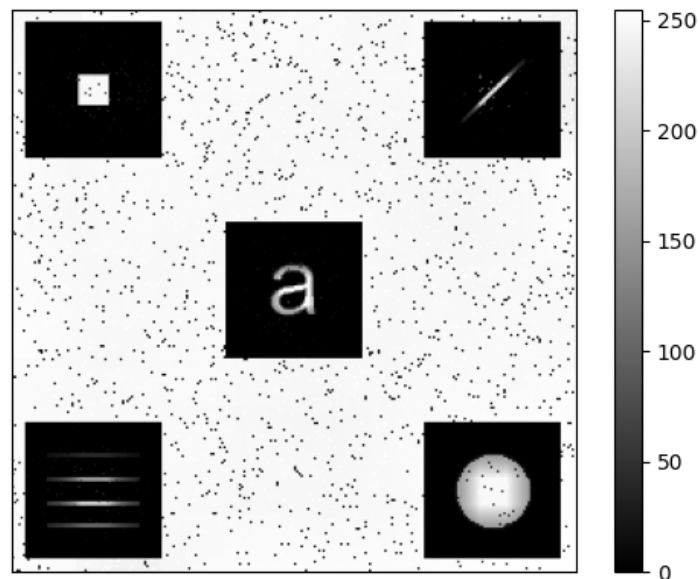
Obtenemos, con una ventana de procesamiento de 20x20 píxeles:



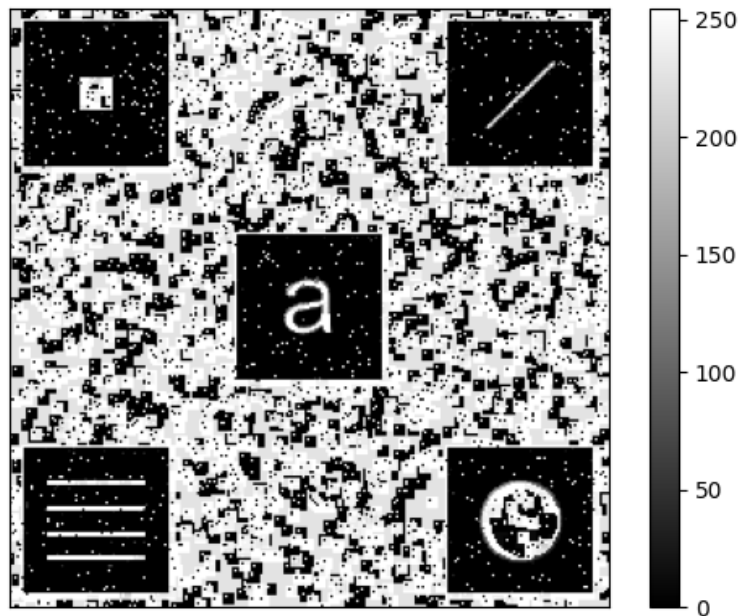
Lo que ocurre aquí es que los cuadrados negros de la imagen no tienen un color uniforme, sino que existen algunos pixeles un poco más oscuros que el resto. Por lo tanto, al hacerse el histograma se tendrán al menos dos niveles de intensidad distintos (y hasta 3, dependiendo del tamaño de la máscara). Luego, al asignar los nuevos valores $s_k = \sum_{j=1}^k p_r(r_j)$ a los píxeles que antes tenían valor r_k , los valores más claros (con k más alto) serán asignados a números más cercanos a 1, blanco después de escalar a uint8. De esta manera, el efecto final es prácticamente equivalente a aumentar el contraste al máximo, por lo que los píxeles que antes diferían por un pequeño valor ahora diferirán por mucho.

Cabe destacar que, con diferentes ventanas de procesamiento, es posible encontrar resultados que podrían resultar más convenientes dependiendo de lo que se esté buscando:

- Con ventanas grandes, tendremos menos ruido, pero los detalles escondidos estarán más difuminados. Por ejemplo con una ventana de 50x50:



- Con ventanas pequeñas, los detalles serán claros, pero a más pequeñas, mayor será el ruido, ya que hay una pequeña cantidad de pixeles con valores ligeramente diferentes distribuidos en toda la imagen. Por ejemplo con una ventana de 5x5:



- Validación de formulario:

Este programa verifica la correctitud de un formulario con el siguiente formato específico:

FORMULARIO A		
Nombre y apellido		
Edad		
Mail		
Legajo		
	Si	No
Pregunta 1		
Pregunta 2		
Pregunta 3		
Comentarios		

Donde se debe corroborar que cada uno de sus campos cumpla con las siguientes restricciones:

1. Nombre y apellido: Debe contener al menos 2 palabras y no más de 25 caracteres en total.
2. Edad: Debe contener 2 o 3 caracteres.
3. Mail: Debe contener 1 palabra y no más de 25 caracteres.
4. Legajo: 8 caracteres formando 1 sola palabra.
5. Preguntas: se debe marcar con 1 caracter una de las dos celdas SI y NO. No pueden estar ambas vacías ni ambas completas.
6. Comentarios: No debe contener más de 25 caracteres.


Para lograr esta tarea más compleja, el programa realiza varios pasos:

1. Saturamos la imagen, convirtiéndola en una imagen binaria, con sólo valores en 0 y 255.
2. Aprovechándonos de la restricción de que todos los formularios tienen el mismo formato en términos de campos y rotación (0°), obtenemos la ubicación de todas las líneas, tanto las horizontales como las verticales, sumando los valores en cada línea horizontal y vertical que tengan un valor de 255.
3. A partir de estas líneas, podemos obtener los cuadros donde se ubicará el texto. Así, creamos una subimagen para cada campo de interés del formulario.
4. Para cada campo aplicamos la función **contar_caracteres_y_palabras()**, y comprobamos la validez de sus entradas en base al resultado de esta función y las restricciones que debe respetar cada uno de los campos. Es importante resaltar que para reconocer la cantidad de caracteres utilizamos la función que provee OpenCV **cv2.connectedComponentsWithStats()**, desestimando la primer componente por ser la componente correspondiente al fondo, de ningún interés para el análisis.

Cabe destacar que en el caso de las preguntas, la función **cv2.connectedComponentsWithStats()** encontrará una componente en la ubicación de la línea vertical que separa los dos campos. Por esta razón, se debió verificar que en los campos de preguntas hayan 2 letras en lugar de 1.

Conclusión

Ciertamente, este trabajo nos da un buen primer pantallazo acerca del poder que tiene aprovechar la representación interna de las imágenes en una computadora. Esta capacidad nos permite realizar una amplia gama de procesamiento y análisis de manera completamente automatizada, sin la necesidad de proporcionar instrucciones explícitas o entrenar a la



computadora para que reconozca conceptos específicos, como imágenes ocultas o campos en un formulario.

A través de técnicas avanzadas, las computadoras pueden captar las características intrínsecas de las imágenes y extraer información relevante sin requerir una descripción detallada de lo que se debe buscar. Esto es posible gracias a algoritmos de reconocimiento, segmentación y clasificación de imágenes.

La automatización de la extracción de información de formularios es otro ejemplo destacado de cómo las computadoras pueden utilizar su capacidad de procesamiento de imágenes para mejorar la eficiencia en tareas cotidianas. Esto ahorra tiempo a través de la automatización y reduce errores humanos.