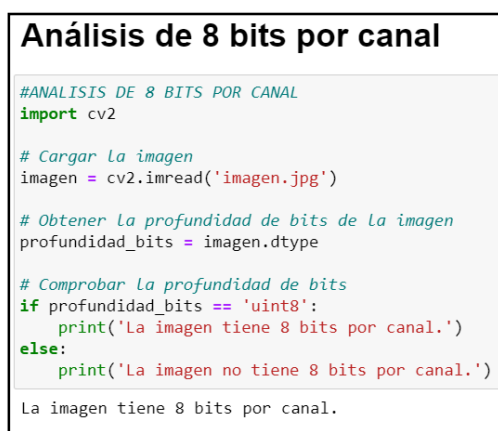


**Anexo D:****HISTOGRAMAS, ANÁLISIS DE IMÁGENES Y ECUALIZACIÓN****1. Análisis de 8 bits por canal**

Este código utiliza la biblioteca “OpenCV” para cargar una imagen “imagen.jpg” y posteriormente verifica la profundidad de bits de la imagen. Esto se refiere a la cantidad de bits utilizados para representar cada píxel en cada canal de color de la imagen. Si esta es 'uint8' (8 bits por canal), el programa imprime un mensaje indicando que la imagen tiene dicha profundidad. En caso contrario, muestra un mensaje que indica que la imagen no tiene una precisión de 8 bits por canal.

**Figura 1.**

*Código de profundidad de bits por canal.*



```
Análisis de 8 bits por canal

#ANALISIS DE 8 BITS POR CANAL
import cv2

# Cargar La imagen
imagen = cv2.imread('imagen.jpg')

# Obtener La profundidad de bits de La imagen
profundidad_bits = imagen.dtype

# Comprobar La profundidad de bits
if profundidad_bits == 'uint8':
    print('La imagen tiene 8 bits por canal.')
else:
    print('La imagen no tiene 8 bits por canal.')

La imagen tiene 8 bits por canal.
```

*Nota:* La figura muestra el código para analizar si una imagen contiene o no 8 bits por canal.

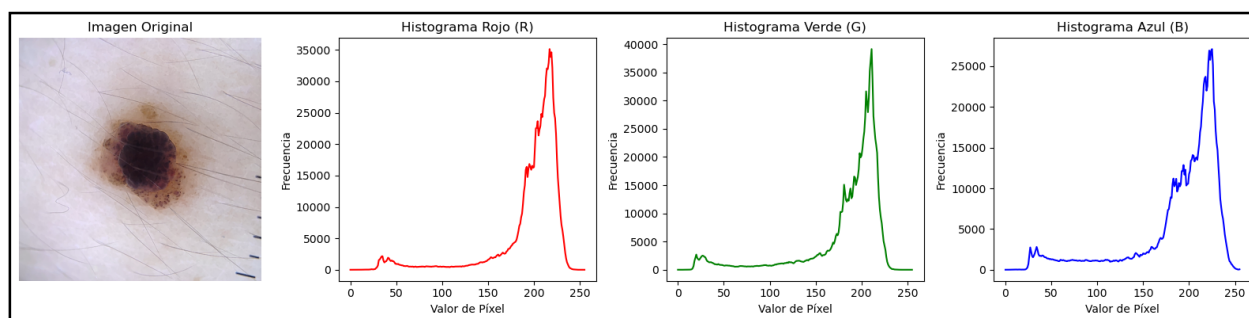
Creado por los autores.

## 2. Imágenes e histogramas RGB separados

Este código utiliza las bibliotecas “OpenCV”, “NumPy” y “Matplotlib”. Primero, la imagen se carga desde el archivo 'imagen.jpg'. Luego, se divide en sus componentes RGB, y se calculan los histogramas de intensidad para cada canal. La visualización se realiza mediante Matplotlib, en la primera columna se muestra la imagen original, mientras que las demás columnas presentan los histogramas de los canales rojo, verde y azul, respectivamente. Este enfoque permite una comprensión detallada de la distribución de intensidades en cada canal de color, facilitando el análisis visual de la imagen. Hay 3 versiones para este código donde se analizan 1, 2 y 3 imágenes al tiempo para mostrar sus resultados.

### Figura 2.

*Análisis de una sola imagen con su respectivo histograma RGB.*



*Nota:* La imagen muestra la respuesta del código para analizar el histograma RGB de una lesión.

Creado por los autores.

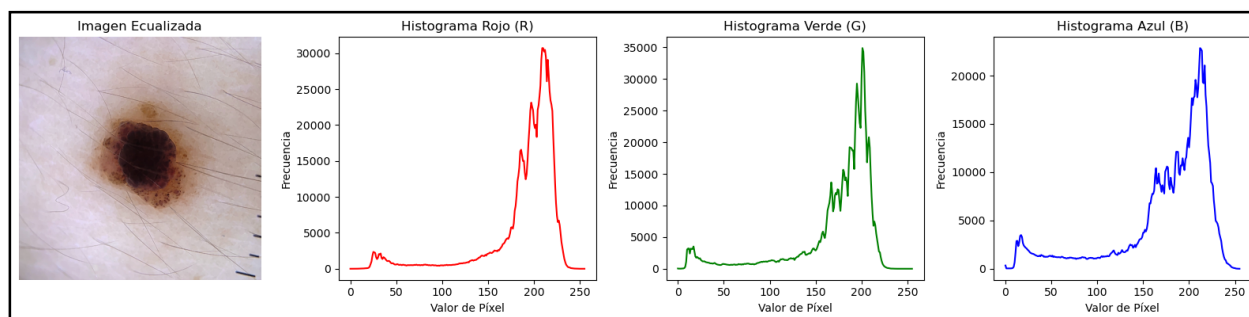
## 3. Ecualización con un parámetro gamma para toda la imagen

Este código utiliza las bibliotecas “OpenCV”, “NumPy” y “Matplotlib” para cargar una imagen RGB desde el archivo “imagen.jpg”. Luego, aplica una corrección gamma a cada canal de

color de la imagen original. La corrección gamma se realiza mediante la potenciación de los valores de píxeles de cada canal y se combinan nuevamente para formar una nueva imagen RGB corregida. Luego de ello, se calculan y visualizan los histogramas de cada canal (rojo, verde y azul) tanto para la imagen original como para la imagen corregida. La representación gráfica se organiza en dos filas: la primera fila muestra la imagen original y sus histogramas RGB, mientras que la segunda fila presenta la imagen corregida y los histogramas RGB correspondientes a la corrección gamma aplicada. Este enfoque facilita la comparación visual entre la imagen original y la corregida, así como la evaluación de cómo la corrección gamma afecta la distribución de intensidades en cada canal de color.

**Figura 3.**

*Resultado de aplicar una Ecualización con un parámetro gamma.*



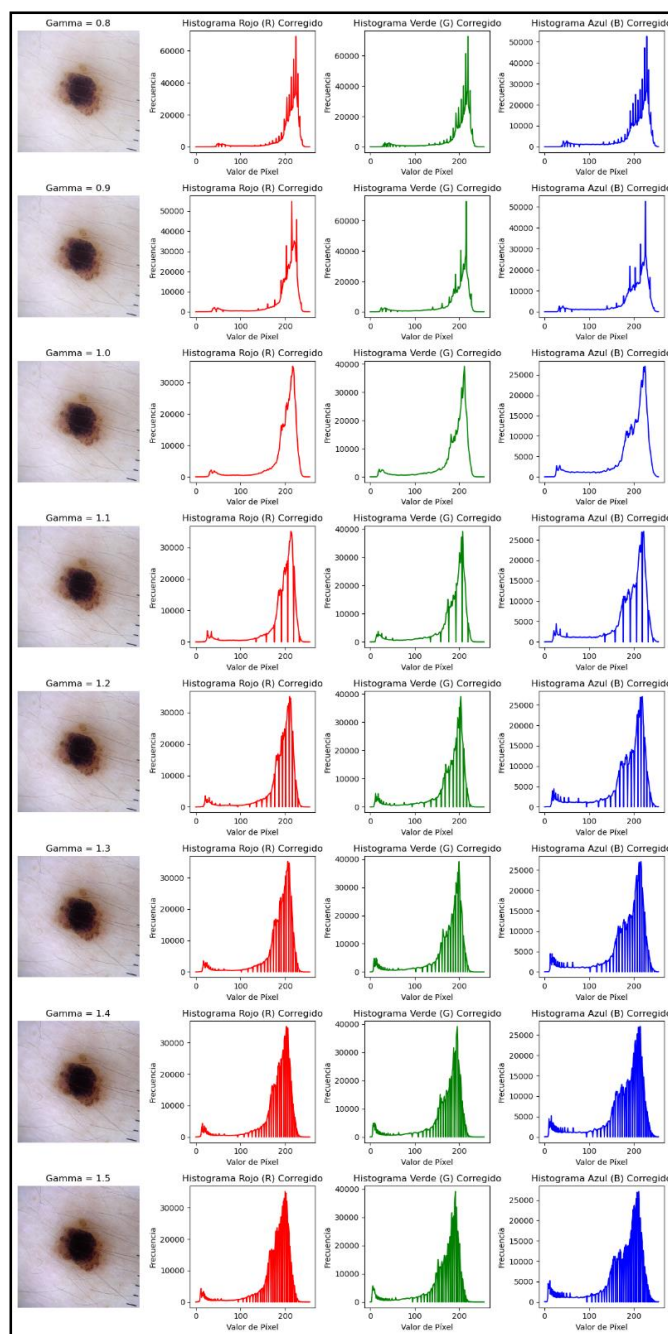
*Nota:* La figura muestra el código de ecualización con un parámetro gamma aplicado en una imagen. Creado por los autores.

#### **4. Prueba de ecualización con variación del parámetro gamma en un rango [0.8 – 1.5] con un espacio de 0.1**

Este código utiliza las bibliotecas “OpenCV”, “NumPy” y “Matplotlib” para cargar una imagen en formato BGR desde el archivo “imagen.jpg”. Luego, realiza una corrección gamma a la imagen para varios valores de gamma, generando múltiples versiones corregidas con diferentes niveles de ajuste gamma. El rango de valores de gamma se define desde 0.8 hasta 1.5 con un paso de 0.1. Para cada valor de gamma, se calcula la corrección individual para los canales de color (rojo, verde y azul) y se fusionan para formar una nueva imagen RGB corregida. Además, se calculan y visualizan los histogramas de cada canal para las imágenes corregidas. La representación gráfica se organiza en filas, donde cada fila muestra una imagen corregida con su respectivo histograma para los canales de color. Este enfoque facilita la comparación visual de cómo la corrección gamma afecta la distribución de intensidades en cada canal para distintos valores de gamma.

**Figura 4.**

*Ecualización variando el parámetro Gamma en un rango de  $[0.8 - 1.5]$  con paso de 0.1.*



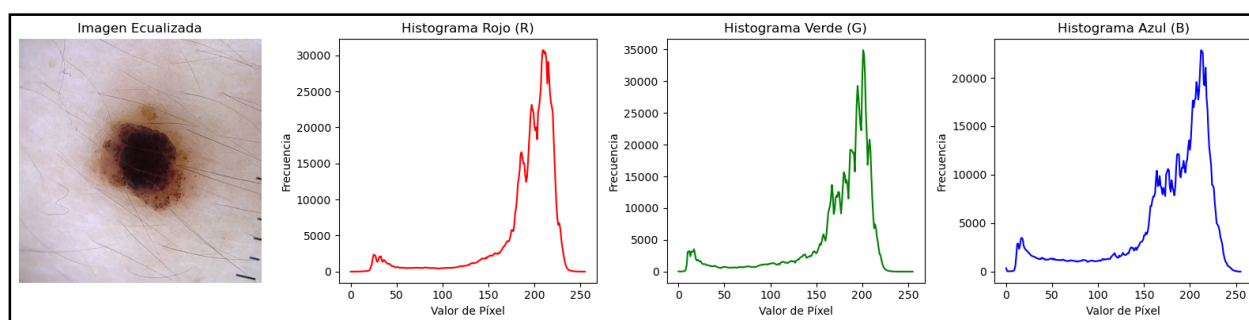
*Nota:* La imagen muestra la respuesta del código para analizar la variación del parámetro gamma aplicado en un rango, y así realizar la selección correcta del mismo para cada color. Creado por los autores.

## 5. Ecualización estándar - Parámetros de gamma individuales (R-G-B)

Este código utiliza las bibliotecas “OpenCV”, “NumPy” y “Matplotlib” para cargar una imagen en formato BGR desde el archivo “imagen.jpg”. Posteriormente, realiza una corrección gamma independiente en cada canal de color (rojo, verde y azul) con parámetros gamma específicos. Luego, muestra tanto la imagen original como la corregida en dos filas de subtramas, donde la primera fila presenta la imagen original junto con los histogramas individuales de cada canal, mientras que la segunda fila muestra la imagen corregida con sus respectivos histogramas. Este enfoque permite visualizar y comparar cómo la corrección gamma afecta la distribución de intensidades en cada canal, y proporciona una herramienta práctica para ajustar los parámetros gamma según las necesidades específicas de la imagen.

**Figura 5.**

*Imagen corregida con diferentes valores de gamma para cada canal del histograma RGB.*



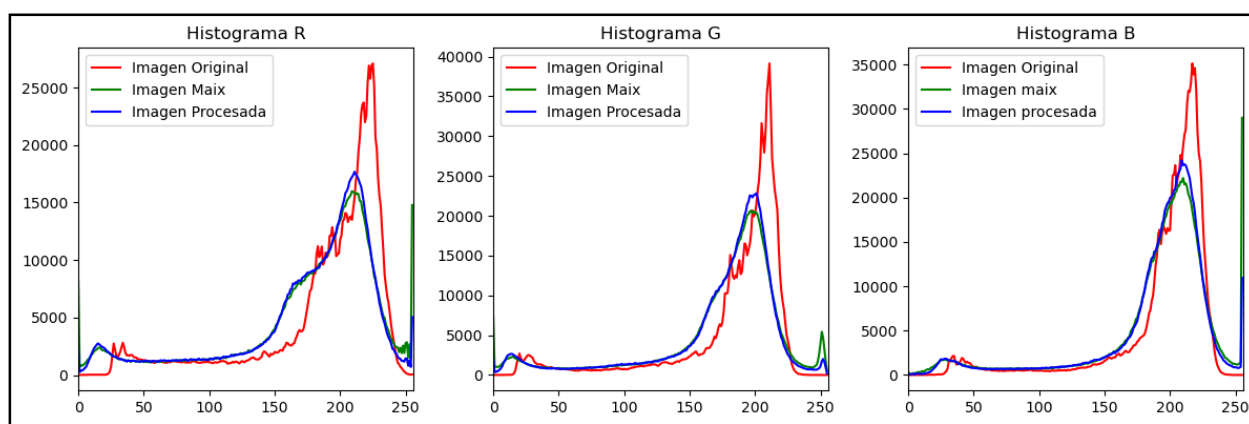
*Nota:* La imagen muestra la respuesta del código luego de aplicar un parámetro gamma a cada color seleccionado anteriormente. Creado por los autores.

## 6. Comparación de los histogramas de 3 imágenes

Este código utiliza “OpenCV” y “Matplotlib” para cargar tres imágenes diferentes: una imagen original, una imagen capturada por el módulo, y una imagen procesada. El objetivo es comparar los histogramas de los tres canales de color (rojo, verde y azul) de estas imágenes en una sola gráfica. La función “mostrar\_histogramas” toma las tres imágenes, calcula sus histogramas por canal de color y los visualiza en una figura dividida en tres subgráficas, una para cada canal. Cada subgráfica muestra las curvas de los histogramas de las tres imágenes, diferenciando cada una con colores distintos (rojo para la imagen original, verde para la imagen del módulo y azul para la imagen procesada). Esta representación gráfica facilita la comparación visual de las distribuciones de intensidad de cada canal en las tres imágenes, proporcionando una herramienta para analizar y contrastar los efectos de las operaciones de procesamiento aplicadas.

**Figura 6.**

*Comparación de los histogramas RGB de 3 imágenes.*



*Nota:* El gráfico muestra la comparación de los histogramas de tres imágenes para observar las características de las mismas. Creado por los autores.

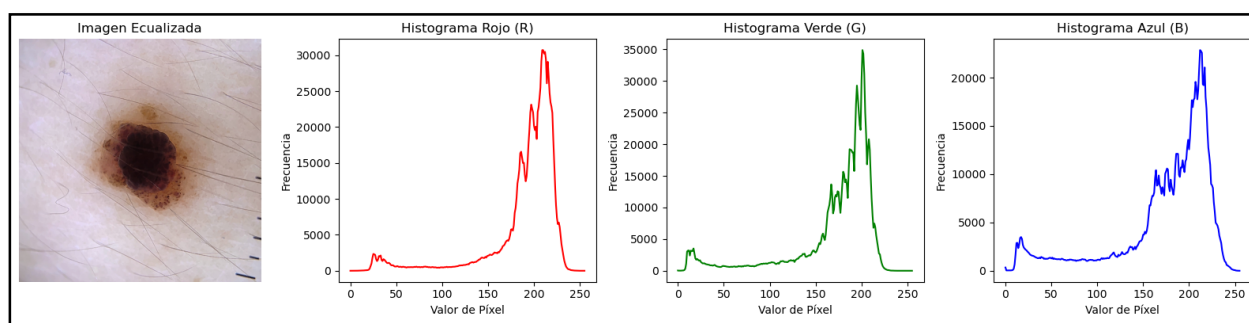
## PROCESAMIENTO DE IMAGENES

### 7. Ecualización de una imagen (será guardada automáticamente como "imagen\_ecualizada")

Este código utiliza la biblioteca “OpenCV” y “Matplotlib” para realizar la corrección gamma en los canales de color de una imagen RGB. Comienza cargando una imagen en formato BGR (el formato predeterminado de OpenCV) y separa sus canales de color (rojo, verde y azul). Luego, define parámetros gamma específicos para cada canal y aplica la corrección gamma a cada uno de ellos. Los canales corregidos se combinan para formar una nueva imagen RGB corregida. A continuación, se calculan y muestran los histogramas para cada canal de la imagen original y la imagen corregida. Este proceso permite ajustar la intensidad de cada canal de color de manera independiente, lo que puede ser útil para realzar o corregir ciertos aspectos visuales de una imagen.

**Figura 7.**

*Ecualización de imagen con corrección gamma para cada canal.*



*Nota:* La figura muestra la respuesta de la imagen luego de aplicar una “ecualización estándar”.

Creado por los autores.

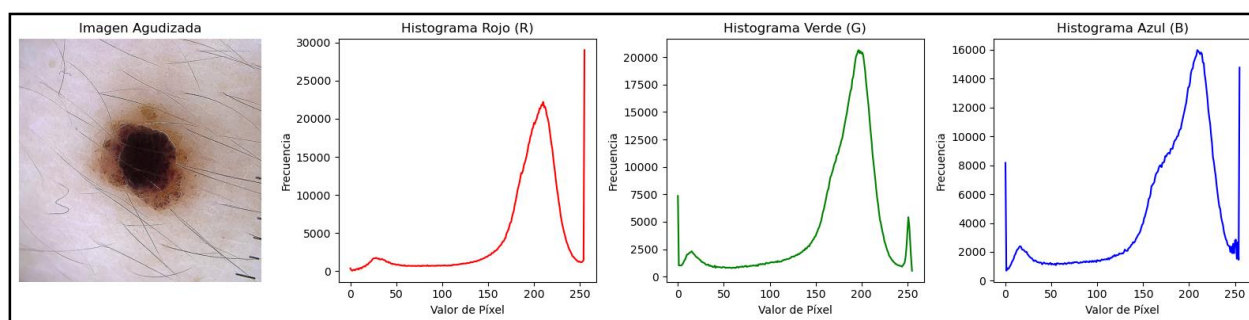


## 8. Agudización de una imagen (será guardada automáticamente como "imagen\_agudizada")

Este código utiliza la biblioteca “OpenCV” y “Matplotlib” para aplicar un filtro de agudización a una imagen. Comienza cargando la imagen ecualizada en formato BGR (el formato predeterminado de OpenCV) y define un kernel de agudización. Luego, aplica este kernel a la imagen mediante la convolución bidimensional usando la función “filter2D” de “OpenCV”. Posteriormente, calcula y muestra los histogramas de los canales de color (rojo, verde y azul) tanto para la imagen original como para la imagen agudizada. La disposición de subtramas facilita la comparación visual entre la imagen original y la imagen agudizada, así como la observación de los cambios en los histogramas. La agudización resalta los detalles y bordes en la imagen, lo que puede mejorar la nitidez y la claridad de los objetos presentes.

**Figura 8.**

*Agudización de una imagen.*



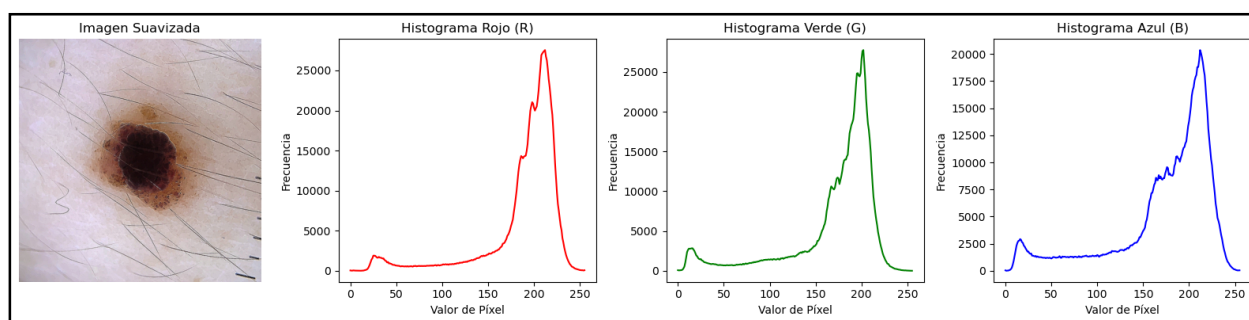
*Nota:* La figura muestra la respuesta de una imagen luego de aplicar una agudización. Creado por los autores.

## 9. Suavizado de una imagen (será guardada automáticamente como "imagen\_suavizada")

Este código utiliza las bibliotecas “OpenCV” y “Matplotlib” para aplicar un filtro de suavizado a una imagen. La imagen agudizada se carga en formato BGR, y se crea un kernel de suavizado, que en este caso es un filtro de promedio para cada canal de color. El filtro se aplica a la imagen mediante la convolución bidimensional usando la función “filter2D” de “OpenCV”. Luego, se calculan y visualizan los histogramas de los canales de color (rojo, verde y azul) tanto para la imagen original como para la imagen suavizada. La disposición de subtramas facilita la comparación visual entre la imagen original y la suavizada, así como la observación de los cambios en los histogramas. El suavizado tiene el efecto de reducir el contraste y atenuar los detalles finos en la imagen, resultando en una apariencia más difuminada y menos nítida.

**Figura 9.**

*Suavizado de una imagen.*



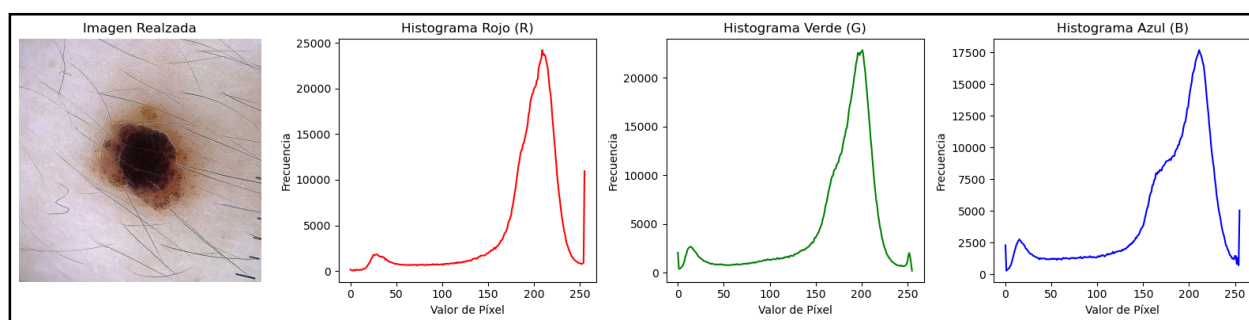
*Nota:* La figura muestra la respuesta de una imagen luego de aplicar un suavizado. Creado por los autores.

## 10. Realce de características en una imagen (será guardada automáticamente como "imagen\_realzada")

Este código utiliza “OpenCV”, “NumPy” y “Matplotlib” para aplicar un filtro de realce de características a una imagen suavizada. Primero, se carga la imagen suavizada, y luego se define un kernel de realce que resalta las características de la imagen. El filtro de realce se aplica a la imagen mediante la convolución bidimensional utilizando la función “filter2D” de “OpenCV”. Después, se calculan y visualizan los histogramas de los canales de color (rojo, verde y azul) tanto para la imagen original como para la imagen realzada. La disposición de subtramas permite la comparación visual entre la imagen suavizada y la realzada, así como la observación de los cambios en los histogramas. El realce de características tiene el efecto de resaltar los detalles y bordes en la imagen, lo que puede mejorar la visibilidad de ciertos elementos.

**Figura 10.**

*Realce de características de una imagen.*



*Nota:* La figura muestra la respuesta de una imagen luego de aplicar un realce de características.

Creado por los autores.