Eduardo Leonel Sanchez Velasco
420004035
leonel2711@ciencias.unam.mx

1 Objetivos

- Realizar diversas modificaciones al histograma con transformaciones básicas como son: negativo, exponencial, logarítmica y gamma.imagen.
- Ecualización el histograma de una imagen.
- Realizar operaciones de suavizado y de reducción de ruido en imágenes utilizando filtros espaciales.
- Realizar operaciones de detección de bordes en imágenes, tanto limpias como ruidosas, utilizando filtros basados en aproximaciones de gradientes y laplacianos.

2 Introducción

- Los histogramas constituyen la base de varias técnicas de procesamiento en el dominio espacial. La manipulación de los histogramas es usada de manera eficiente en el realce o mejoramiento de la calidad de una imagen. La información estadistica obtenida a partir de los histogramas se utiliza en diversas aplicaciones como compresion y segmentacion de imagenes. La facilidad con la que se pueden calcular los histogramas usando software y su bajo consumo de recursos de hardware en su implementacion, han hecho de esta herramienta una de las mas usadas en el procesamiento en tiempo real.
  El histograma de una imagen es la representacion grafica de la distribucion que existe de las distintas tonalidades de grises con relacion al numero de pixeles o porcentaje de los mismos, es decir, un histograma representa la frecuencia relativa de ocurrencia de los niveles de gris. La representacion de un histograma ideal seria la de una recta horizontal, ya que eso nos indicaria que todos los posibles valores de grises estan distribuidos de manera uniforme en nuestra imagen.
  La ecualización del histograma es una tecnica bastante conocida y sirve para obtener un histograma uniforme de tal manera que los niveles de gris son distribuidos sobre la escala y un numero igual de pixeles son colocados en cada nivel de gris. Para un observador, esta ecualizacion hace que las imagenes se vean mas balanceadas y con mejor contraste. Como consecuencia, una imagen ecualizada, permite que ciertos detalles sean visibles en regiones oscuras o brillantes.
- Los filtros espaciales tienen como objetivo modificar la contribucion de determinados rangos de frecuencias de una imagen. El termino espacial se refiere a que el filtro se aplica directamente a la imagen y no a una transformada de la misma, es decir, el nivel de gris de un pixel se obtiene directamente en funcion del valor de sus vecinos. La convolucion es la operacion con la cual se hace filtrado espacial. Los filtros espaciales pueden clasificarse basandose en su linealidad en filtros lineales y en filtros no lineales. A su vez los filtros lineales pueden ser

3 Desarrollo

Resuelve los problemas de la lista siguiente y describe tu solución en cada inciso. Los incisos en donde únicamente tengas que desplegar imágenes no requieren de ninguna descripción.

1. Aplicar a una imagen las diferentes transformaciones : negativa, logaritmica, exponencial y gama.

a) Desplegar la imagen original y su histograma (Nota: si trabajas con MATLAB utilizas la función subplot y usa títulos en sus gráficas con title). b) Aplicar cada una de las transformaciones referidas a la imagen leída en (a) y desplegar en la misma ventana la imagen resultado con su respectivo histograma.
c) Tu programa deberá ser capáz de tener como entrada diferentes imágenes. Cada transformación resultado deberá ser desplegada en una ventana a parte, de manera que tendrás 5 ventanas: una con la imagen original y otras 4 cada una con las transformaciones referidas.
d) Ecualizar el histograma una imagen. Desplagar en la misma ventana: imagen original, su histograma, imagen ecualizada y su histograma.
e) Nota: Si trabajas en MATLAB está prohibido utilizar las funciones: imhist, imadjust, imcontrast, histeq.

2. Aplicar a una imagen sin ruido y la misma imagen con ruido filtros de suavizamiento y realce. La imagen con ruido se puede generar a partir de la imagen sin ruido usando el siguiente comando de MATLAB:J=imnoise $(\mathbf{I}, \mathbf{TIPO}, \ldots)$, donde TIPO es una cadena que puede tomar valores 'gaussian', 'salt & peper', etc. Ver comando help imnoise.

a) Aplicar los filtros paso bajas promedio estándar a la imagen sin ruido y a la imagen con ruido usando filtros de orden $3 \times 3, 5 \times 5$, 7x7 y 11x11.
b) Aplicar los filtros paso bajas promedio ponderado a la imagen sin ruido y a la imagen con ruido usando filtros de orden $3 \times 3, 5 \times 5$, 7x7 y 11x11.
c) Aplicar a la imagen con ruido el filtro mediana de 3x3, 5x5, 7x 7y 11x11. Utilizar ruido 'sal y pimienta' y 'gaussiano'. Compararlos.
d) Aplicar a la imagen sin ruido y con ruido los filtros basados en la primera derivada o detectores de borde siguientes :

- Prewitt en la dirección X y en la dirección Y, asi como su magnitud.
- Sobel en la dirección X y en la dirección Y, asi como su magnitud.

e) Aplicar a una imagen el realce basado en las segundas derivadas (Laplaciano). Isotrópicos a 45 grados así como a 90 grados. f) Difuminar las imágenes sin ruido y con ruido usando un filtro paso bajas de orden 5×5, de tal manera que se obtenga una imagen sin ruido y con pérdida de nitidez y otra imagen con ruido y perdida de nitidez. Para cada uno de los siguientes incisos, filtrar las imágenes utilizando el filtro unsharp masking encontrado con los siguientes tipos de filtro paso bajas:

- Filtro paso bajas promedio estándar de orden 3x3 y 7x7.
- Filtro paso bajas promedio ponderado de orden 3x3 y 7x7.

g) Nota: Si trabajan en MATLAB está prohibido utilizar la función: conv2.

4 Código

A continuación muestro las funciones creadas para esta practica junto a los resultados obtenidos:
Primero se crean unas funciones de apoyo para poder mostrar las imágenes en consola junto con su histograma, al igual de objetos de apoyo para poder realizar los ejercicios de manera automatizada:

```
##Objetos y funciones globales necesarios
imagenes = {}
def get_histograma(img):
    img2 = img * 255
    y_space, x_space = img.shape
    tonalidades = dict.fromkeys(list(range(256)),0)
    for y in range(y_space):
        for x in range(x_space):
            tonalidades[img2[y][x]] += 1
    return tonalidades
def plot_img_histo(img,histograma):
    name, image = img
    fig = plt.gcf()
    fig.set_size_inches(10,5)
    plt.subplot(1,2,1)
    plt.title(name)
    plt.imshow(image)

    plt.subplot(1,2,2)
    plt.title(name)
    plt.bar(range(len(histograma)), list(histograma.values()), align='center',color = 'black')
    plt.show()
```

Figure 1: Funciones Globales1

```python
def plot_imgs_histos(img):
    name, imagenes = img
    sizes = (3,5,7,11)
    j = 0
    for img2 in imagenes:
        histograma = get_histograma(img2)
        plot_img_histo((name+' {}x{}'.format(sizes[j],sizes[j]),img2),histograma)
        j += 1


def gaussian_noise(img):
    img2 = img.copy() * 255
    noise = np.random.normal(0, 50, img2.shape)
    img_noised = img2 + noise
    img_noised = np.clip(img_noised, 0, 255)
    return np.rint(img_noised)/255
```

Figure 2: Funciones Globales2

```python
def salt_pepper_noise(img):
    img2 = img.copy() *255
    y_space, x_space = img.shape
    pepper = 0.005
    salt = 1 - pepper
    for y in range(y_space):
        for x in range(x_space):
            rdn = np.random.random()
            if rdn < pepper:
                img2[y][x] = 0
            elif rdn > salt:
                img2[y][x] = 1
    return img2/255
```

Figure 3: Funciones Globales3

Para la primera parte del ejercicio, debido a que estas transformaciones deben de cumplir la condición de que cada nivel de intensidad de entrada debe de ser ”ploteda” al mismo rango de intensidades, en las transformaciones buscamos la constante que nos permita cumplir con esa condición.

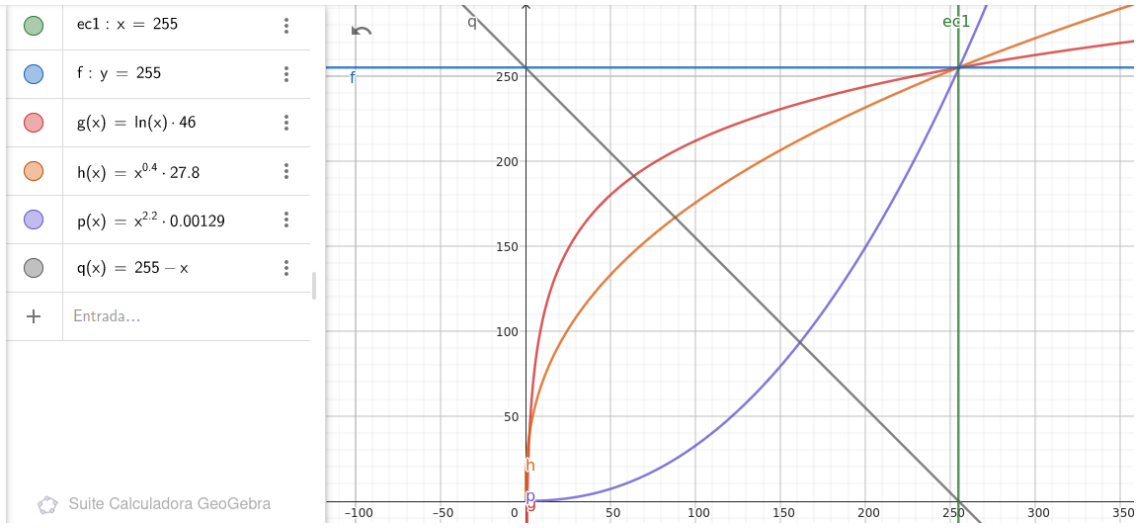
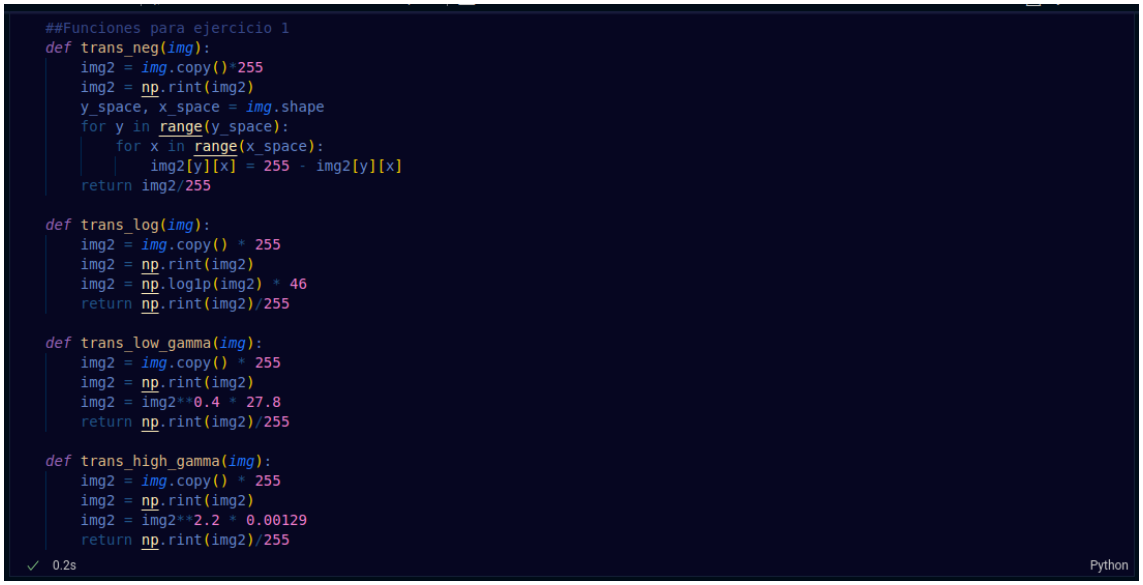


Figure 4: Rango de intensidades

Así nuestras funciones quedan de la siguiente manera:



```python
##Funciones para ejercicio 1
def trans_neg(img):
    img2 = img.copy()*255
    img2 = np.rint(img2)
    y_space, x_space = img.shape
    for y in range(y_space):
        for x in range(x_space):
            img2[y][x] = 255 - img2[y][x]
    return img2/255

def trans_log(img):
    img2 = img.copy() * 255
    img2 = np.rint(img2)
    img2 = np.log1p(img2) * 46
    return np.rint(img2)/255

def trans_low_gamma(img):
    img2 = img.copy() * 255
    img2 = np.rint(img2)
    img2 = img2**0.4 * 27.8
    return np.rint(img2)/255

def trans_high_gamma(img):
    img2 = img.copy() * 255
    img2 = np.rint(img2)
    img2 = img2**2.2 * 0.00129
    return np.rint(img2)/255
```


Figure 5: Funciones ejercicio 1 primera parte

Para la ecualización, creamos las funciones que nos den el pdf y el cdf:

```python
## Ecualizacion
def get_acumulado(p_histo):
    acumulado = 0
    acumulados = dict.fromkeys(list(range(256)),0)
    for tonailidad in p_histo.keys():
        acumulado += p_histo[tonailidad]
        acumulados[tonailidad] = acumulado
    return acumulados
def get_probabiliades(histograma, size):
    probabilidades = dict.fromkeys(list(range(256)),0)
    for tonalidad in histograma.keys():
        probabilidades[tonalidad] = histograma[tonalidad]// size
    return probabilidades

def get_img_ec(img):
    histograma = get_histograma(img)
    size = np.prod(img.shape)
    cdf = get_acumulado(get_probabiliades(histograma,size))
    img2 = img.copy() *255
    y_space, x_space = img2.shape
    for y in range(y_space):
        for x in range(x_space):
            img2[y][x] = cdf[img2[y][x]]
    return img2
```


Figure 6: Funciones Ecualización

Para el ejercicio 2, se crean las funciones para aplicar el filtro, independientemente de cual sea el filtro a utilizarse:

```python
def convolution(section, filter):
    convol = np.multiply(section,filter)
    return abs(convol.sum())


def apply_filter(img,filter):
    fileter_size = filter.shape[0]
    padding = fileter_size // 2
    img2 = img.copy() * 255
    img3 = np.pad(img2, pad_width=padding, mode='constant', constant_values=0)
    y_space, x_space = img3.shape
    for y in range(padding,y_space-padding):
        for x in range(padding,x_space-padding):
            section = img3[y-padding:y+fileter_size-padding, x-padding:x+fileter_size-padding]
            img2[y-padding][x-padding] = convolution(section,filter)
    return np.clip(np.rint(img2),0,255)/255
```


Figure 7: Funciones para aplicar los filtros con convolución

Así solo es necesario crear los filtros para los cuales queremos aplicar a las imágenes. Se crea para los filtros ponderados una función que nos de este filtro dependiendo el tamaño que queramos que sea:

```python
def make_filter(shape):
    return np.ones(shape)
def make_filter_p(shape):
    filter =  np.ones(shape)
    y_space, x_space = shape
    y_aux, x_aux = y_space-1, x_space-1
    y_2 = y_aux
    aux = y_space//2
    for y in range(y_space):
        x_aux = x_space-1
        x_2 = x_aux
        for x in range(x_space):
            filter[y][x] = 2**(y_aux - np.clip(y_2,0,y_space)) * 2**(x_aux - np.clip(x_2,0,x_space))
            x_aux -= 1
            x_2 -= 2
        y_aux -= 1
        y_2 -=2
    return filter
```

✓ 0.7s Python

Figure 8: Funciones para crear los filtros estándar y ponderados

Por ultimo se crean las funciones para el filtro mediana y el método unsharp, ya que estos no hacen uso de convolución.

```python
def mediana(section):
    lista = section.flatten()
    return np.median(lista)

def filtro_mediana(img,size):
    padding = size // 2
    img2 = img.copy() * 255
    img3 = np.pad(img2, pad_width=padding, mode='constant', constant_values=0)
    y_space, x_space = img3.shape
    for y in range(padding,y_space-padding):
        for x in range(padding,x_space-padding):
            section = img3[y-padding:y+size-padding, x-padding:x+size-padding]
            img2[y-padding][x-padding] = mediana(section)
    return np.rint(img2)/255
```

✓ 0.8s Python

Figure 9: Función mediana

```python
def unsharp_masking(img,filter,k):
    blur_img = apply_filter(img,filter)
    mask = np.subtract(img *255,blur_img *255)
    return np.clip(np.rint(np.add(img*255,k*mask)),0,255)/255
```

✓ 0.1s Python

Figure 10: Función unsharp

A continuación se muestran las imágenes resultado, junto a su histograma para el ejercicio 1, no se mostraran todas debido a que son demasiadas imágenes, para las imágenes en las que se resalta los bordes se mostraran las de tugsteno debido a que solo las que mejor resultado muestran.

- Imágenes originales:

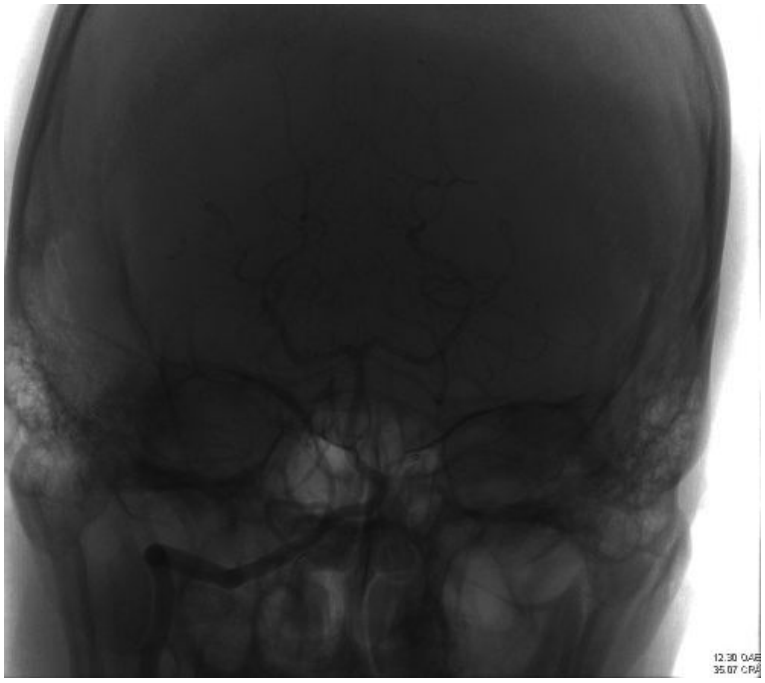


Figure 11: Imagen original

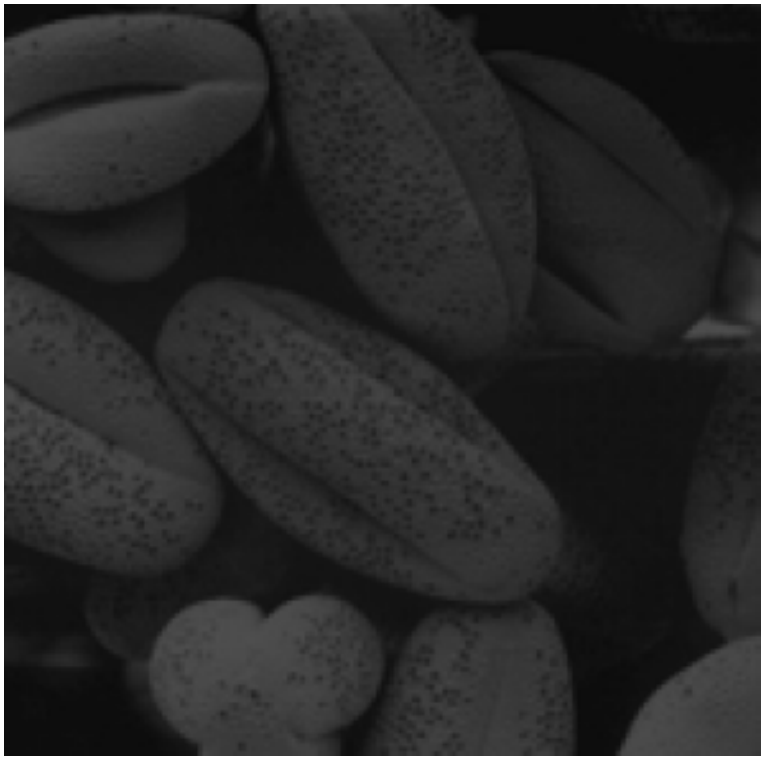


Figure 12: Imagen original

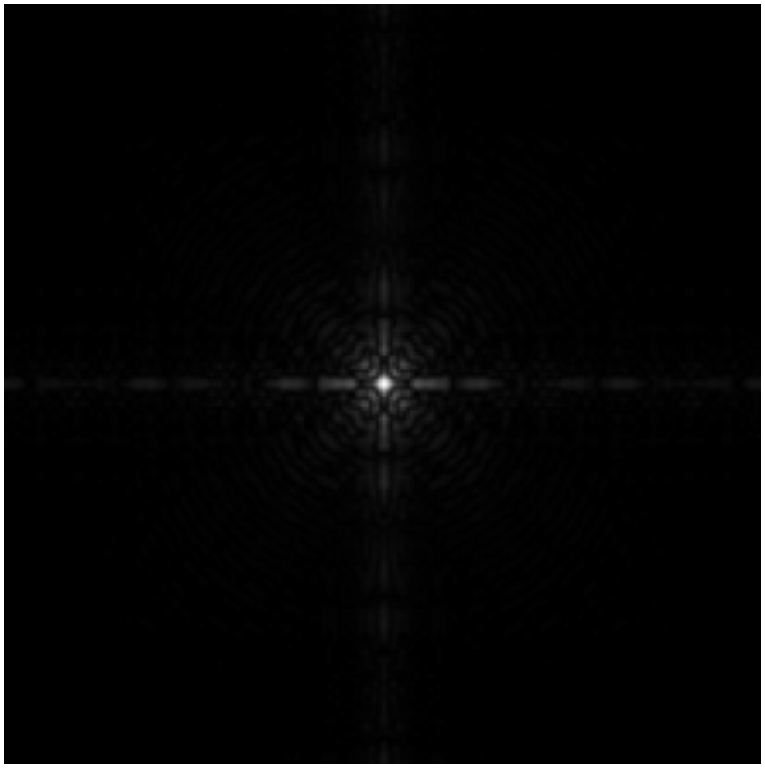


Figure 13: Imagen original

Figure 14: Imagen original

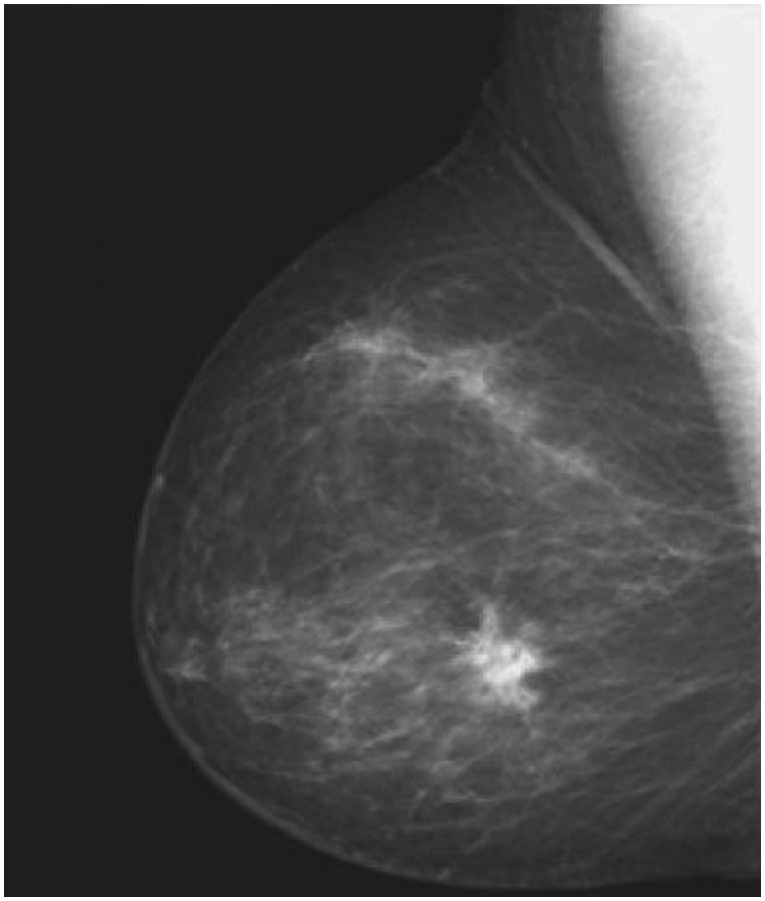


Figure 15: Imagen original

Figure 16: Imagen original

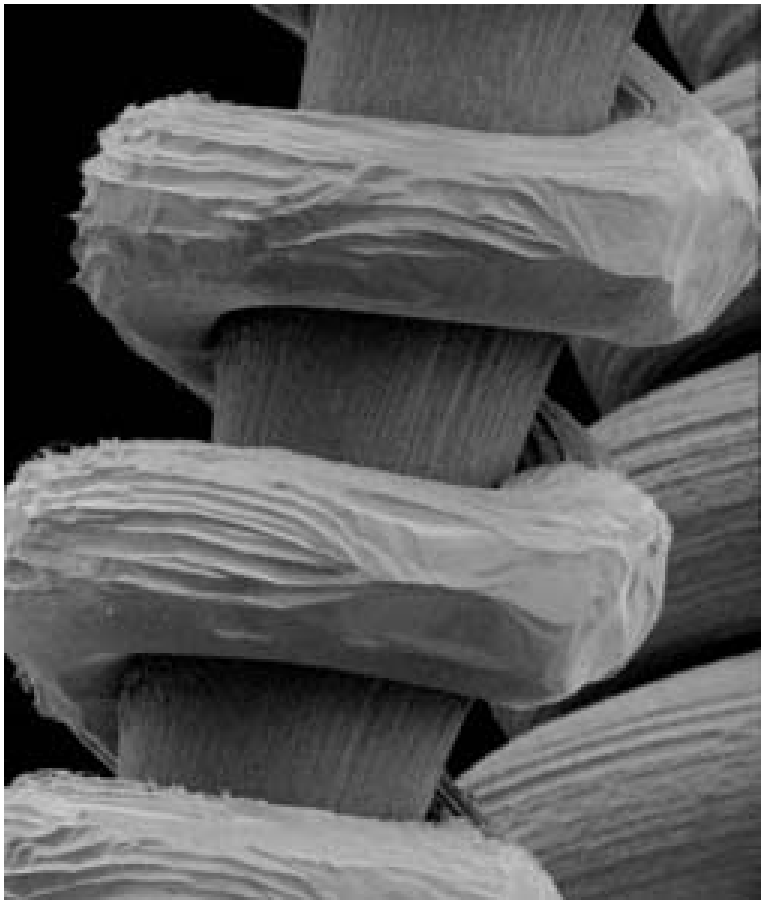


Figure 17: Imagen original

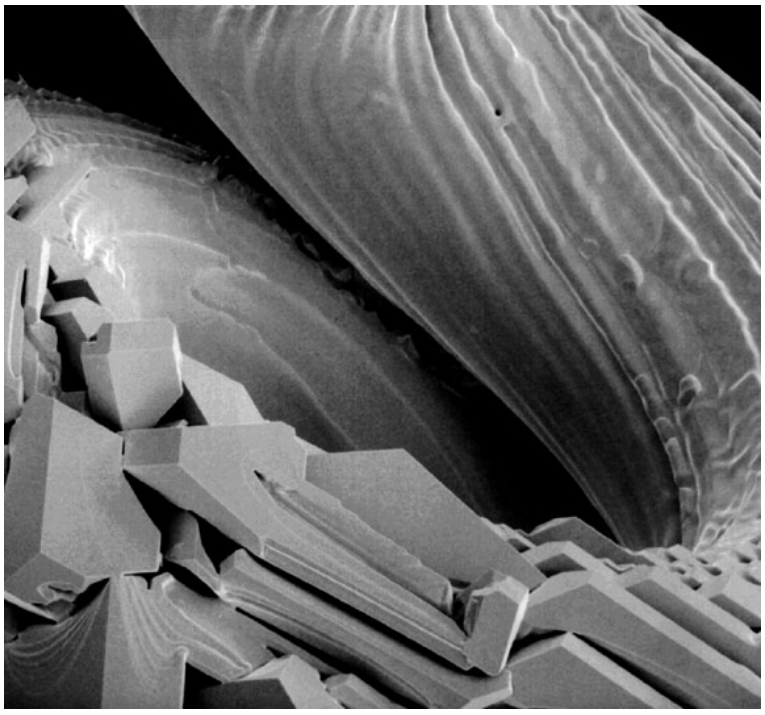


Figure 18: Imagen original

- Imagenes negativas

Figure 19: Imagen negativa

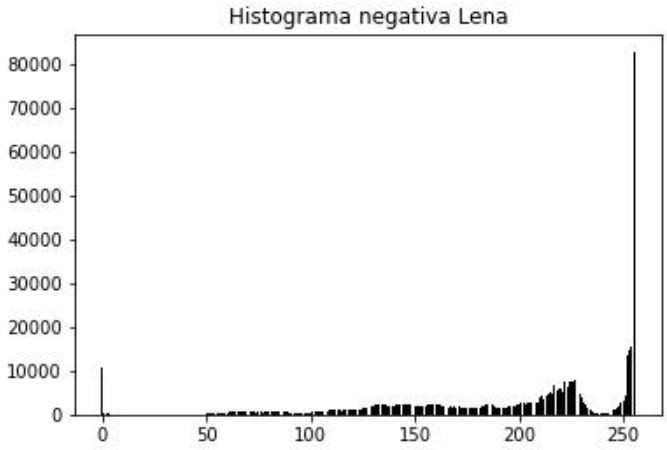


Figure 20: Histograma de imagen negativa

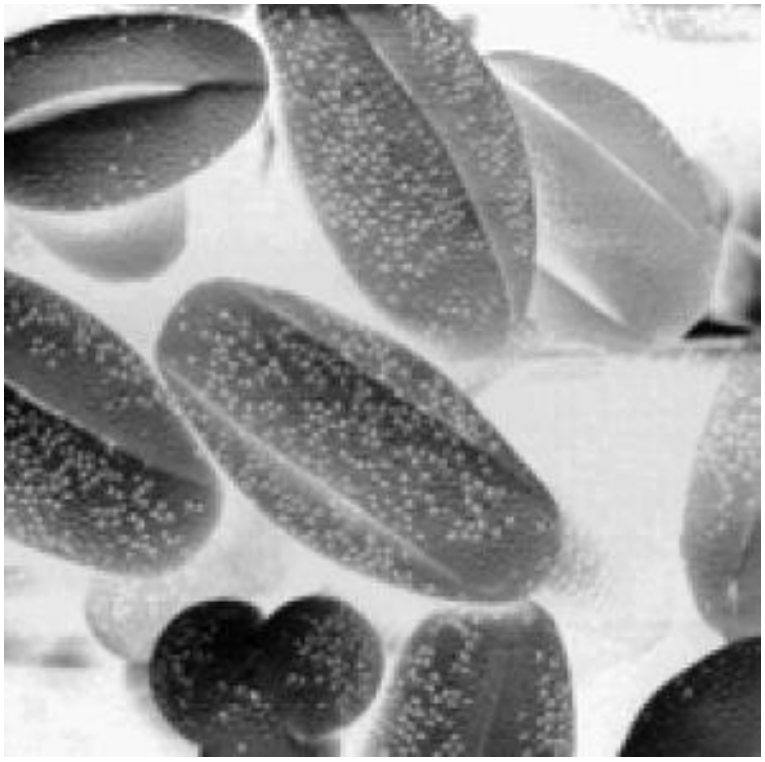


Figure 21: Imagen negativa

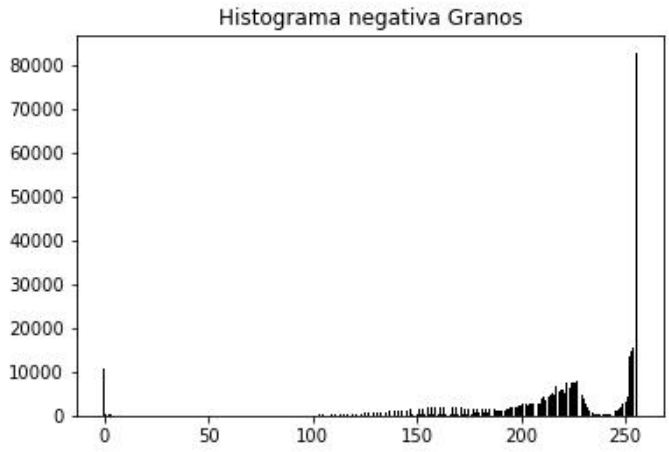


Figure 22: Histograma de imagen negativa

- Imágenes aplicando filtro logarítmico

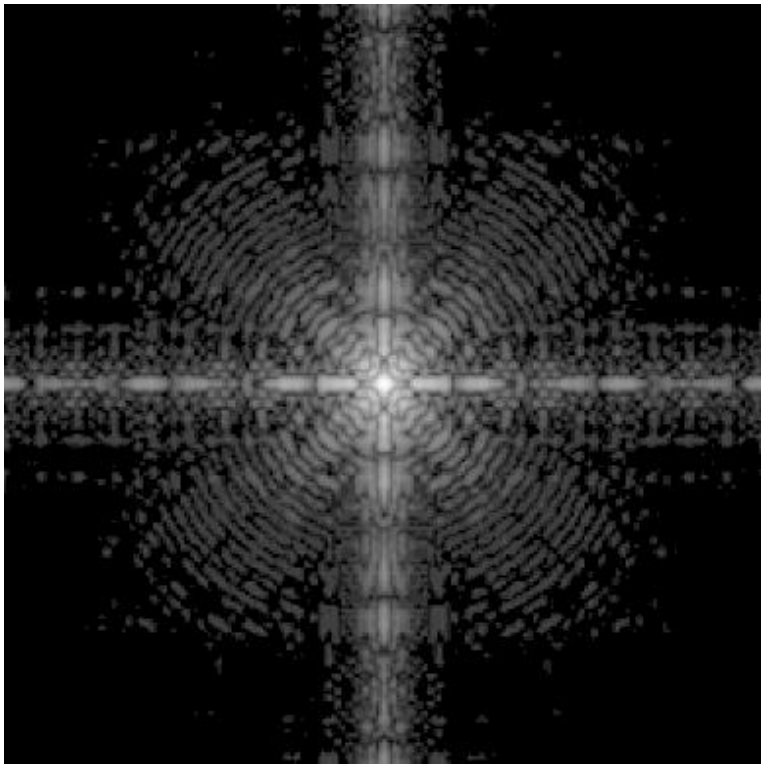


Figure 23: Imagen filtro logarítmico

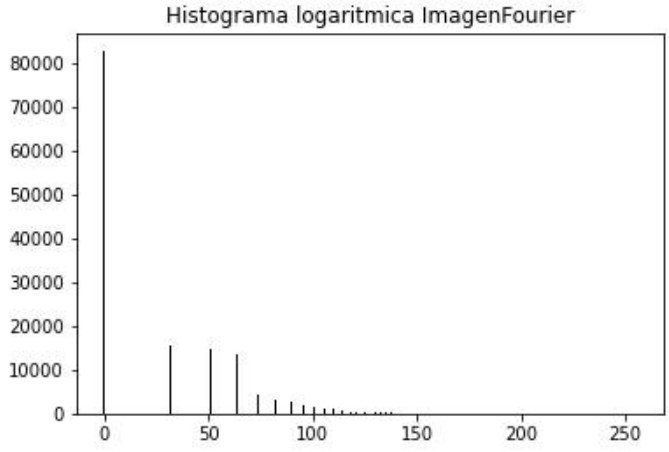


Figure 24: Histograma filtro logarítmico

Figure 25: Imagen logaritmico

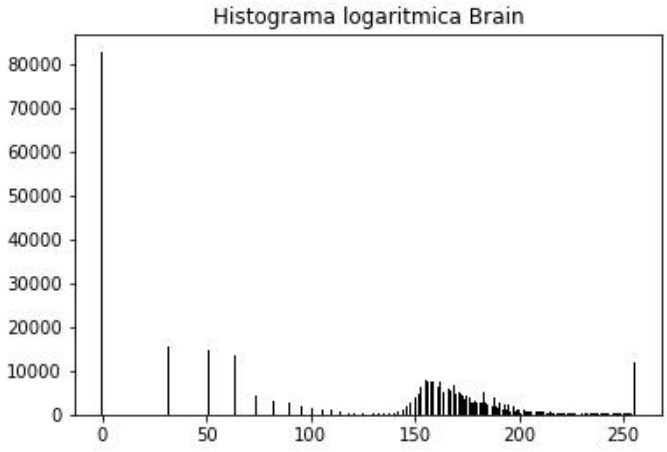


Figure 26: Histograma logaritmico

- Imágenes usando filtro low gamma

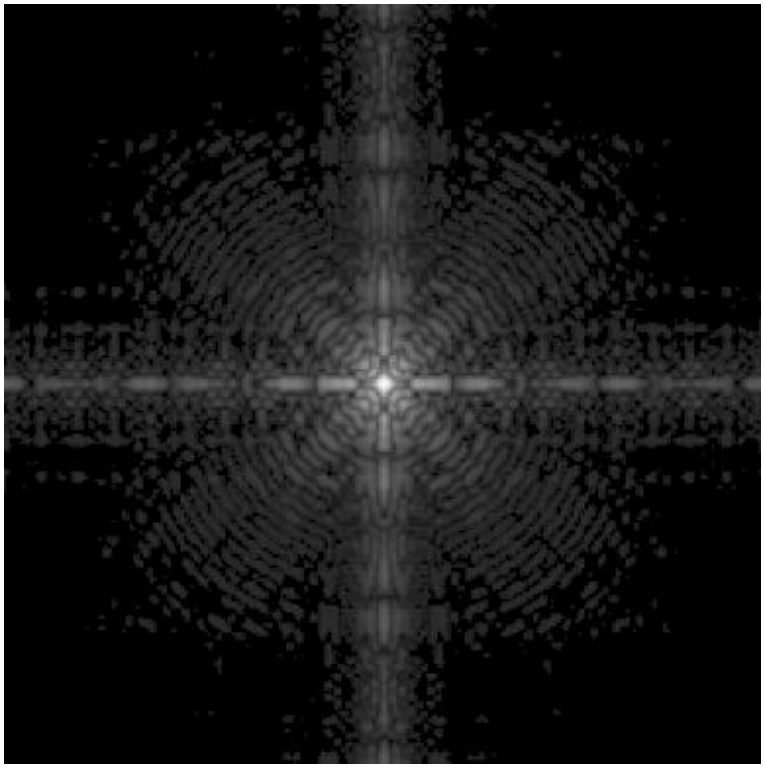


Figure 27: Imagen filtro low gamma

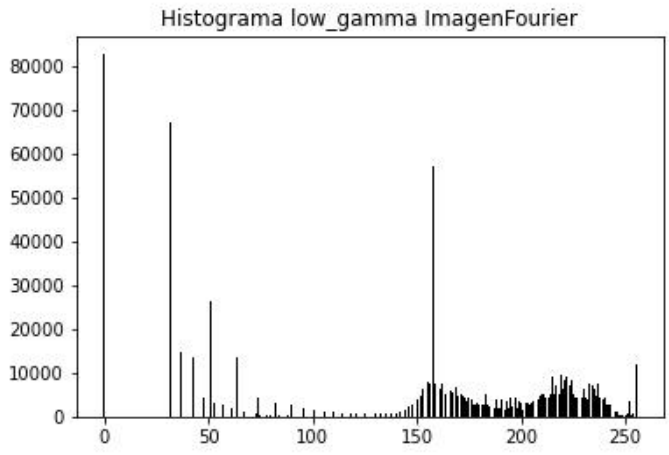


Figure 28: Histograma imagen filtro low gamma

- Imágenes high gamma

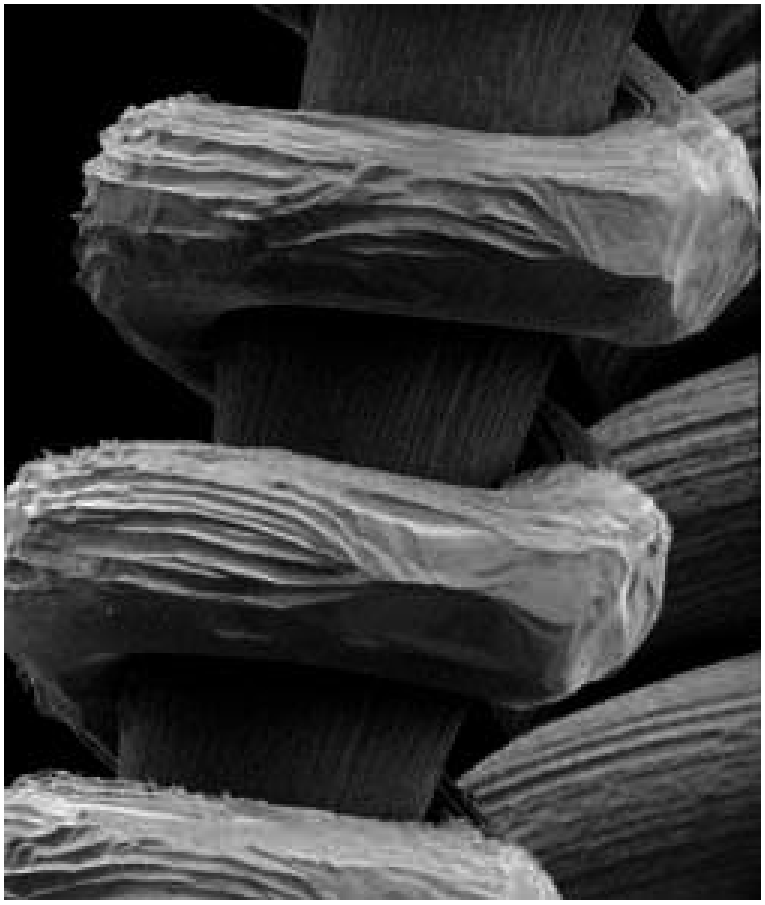


Figure 29: Imagen filtro high gamma

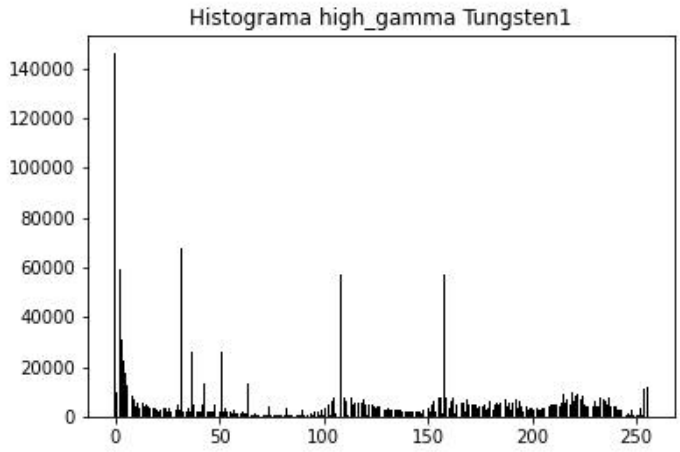


Figure 30: Histograma imagen filtro high gamma

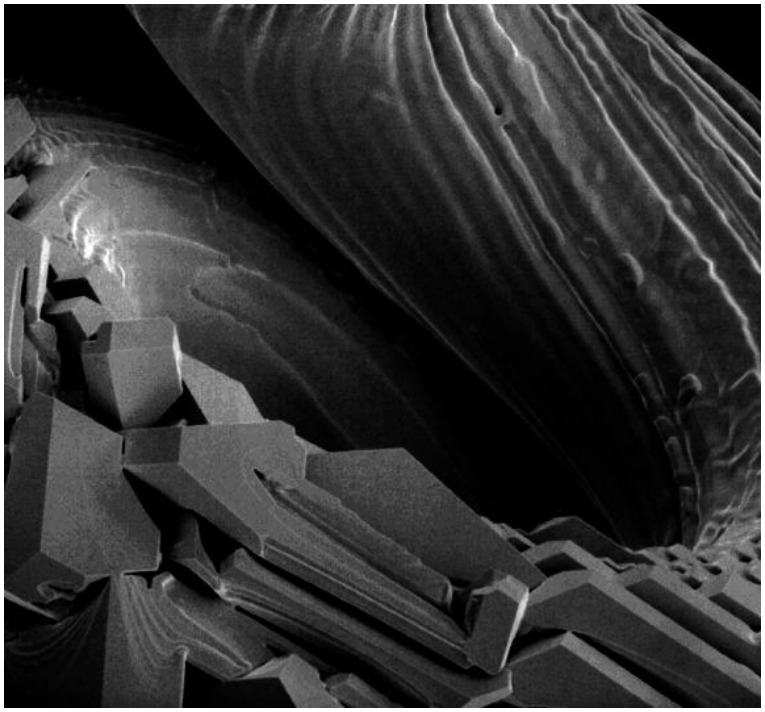


Figure 31: Imagen filtro high gamma

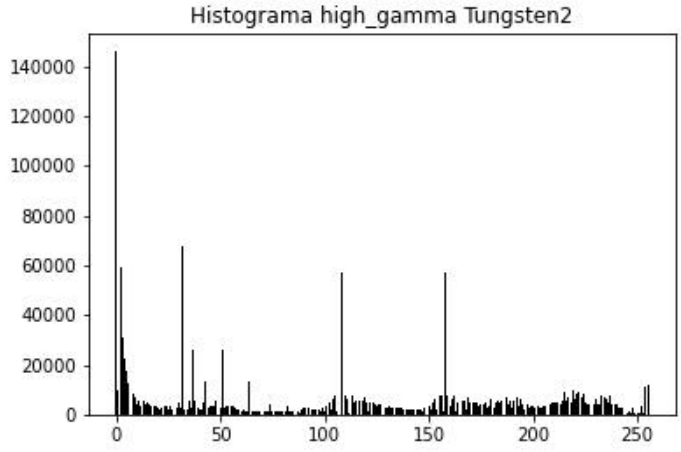


Figure 32: Histograma imagen filtro high gamma

- Imágenes con filtro estándar

Figure 33: Imagen filtro estándar 3x3

Figure 34: Imagen con ruido filtro estándar 3x3

- Imágenes con filtro ponderado

Figure 35: Imagen filtro ponderado 5x5

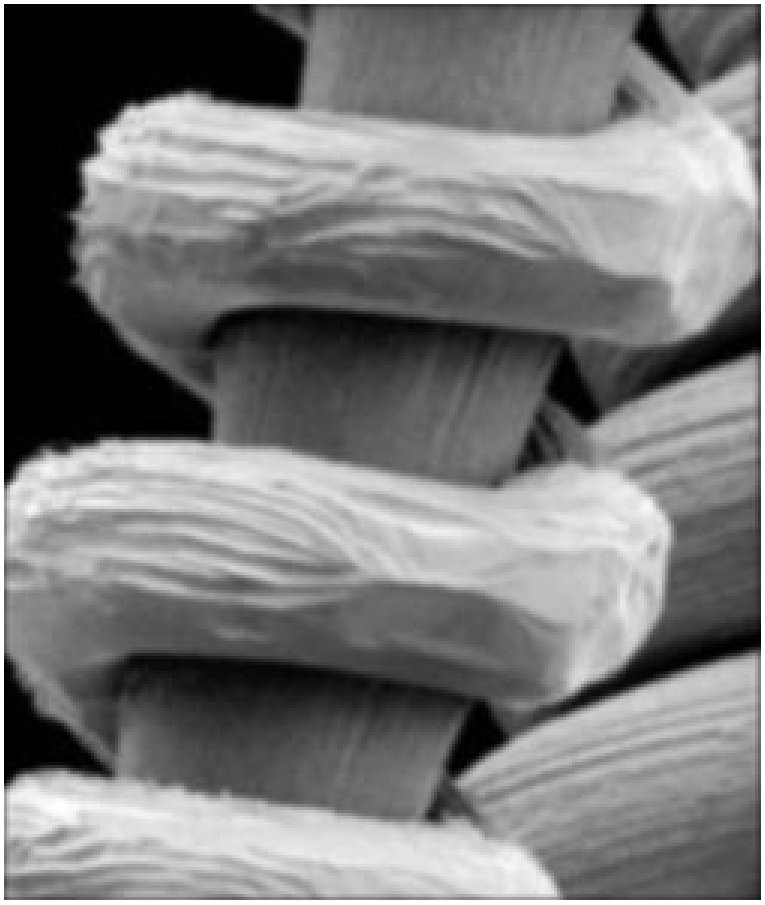


Figure 36: Imagen filtro ponderado 5x5

- Imágenes Filtro Mediana

Figure 37: Imagen con filtro mediana 7x7

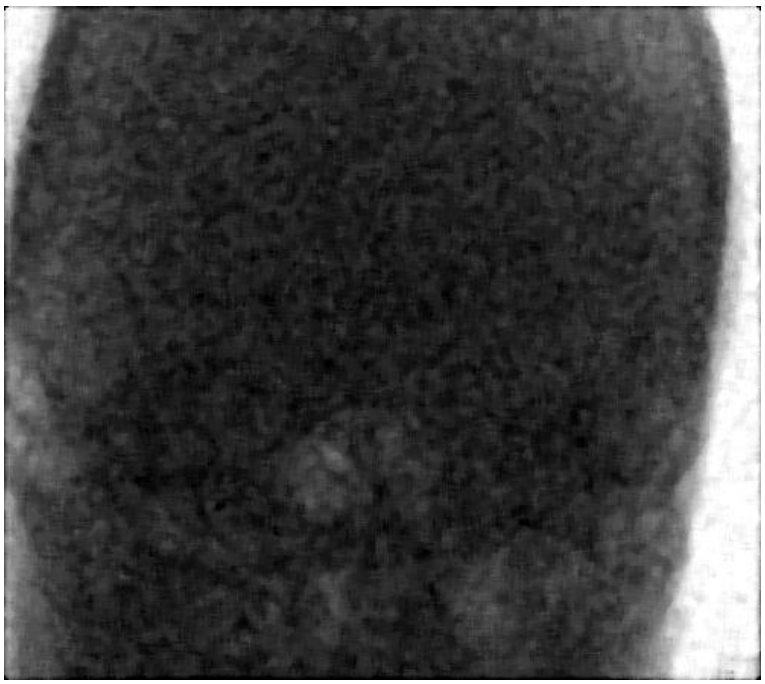


Figure 38: Imagen con filtro mediana 7x7

- Imagen filtro prewitt y sobel para extension de borde

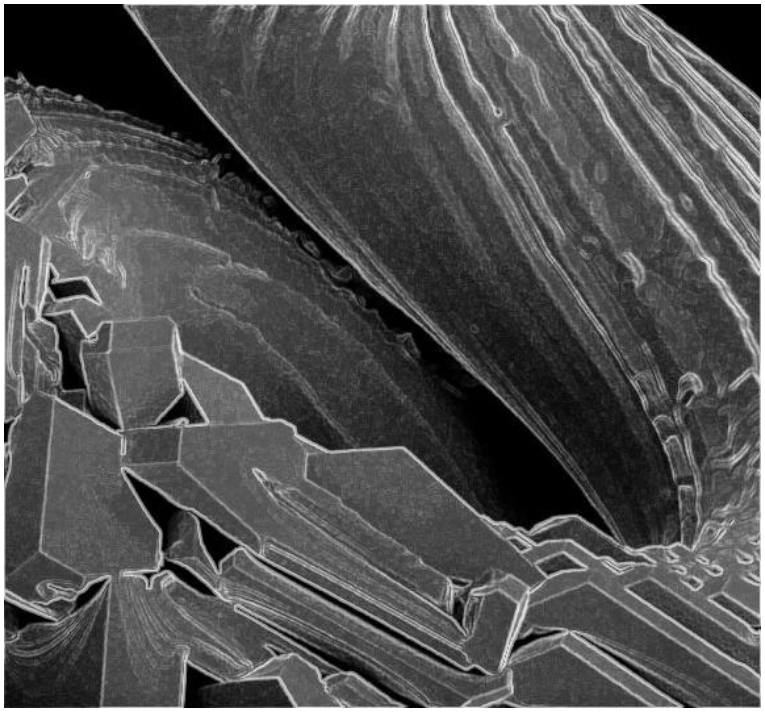


Figure 39: Imagen filtro prewitt

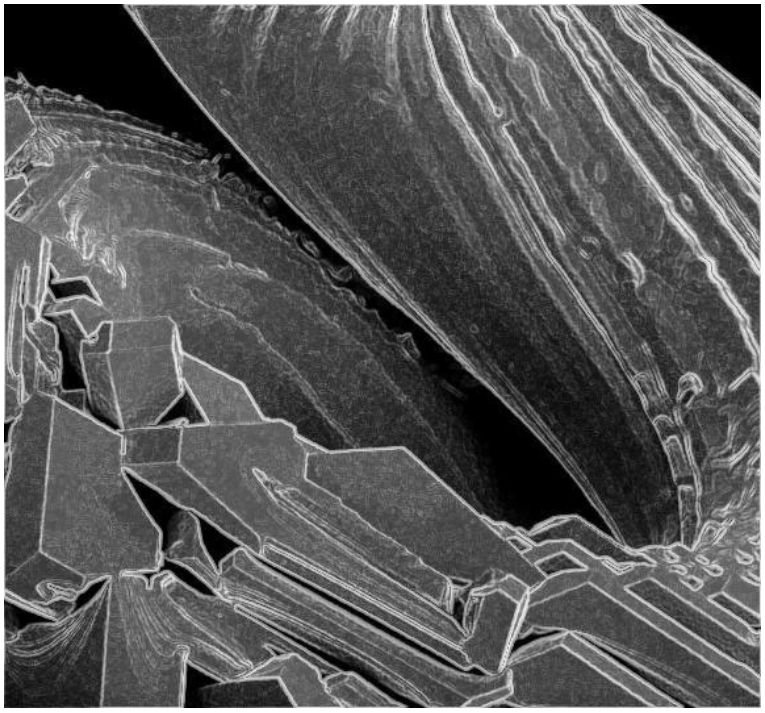


Figure 40: Imagen filtro sobel

- Imagen filtro laplaciano de 45 grados y 90 grados para extension de borde

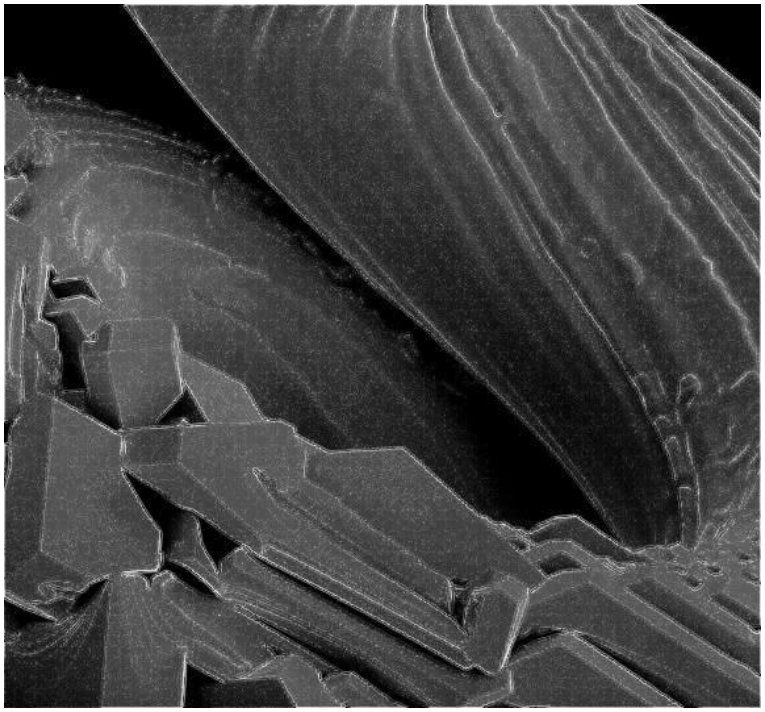


Figure 41: Imagen filtro laplaciano 45 grados

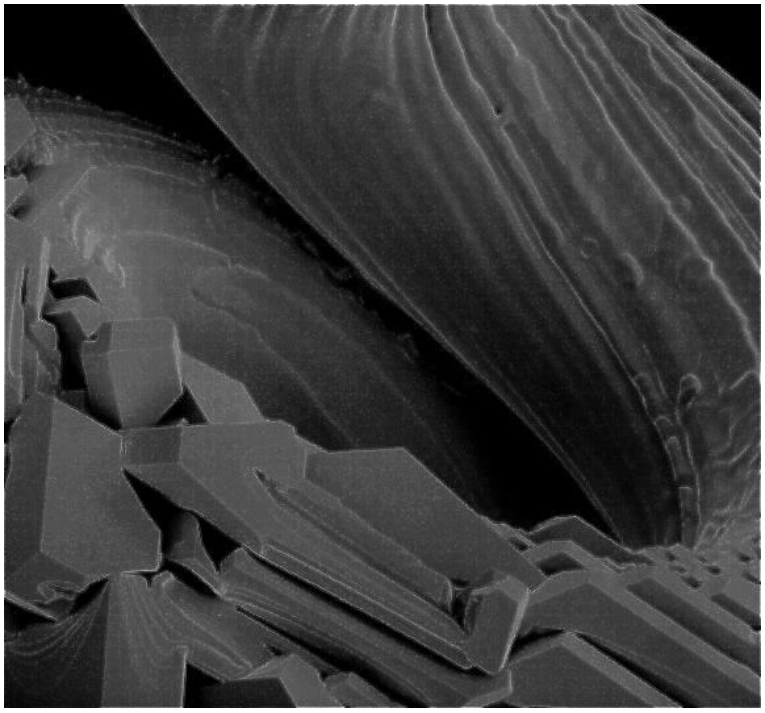


Figure 42: Imagen filtro laplaciano 90 grados

- Imágenes usando el método unsharp

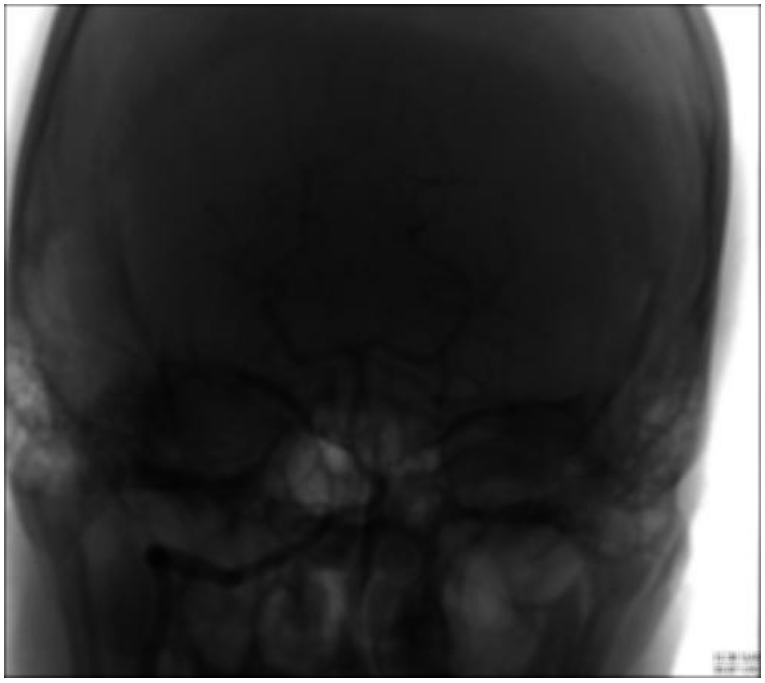


Figure 43: Imagen unsharp filtro estandar

Figure 44: Imagen unsharp filtro ponderado

5 Conclusión

Logre realizar diversas modificaciones al histograma con las transformaciones básicas: negativo, exponencial, logarítmica y gamma.imagen, conociendo la importancia del factor constante que logra distribuir de manera correcta las transformaciones de intensidades.
Observar el pro de la ecualización el histograma de una imagen y como solo nos funciona para imágenes donde el borde no se logra ver bien definido, a las imágenes que ya cumplen con lo previo, las ”empeora” en este sentido.
Logre realizar las operaciones de suavizado y de reducción de ruido en imágenes utilizando filtros espaciales y

ver que no ayudan a reducir el ruido de las imágenes, no de manera esperada.
Por ultimo logre realizar las operaciones de detección de bordes en imágenes, tanto limpias como ruidosas, utilizando filtros basados en aproximaciones de gradientes y laplacianos. Observe como el filtro sobel logra mostrar muy contrastado estos bordes, y el laplaciano de 90 detectarlos sin hacer un cambio abrupto en ellos como lo hacen los demás. Por ultimo, observe la importancia de tener las imágenes sin ruido, ya que como esta definido este problema, estos filtros no logran detectar los bordes, en cambio enruidesen mas la imagen.

6 Referencias

1. https://matplotlib.org/stable/tutorials/pyplot.htm
2. https://numpy.org/
3. https://scikit-image.org/
4. https://pillow.readthedocs.io/en/stable/reference/Image.html