SESION 6

PROCESADO DE IMAGEN Y VISIÓN POR COMPUTADOR

8 DE NOVIEMBRE DE 2023

Ricardo Martínez Guadalajara

Introducción

En esta sesión se va a trabajar operaciones básicas y manipulación de imágenes

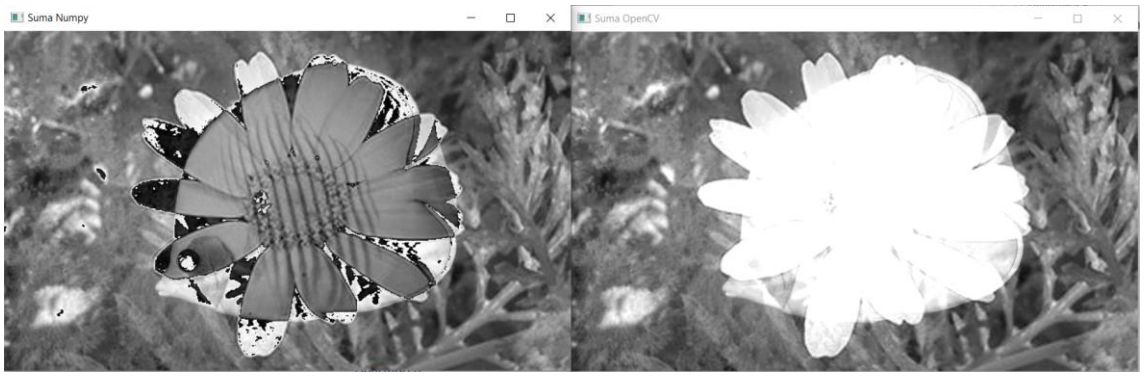
Objetivo

Desarrollar los apartados del guion de prácticas como programas de Python

Contenido

Apartado 1

En este apartado se trata de comparar los distintos métodos de suma de imágenes, con numpy y con openCV. Se aporta el programa, apartado1.py en el que se muestran las dos imágenes sumadas con ambos métodos y se pueden observar las diferencias en los resultados



La justificación es la siguiente :

#La suma con numpy se hace elemento a elemento, mientras que la suma con OpenCV se hace con la función cv2.add(),hace una suma saturada. Esto significa que si el resultado de la suma es mayor que 255 , el resultado se establece en 255. Por lo tanto, la suma con OpenCV puede dar resultados diferentes a la suma con numpy si los valores de los píxeles en las imágenes son grandes.

Apartado 2

Alpha = 0.1

Alpha = 0.5

Alpha = 1

Como conclusión se puede decir que se obtiene el resultado deseado, ya que la implementación de la función addWighted de openCV hace que se represente la suma ponderada de dos imágenes con el parámtro Alpha, que indica la transparencia de la primera imagen, complementaria de la de la segunda imagen. Produciendo un efecto fusión.

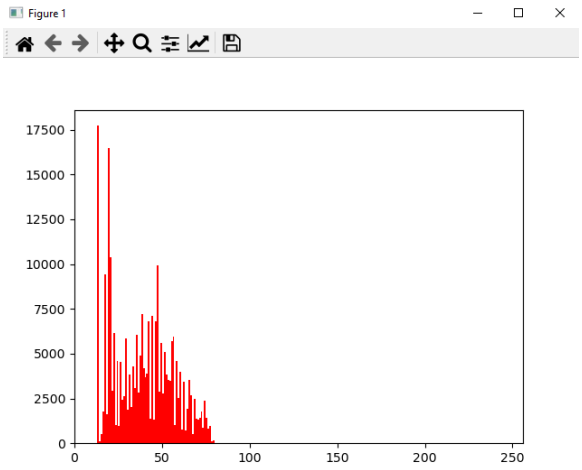
Apartado 3

En este apartado se presenta el programa apartado3.py en el que se consigue una iluminación más uniforme en la imagen Radiografia.jpg

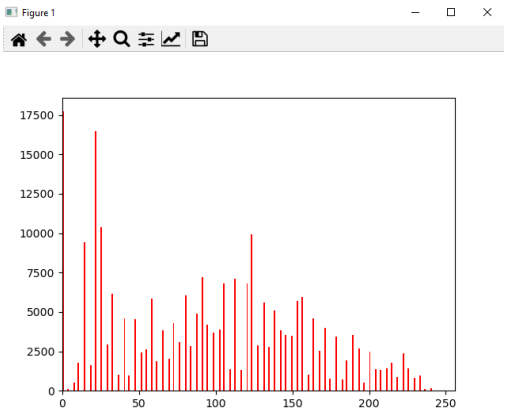
Apartado 4

En este punto se aporta un script llamado apàratod4.py en el que se normaliza y ecualiza un histograma, se muestran junto con el original por pantalla y luego muestra las tres imágenes concatenadas

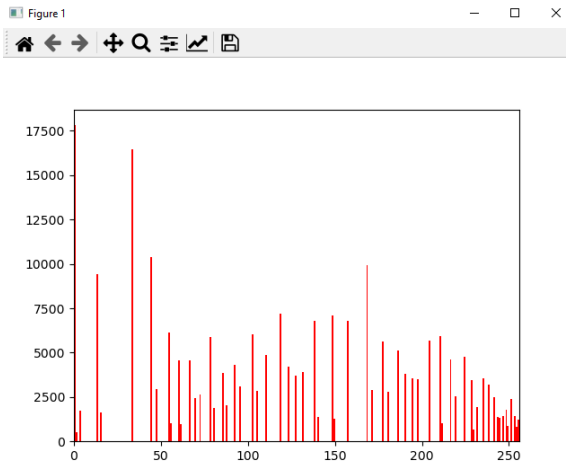
Histograma original



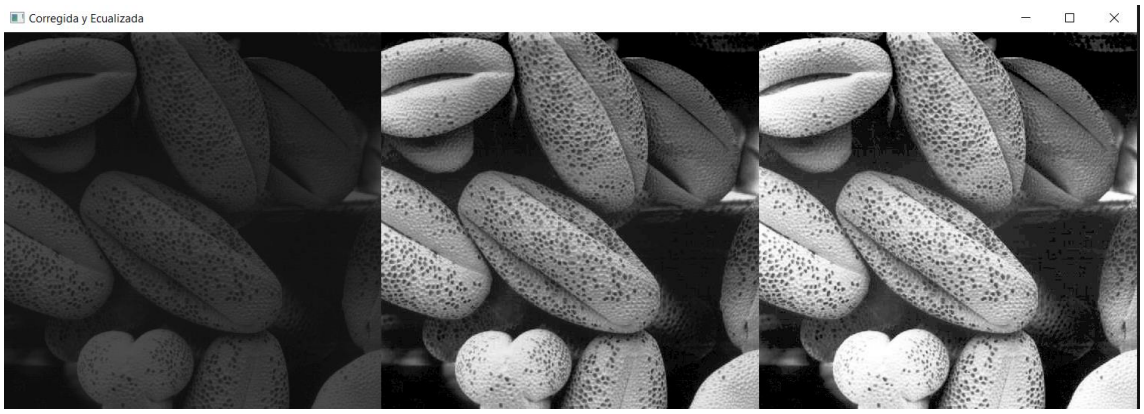
Histograma normalizado



Histograma ecualizado



Imágenes original, normalizada y ecualizada



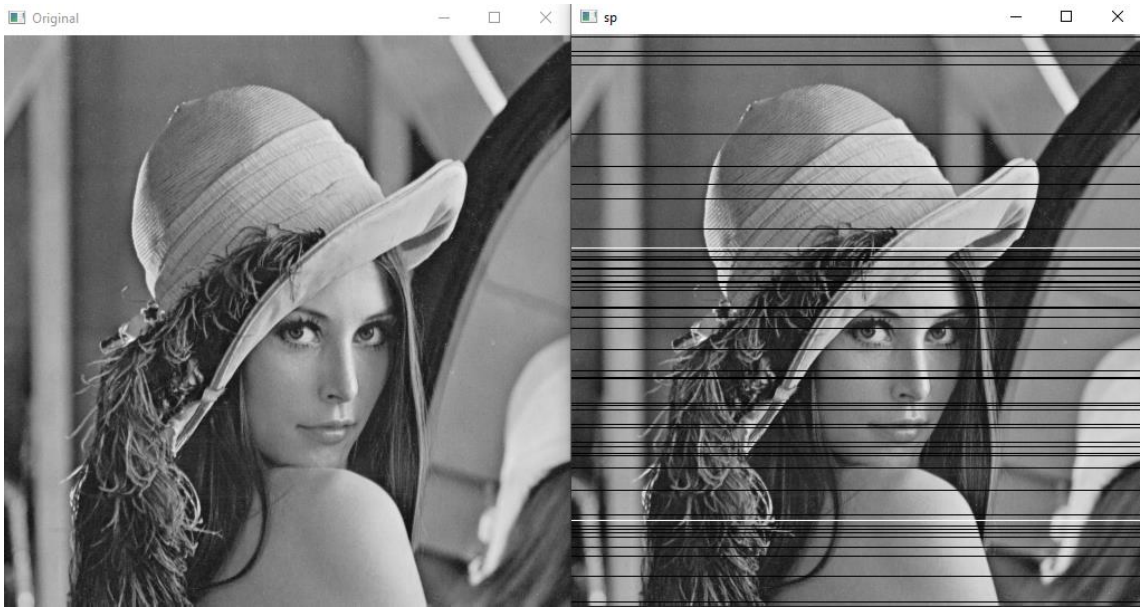
Como conclusión se pueden comparar los tres histogramas, donde se ve muy gráficamente como la normalización lo extiende a todo el rango de grises y la ecualización equilibra cada valor, produciendo una dinámica menor entre la repetición de cada nivel de gris

Apartado 5.1

En este apartado se aporta un programa apartado5.py (el mismo que para el apartado 5.2) en el que se añade ruido del tipo recogido por argumento de línea de comandos (sp=impulsivo, gauss=gaussiano) y que le aplica ese ruido.

En el caso de impulsivo, se modela con dos parámetros que controlan la cantidad de ruido de tipo sal y pimienta que se inserta.

Aquí un ejemplo con s_vs_p = 0.01 y amount = 0.0001



Tras realizar la corrección con el filtro de mediana de 3x3 se obtiene esto, con un PSNR:35.2796

Aquí se muestra un ejemplo con un filtro de 5x5

Se puede observar comparando ambos resultados que con 5x5 hay una reducción mayor de ruido pero también se produce un desenfoque de la imagen que se puede apreciar muy bien en la línea clara a la izquierda de la imagen, que aparece ligeramente deformada. En cuan al pnsr se obtiene PSNR:33.9655, un valor inferior al obtenido con el filtro de 3x3 lo que sustenta el análisis visual.

Si se pone un filtro par 4x4, se obtiene el siguiente error:

cv2.error: OpenCV(4.8.1) D:\a\opencv-python\opencv-python\opencv\modules\imgproc\src\median_blur.dispatch.cpp:285: error: (-215:Assertion failed) (ksize % 2 == 1) && (_src0.dims() <= 2) in function 'cv::medianBlur'

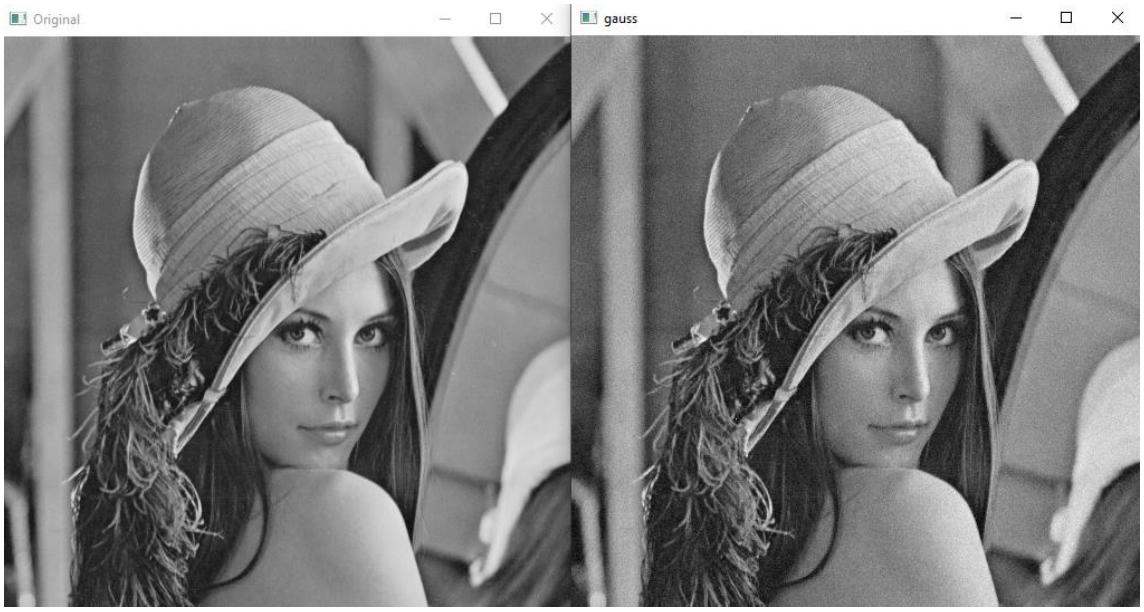
Este error se debe a que el tamaño del kernel que se está utilizando para aplicar el filtro de mediana es par. El kernel es la matriz que se utiliza para calcular el valor de cada píxel en la imagen filtrada. En OpenCV, el tamaño del kernel debe ser un número impar para que el cálculo sea correcto.

La razón por la que el tamaño del kernel debe ser impar es porque necesitamos un píxel central para calcular el valor de salida. Si el tamaño del kernel es par, no hay un píxel central y no podemos calcular el valor de salida de manera adecuada.

Esta información ha sido extraída de la documentación de opencv

Apartado 5.2

En este apartado se añade ruido gaussiano y no impulsivo el resultado obtenido con mean = 0 y var = 50 es el siguiente



Y su recuperación con filtro mediana y PSNR:34.3459 es el siguiente. Se observa una buena corrección del ruido

Si ahora se hace una recuperación con un filtro gaussiano se obtiene esto: PSNR median blur:34.3656 y PSNR Gaussian:33.8912

Con filtro gaussiano se obtiene un psnr menor, lo que sustenta la apreciación que la tercera imagen, la recuperada con filtro gaussiano es ligeramente más oscura, lo que puede ser que tiene un histograma menos normalizado, ya que ser repiten más veces las mismas tonalidades de grises

Apartado 6

Para este apartado se ha realizado el script apartado6Mod.py, en el que se muestran los PNSR para imágenes transformadas y luego para distintos tipos de algoritmos de interpolación.

PSNR for CUBIC: 34.41544292150627

PSNR for AREA: 31.155445100144274

PSNR for LINEAR: 32.309143466093644

PSNR for LANCZOS4: 33.83480943140484

La conclusión a la que se puede llegar es que al estar utilizando diferentes métodos de interpolación para redimensionar la imagen. Cada método de interpolación tiene un enfoque

diferente para estimar los valores de los píxeles, lo que puede resultar en diferentes calidades de reconstrucción.

- INTER_CUBIC: Este es un método bicúbico sobre 4x4 píxeles de la vecindad. Es el método más lento, pero produce mejores resultados.
- INTER_AREA: Este método utiliza la remuestreo de píxeles utilizando la relación de área de píxeles. Es el mejor método para reducir el tamaño de la imagen.
- INTER_LINEAR: Este es un método bilineal (utilizado por defecto). Es el método más rápido.
- INTER_LANCZOS4: Este es un método Lanczos que utiliza 8x8 píxeles de la vecindad.

Por lo tanto, si ves diferentes valores de PSNR para diferentes métodos de interpolación, esto se debe a que cada método de interpolación reconstruye la imagen de una manera ligeramente diferente, lo que resulta en diferentes calidades de reconstrucción.

La conclusión que puedes extraer de las diferencias en los valores de PSNR es que algunos métodos de interpolación son mejores para reconstruir una imagen después de redimensionarla. En general, puedes esperar que INTER_CUBIC e INTER_LANCZOS4 produzcan los mejores resultados, pero son más lentos que INTER_AREA e INTER_LINEAR.

Esta información se ha extraído de este enlace

https://docs.opencv.org/2.4/modules/imgproc/doc/geometric_transformations.html