Usos reales de filtros espaciales y morfológicos

Introducción

En el campo de procesamiento de imágenes, es necesario realizar un conjunto de operaciones para poder realizar la mejora de la imagen y su calidad visual mejore con dichas técnicas. En las técnicas de procesamiento digital de imagen siempre se tendrá como entrada la imagen que puede contener ruido en su interior y se obtendrá una imagen de salida, ya sea esta con aplicando operaciones punto a punto u operaciones de grupo de pixeles.

Las operaciones mencionadas como punto a punto, intentan mejoran el contraste o tonalidad de la imagen donde se puede diferenciar entre los valores más oscuros y más claros con un mejor detalle que la imagen original. Como se pudo observar en el trabajo realizado en la actividad 1, donde mediante la ecualización del histograma se pudo mejorar las imágenes.

Mientras que las operaciones en un grupo de pixeles, mejoran el contraste espacial en la imagen, se lo puede decir como que es diferenciar el valor del brillo determinado en un pixel y sus vecinos, dependiendo de la técnica aplicada ayuda a suavizar, reforzar o detectar bordes en las imágenes a procesar, de forma tal que los pixeles en una determinada posición de la imagen se pueden asemejar o simplemente diferenciar de los pixeles que lo rodea.

En esta actividad se tiene como objetivo la detección de bordes, el problema viene dado que es necesario identificar los bordes o fronteras de una determinada imagen o imágenes. El realce de borde en las imágenes se implementan a través de filtros espaciales. Los más utilizados son: filtros laplacianos para poder, gradiente de Prewitt, Roberts, Sobel y Kirsch.

En este trabajo se va a hacer la demostración del uso de los filtros laplacianos en una para poder realzar los bordes en una imagen, independientemente de la posición que esta se encuentre.

Esta operación está basada en la tasa de cambio de la pendiente del brillo dentro de un núcleo de pixeles de dimensión 3x3. La máscara Laplaciano más común está formada por un 8 en la posición central y -1 en las posiciones que lo rodean.

$$\begin{matrix} -1 & -1 & -1 \\ -1 & +8 & -1 \\ -1 & -1 & -1 \end{matrix}$$

Los coeficientes suman 0 y como en el caso de una máscara de filtro de paso alto, coeficientes con valores negativos rodean al coeficiente del centro que es un valor positivo grande. La operación de realce Laplaciano genera picos más marcados o abruptos en los bordes que la operación de gradiente. Cualquier pendiente de brillo, ya sea positiva o negativa, es acentuada dando al Laplaciano su carácter omnidireccional.

En una región de una imagen que es uniforme en brillo o con gradiente de brillo uniforme, el resultado de aplicar la máscara anterior es reducir el nivel de gris a 0. Cuando una discontinuidad está presente dentro de una vecindad en forma de punto, línea o borde, el resultado del Laplaciano es un valor no nulo o negativo dependiendo de donde se encuentre el punto central con respecto al borde. A fin de exhibir el resultado cuando surgen valores de pixeles tanto positivos como negativos, es común sumar un valor medio de gris (128 para el caso de imágenes con 1 sólo bit por pixel con valores de gris en el rango de 0 o 255) de modo que los puntos con valores 0 son gris medio y los valores brillantes y oscuros productos por el Laplaciano puede identificarse. [filtro-espacial]

Demostración del Procesamiento de Imagen con el Filtro Laplaciano para detección de bordes

Para esta demostración se hará uso de la librería OpenCV, ya que este cuenta con un conjunto de herramientas de procesamiento de imagen implementadas. Adicional, es considerada una de las librerías más utilizadas en este campo.

Adicional, se hará una comparación con otras dos implementaciones. Y se podrá hacer la comparación de cuál de estos realza mejor los bordes de la imagen. Los pasos seguidos para la implementación del algoritmo fueron los siguientes:

```python
def procesar_imagen_con_opencv(path_image: str):
    ddepth = cv2.CV_16S
    kernel_size = 3
    source = cv2.imread(path_image, cv2.IMREAD_COLOR)

    # Eliminación del ruido difuminado con el filtro gaussiano
    imagen_filtro_guassiano = cv2.GaussianBlur(source, (3, 3), 0)

    # Convertir la imagen a escala de grises
    imagen_escala_grises = cv2.cvtColor(imagen_filtro_guassiano, cv2.COLOR_BGR2GRAY)

    # Aplicación de la función Laplace para el realce de bordes
    imagen_procesada = cv2.Laplacian(imagen_escala_grises, ddepth, ksize=kernel_size)
    imagen_procesada = cv2.convertScaleAbs(imagen_procesada)

    return imagen_procesada
```

1. Definición de variables: Se usa un kernel de tamaño 3 y una profundidad deseada de la imagen igual a cv2.CV_16S.
2. Se procede con la carga de la imagen y se almacena en la variable source. Se le pasa a la función de lectura que al cargar la imagen esta lea en formato RGB respetando los colores de la imagen.
3. Se aplica GaussianBlur para la eliminación del ruido difuminados presenta en la imagen a procesar.

4. Para poder hacer uso del filtro, es necesario la conversión a un formato de escala de grises por lo que se procede a hacer el uso cv2.cvtColor con los parameros de la imagen resultado del filtro gaussiano y el parámetro de modificación de la imagen COLOR_BGR2GRAY.
5. Se usa la función cv2.Laplacian con los parámetros de la imagen en la escala de grises, el valor de la profundidad deseada ddpeth y el valor del kernel ksize. En esta hay que recordar que la función esta implementada mediante la siguiente formula:

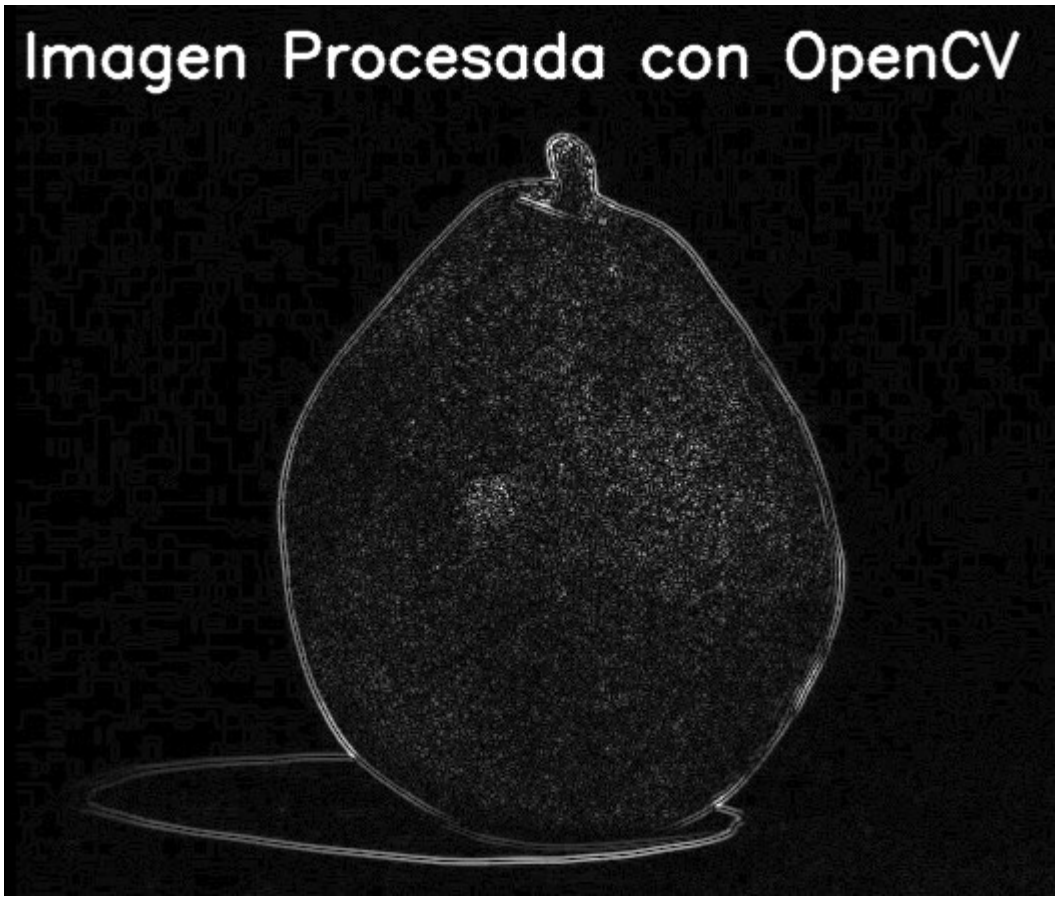
$$Laplace(f) = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

6. Se convierte los valores a formato uint8 mediante la función cv2.convertScaleAbs y listo, se tiene procesada la imagen la misma que se podrá observar con el cambio.

Resultados

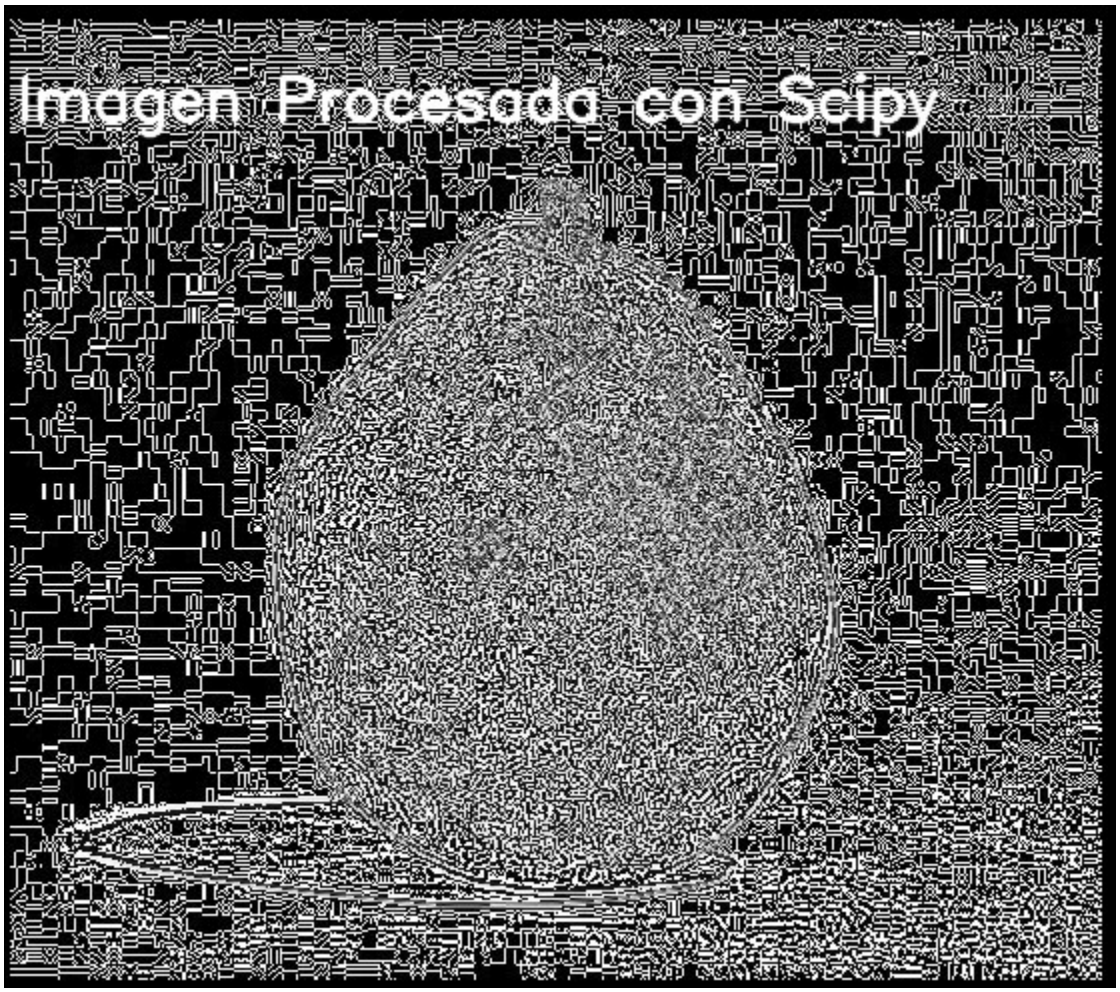
Imagen Original

Imagen Procesada



Se puede observar claramente el resultado de la aplicación del filtro Laplaciano a la imagen, pudiendo distinguir su contorno con mayor facilidad y con este resultado, se podría extraer el mismo de la imagen.

Pero el análisis de este trabajo tomo en cuenta diferentes implementaciones del filtro por lo que se procedió hacer uso de la librería Scipy para poder medir el mejor filtro para una posterior utilización. Por lo que se indica los resultados a continuación:



Como se puede apreciar en la imagen, el uso de la librería Scipy distorsiona la imagen y lo vuelve ilegible. Por lo que se puede deducir que está librería no detecta bordes del objeto, sino que detecta los bordes de toda la imagen. Dejando a esta no utilizable para una posterior toma de decisión.

Y por último se hizo el uso de un algoritmo desarrollado por nishagdhi el mismo que se puede encontrar en el repositorio de GitHub (Se adjunta URL en el notebook que acompaña a esta actividad.

Esta implementación del algoritmo, también puede distinguirse con claridad los bordes de la imagen. Pudiendo ser esta utilizada para posteriores usos.

Como conclusión final puedo decir, en escala de la mejor a la peor. El mejor algoritmo de implementación es la proporcionada por OpenCV, seguido de la implementación de nishagandhi y la peor es la proporcionada por la implementación con Scipy.

Referencias

Aldalur B., Santamaría M. (2002). Realce de imágenes: filtrado espacial. Universidad Nacional del Sur. Bahía Blanca, Argentina

Ashwin Pajankar (2022). Raspberry Pi Image Processing Programming. Nashik, Maharashtra, Indian.