EAST WEST UNIVERSITY

CSE438

Section: 01

Lab: 04 Report

Topic: min and max filtering, Gaussian,

Submitted By:

Name: Md Sifat Ullah Sheikh

ID: 2022-1-60-029

Submitted To:

K.M. Safin Kamal

Lecturer

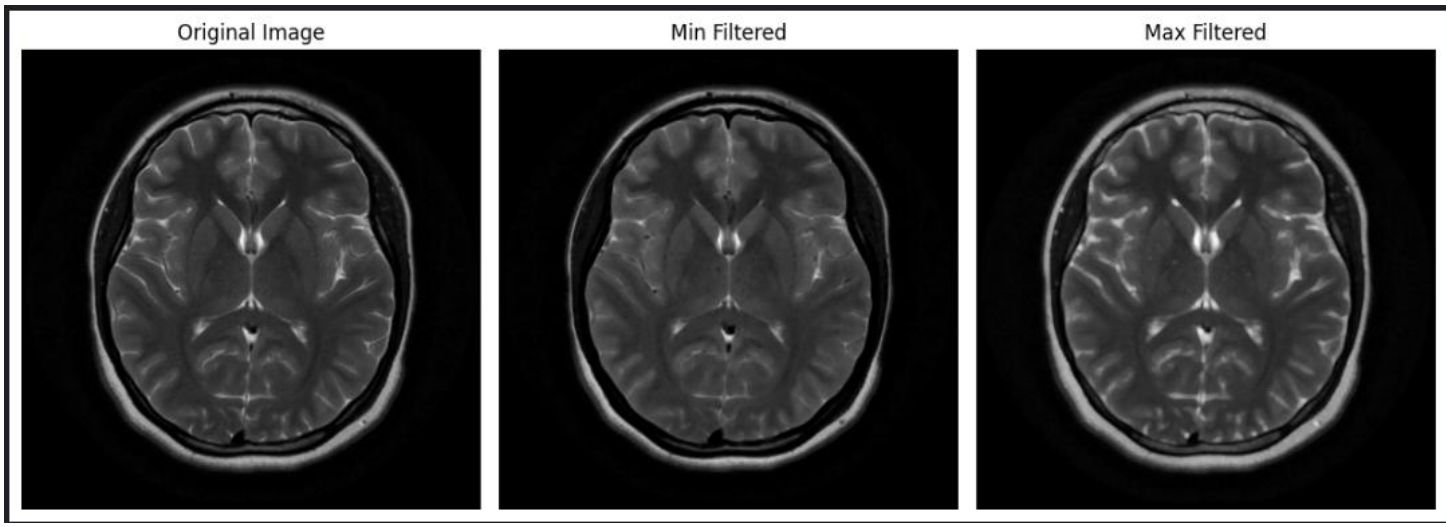
Department of Computer Science & Engineering

Date: 21 July 2025

Q1. Use min and max filtering technique. Show input and output side by side.

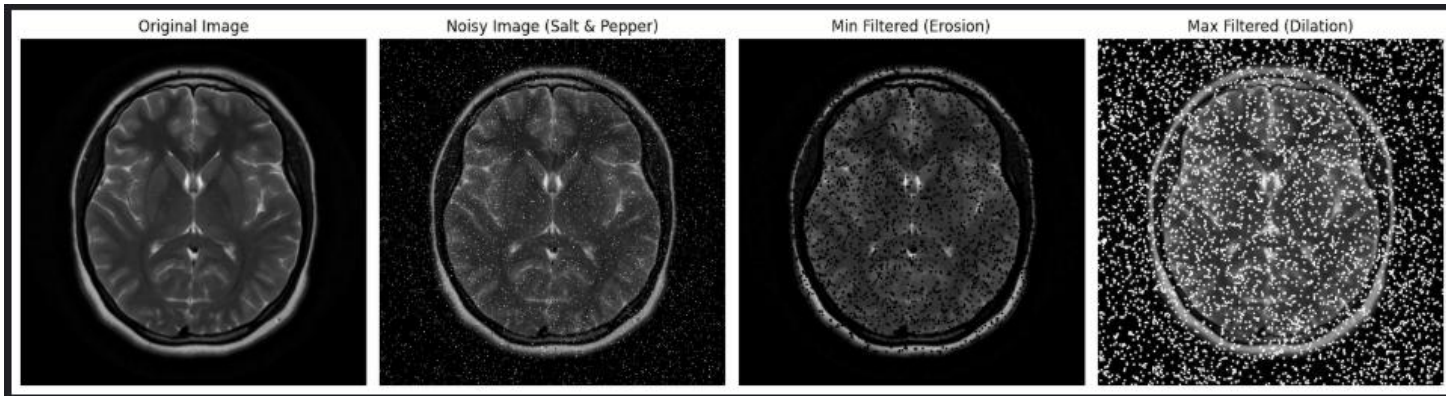
CODE:

```
 import cv2
import numpy as np
import matplotlib.pyplot as plt
img = cv2.imread('/kaggle/input/lab-04/lab_04_datasets/Picture1.png', cv2.IMREAD_GRAYSCALE)
kernel = np.ones((3, 3), np.uint8)
min_filtered = cv2.erode(img, kernel)
max_filtered = cv2.dilate(img, kernel)
plt.figure(figsize=(12, 6))

plt.subplot(1, 3, 1)
plt.title("Original Image")
plt.imshow(img, cmap='gray')
plt.axis('off')

plt.subplot(1, 3, 2)
plt.title("Min Filtered")
plt.imshow(min_filtered, cmap='gray')
plt.axis('off')

plt.subplot(1, 3, 3)
plt.title("Max Filtered")
plt.imshow(max_filtered, cmap='gray')
plt.axis('off')

plt.tight_layout()
plt.show()
```




```
 def add_salt_and_pepper_noise(image, salt_prob, pepper_prob):
   noisy_img = image.copy()
   total_pixels = image.size


   num_salt = int(total_pixels * salt_prob)
   salt_coords = [np.random.randint(0, i - 1, num_salt) for i in image.shape]
   noisy_img[salt_coords[0], salt_coords[1]] = 255


   num_pepper = int(total_pixels * pepper_prob)
   pepper_coords = [np.random.randint(0, i - 1, num_pepper) for i in image.shape]
   noisy_img[pepper_coords[0], pepper_coords[1]] = 0

   return noisy_img

salt_prob = 0.02
pepper_prob = 0.02
noisy_img = add_salt_and_pepper_noise(img, salt_prob, pepper_prob)
kernel = np.ones((3, 3), np.uint8)


min_filtered = cv2.erode(noisy_img, kernel)


max_filtered = cv2.dilate(noisy_img, kernel)
plt.figure(figsize=(16, 6))

plt.subplot(1, 4, 1)
plt.title("Original Image")
```

```
plt.imshow(img, cmap='gray')
plt.axis('off')

plt.subplot(1, 4, 2)
plt.title("Noisy Image (Salt & Pepper)")
plt.imshow(noisy_img, cmap='gray')
plt.axis('off')

plt.subplot(1, 4, 3)
plt.title("Min Filtered (Erosion)")
plt.imshow(min_filtered, cmap='gray')
plt.axis('off')

plt.subplot(1, 4, 4)
plt.title("Max Filtered (Dilation)")
plt.imshow(max_filtered, cmap='gray')
plt.axis('off')

plt.tight_layout()
plt.show()
```

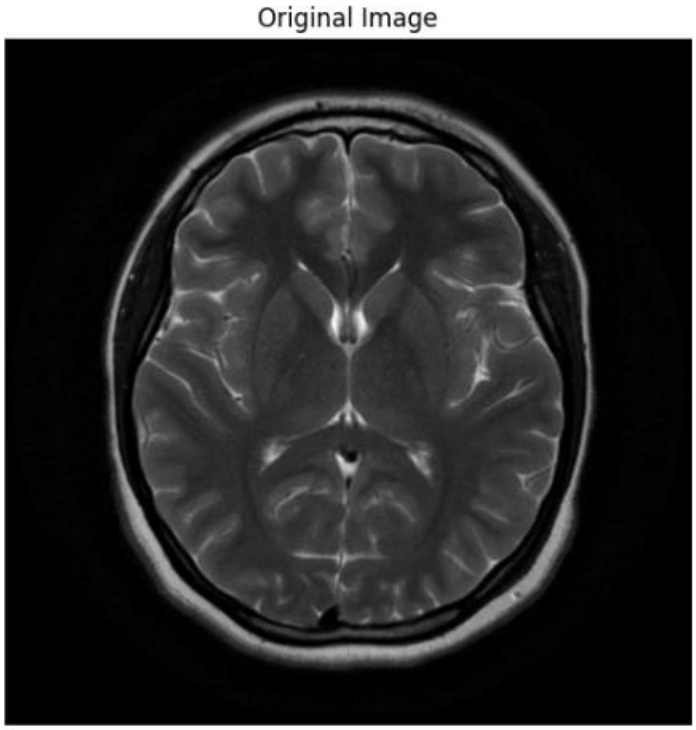


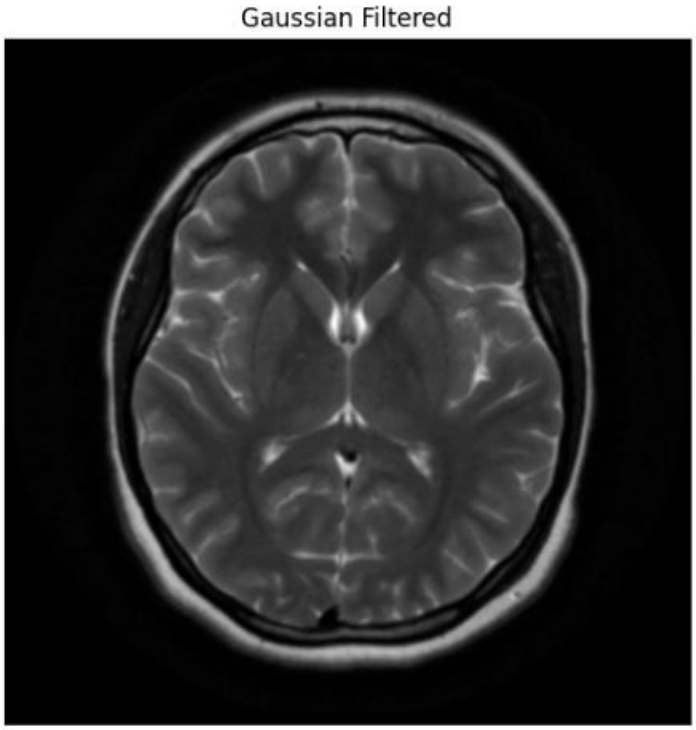
Q2. Use Gaussian filtering. Show input and output side by side.

CODE

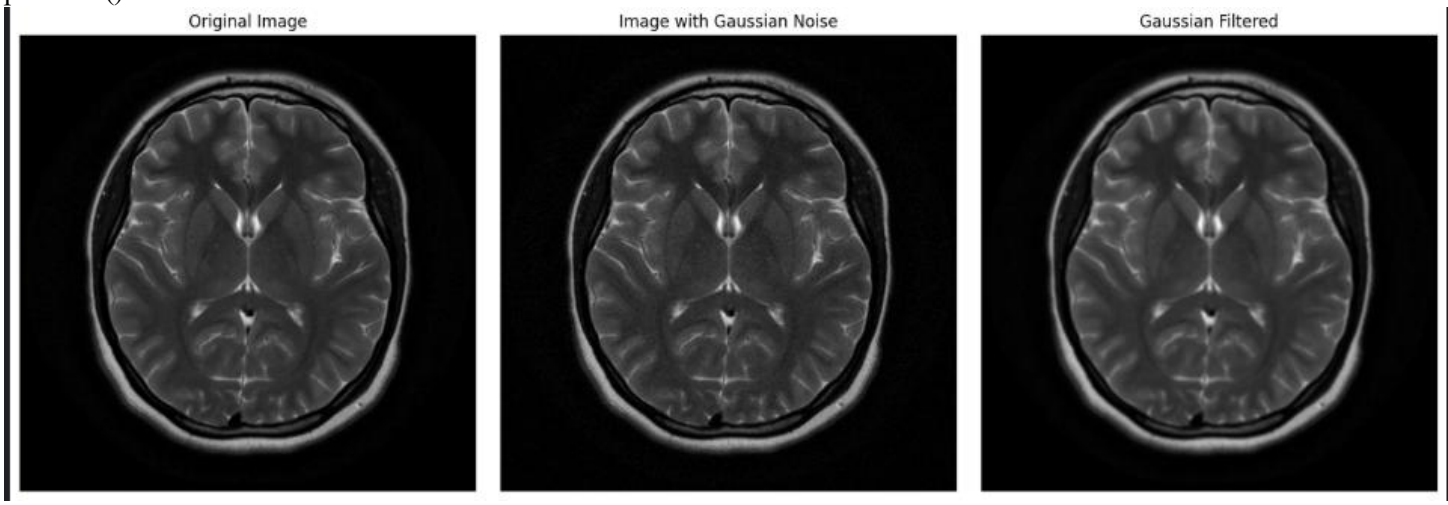
```
def add_gaussian_noise(image,mean=0,var=1):
```

```
    sigma = var ** 0.5
    gaussian = np.random.normal(mean,sigma,image.shape)
    noisy_img = image + gaussian
    noisy_img = np.clip(noisy_img,0,255).astype(np.uint8)
    return noisy_img
noisy_img = add_gaussian_noise(img,mean=0,var=25)
gaussian_filtered = cv2.GaussianBlur(img, (5, 5), 0)


plt.figure(figsize=(10, 5))

plt.subplot(1, 2, 1)
plt.title('Original Image')
plt.imshow(img, cmap='gray')
plt.axis('off')

plt.subplot(1, 2, 2)
plt.title('Gaussian Filtered')
plt.imshow(gaussian_filtered, cmap='gray')
plt.axis('off')

plt.tight_layout()
plt.show()
```





```
plt.figure(figsize=(15, 5))

plt.subplot(1, 3, 1)
plt.title('Original Image')
plt.imshow(img, cmap='gray')
plt.axis('off')

plt.subplot(1, 3, 2)
plt.title('Image with Gaussian Noise')
plt.imshow(noisy_img, cmap='gray')
plt.axis('off')

plt.subplot(1, 3, 3)
plt.title('Gaussian Filtered')
plt.imshow(gaussian_filtered, cmap='gray')
plt.axis('off')

plt.tight_layout()
plt.show()
```

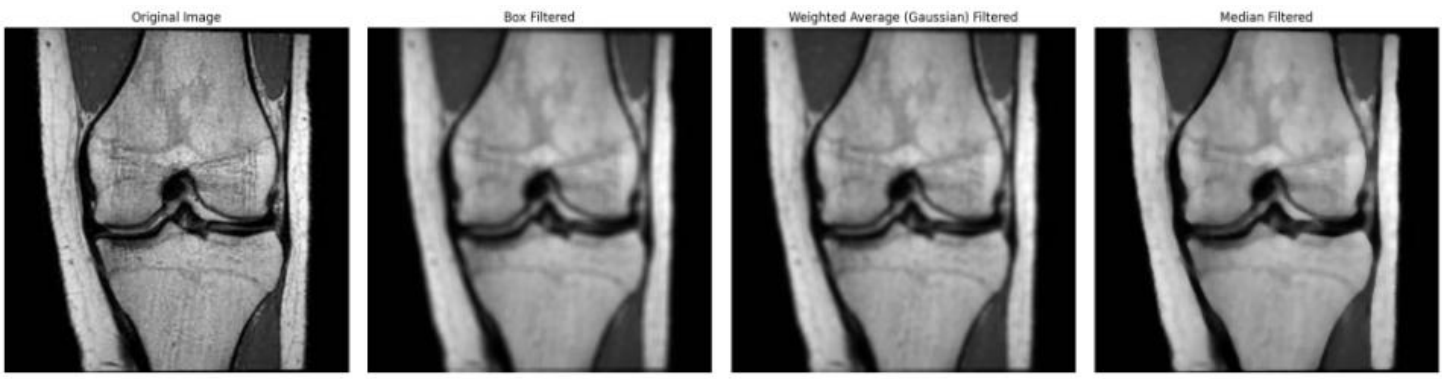


Q3. Apply the following filters:
a) Box filtering
b) Average filtering
c) Median filtering
Show input and output side by side. Also show the comparison between the 3 techniques. Mention which method works better than others.

CODE:

```
img_3 = cv2.imread('/kaggle/input/lab-04/lab_04_datasets/Picture3.png', cv2.IMREAD_GRAYSCALE)
```

```
box_filtered = cv2.boxFilter(img_3, ddepth=-1, ksize=(5, 5), normalize=True)
weighted_filtered = cv2.GaussianBlur(img_3, (5, 5), sigmaX=1.5)
median_filtered = cv2.medianBlur(img_3, ksize=5)
plt.figure(figsize=(20, 5))

plt.subplot(1, 4, 1)
plt.title('Original Image')
plt.imshow(img_3, cmap='gray')
plt.axis('off')

plt.subplot(1, 4, 2)
plt.title('Box Filtered')
plt.imshow(box_filtered, cmap='gray')
plt.axis('off')

plt.subplot(1, 4, 3)
plt.title('Weighted Average (Gaussian) Filtered')
plt.imshow(weighted_filtered, cmap='gray')
plt.axis('off')

plt.subplot(1, 4, 4)
plt.title('Median Filtered')
plt.imshow(median_filtered, cmap='gray')
plt.axis('off')

plt.tight_layout()
plt.show()
```



Q2. Using the following image, solve questions a - f.
a) Read and show the image.
b) Show the matrix form of the image.
c) Show the pixel information by hovering the cursor on the image.
d) Find the value of the pixel (10, 78).
e) Show the size of the image.
f) Show the all the information of the image.

CODE

```
img_4 = cv2.imread('/kaggle/input/lab-04/lab_04_datasets/Picture4.png', cv2.IMREAD_GRAYSCALE)

plt.imshow(img_4, cmap='gray')

plt.title('Input Image_4')

plt.axis('off')

plt.show()

print("Matrix form of the image (as numpy array):")

print(img_4)
```

```
pixel_value = img[10, 78]
print(f"Value of the pixel at (10, 78): {pixel_value}")
height, width = img.shape
print(f"Size of the image: Height = {height}, Width = {width}")
print("\nFull image information:")
print(f"Shape (Height, Width): {img.shape}")
print(f"Data type: {img.dtype}")
print(f"Number of dimensions: {img.ndim}")
print(f"Min pixel value: {img.min()}")
print(f"Max pixel value: {img.max()}")
print(f"Mean pixel value: {img.mean()}")
```

```
Matrix form of the image (as numpy array):
[[235 245 252 ...  39  43  39]
 [160 203 239 ...  28  43  39]
 [ 76 137 207 ...  21  34  34]
 ...
 [ 39  34  28 ...   0  21  28]
 [ 39  34  28 ...   0   0   0]
 [ 39  34  28 ...   0   0   0]]
```

```
Value of the pixel at (10, 78): 28
```

```
Size of the image: Height = 295, Width = 289
```

```
Full image information:
Shape (Height, Width): (295, 289)
Data type: uint8
Number of dimensions: 2
Min pixel value: 0
Max pixel value: 255
Mean pixel value: 35.1648818251129
```

Q5. Using the following images, solve questions a - i.

a) Read and show all three types of images (RGB, Grayscale, and Indexed).
b) Turn the RGB image to Grayscale image.
c) Turn the Indexed image to Grayscale image.
d) Turn the Indexed image to RGB image.
e) Convert the Grayscale image to a Binary image.
f) Show the inverted form of that Binary image.
g) Show the histogram of the Grayscale image.
h) Invert the RGB image.
i) Blur the RGB image.

CODE:

```
from PIL import Image
rgb_path = '/kaggle/input/lab-04/lab_04_datasets/Picture5.png'
grayscale_path = '/kaggle/input/lab-04/lab_04_datasets/Picture6.png'
indexed_path = '/kaggle/input/lab-04/lab_04_datasets/Picture7.png'
rgb_img_bgr = cv2.imread(rgb_path)
rgb_img = cv2.cvtColor(rgb_img_bgr, cv2.COLOR_BGR2RGB)



gray_img = cv2.imread(grayscale_path, cv2.IMREAD_GRAYSCALE)
indexed_img_array = cv2.imread(indexed_path, cv2.IMREAD_UNCHANGED)
indexed_img_array = np.array(indexed_img_pil)
plt.figure(figsize=(15, 5))

plt.subplot(1, 3, 1)
plt.title('RGB Image')
plt.imshow(rgb_img)
plt.axis('off')

plt.subplot(1, 3, 2)
plt.title('Grayscale Image')
plt.imshow(gray_img, cmap='gray')
plt.axis('off')

plt.subplot(1, 3, 3)
plt.title('Indexed Image')
plt.imshow(indexed_img_array, cmap='gray')
plt.axis('off')
```

```
plt.show()
rgb_to_gray = cv2.cvtColor(rgb_img_bgr, cv2.COLOR_BGR2GRAY)
plt.figure()
plt.title('RGB to Grayscale')
plt.imshow(rgb_to_gray, cmap='gray')
plt.axis('off')
plt.show()
indexed_to_gray_pil = indexed_img_pil.convert('L')
indexed_to_gray = np.array(indexed_to_gray_pil)
plt.figure()
plt.title('Indexed to Grayscale')
plt.imshow(indexed_to_gray, cmap='gray')
plt.axis('off')
plt.show()
indexed_to_rgb_pil = indexed_img_pil.convert('RGB')
indexed_to_rgb = np.array(indexed_to_rgb_pil)

plt.figure()
plt.title('Indexed to RGB')
plt.imshow(indexed_to_rgb)
plt.axis('off')
plt.show()
_, binary_img = cv2.threshold(gray_img, 127, 255, cv2.THRESH_BINARY)
plt.figure()
plt.title('Grayscale to Binary')
plt.imshow(binary_img, cmap='gray')
```

```
plt.axis('off')
plt.show()
binary_inverted = cv2.bitwise_not(binary_img)
plt.figure()
plt.title('Inverted Binary Image')
plt.imshow(binary_inverted, cmap='gray')
plt.axis('off')
plt.show()

plt.figure(figsize=(8,5))
plt.title('Histogram of Grayscale Image')
plt.xlabel('Pixel intensity')
plt.ylabel('Frequency')
plt.hist(gray_img.ravel(), bins=256, range=[0,256], color='blue')
plt.show()
rgb_inverted = 255 - rgb_img
plt.figure()
plt.title('Inverted RGB Image')
plt.imshow(rgb_inverted)
plt.axis('off')
plt.show()
rgb_blurred = cv2.GaussianBlur(rgb_img_bgr, (7,7), 0)
rgb_blurred = cv2.cvtColor(rgb_blurred, cv2.COLOR_BGR2RGB)
rgb_blurred = cv2.GaussianBlur(rgb_img_bgr, (7,7), 0)
rgb_blurred = cv2.cvtColor(rgb_blurred, cv2.COLOR_BGR2RGB)
plt.figure()
plt.title('Blurred RGB Image')
```

```
plt.figure(figsize=(20, 12))

plt.subplot(3, 3, 1)
plt.title('RGB to Grayscale')
plt.imshow(rgb_to_gray, cmap='gray')
plt.axis('off')

plt.subplot(3, 3, 2)
plt.title('Indexed to Grayscale')
plt.imshow(indexed_to_gray, cmap='gray')
plt.axis('off')

plt.subplot(3, 3, 3)
plt.title('Indexed to RGB')
plt.imshow(indexed_to_rgb)
plt.axis('off')

plt.subplot(3, 3, 4)
plt.title('Grayscale to Binary')
plt.imshow(binary_img, cmap='gray')
plt.axis('off')

plt.subplot(3, 3, 5)
plt.title('Inverted Binary')
plt.imshow(binary_inverted, cmap='gray')
plt.axis('off')
```

```
plt.subplot(3, 3, 6)
plt.title('Inverted RGB')
plt.imshow(rgb_inverted)
plt.axis('off')

plt.subplot(3, 3, 7)
plt.title('Blurred RGB')
plt.imshow(rgb_blurred)
plt.axis('off')

plt.tight_layout()
plt.show()
plt.imshow(rgb_blurred)
plt.axis('off')
plt.show()
```

RGB Image

Grayscale Image

Indexed Image

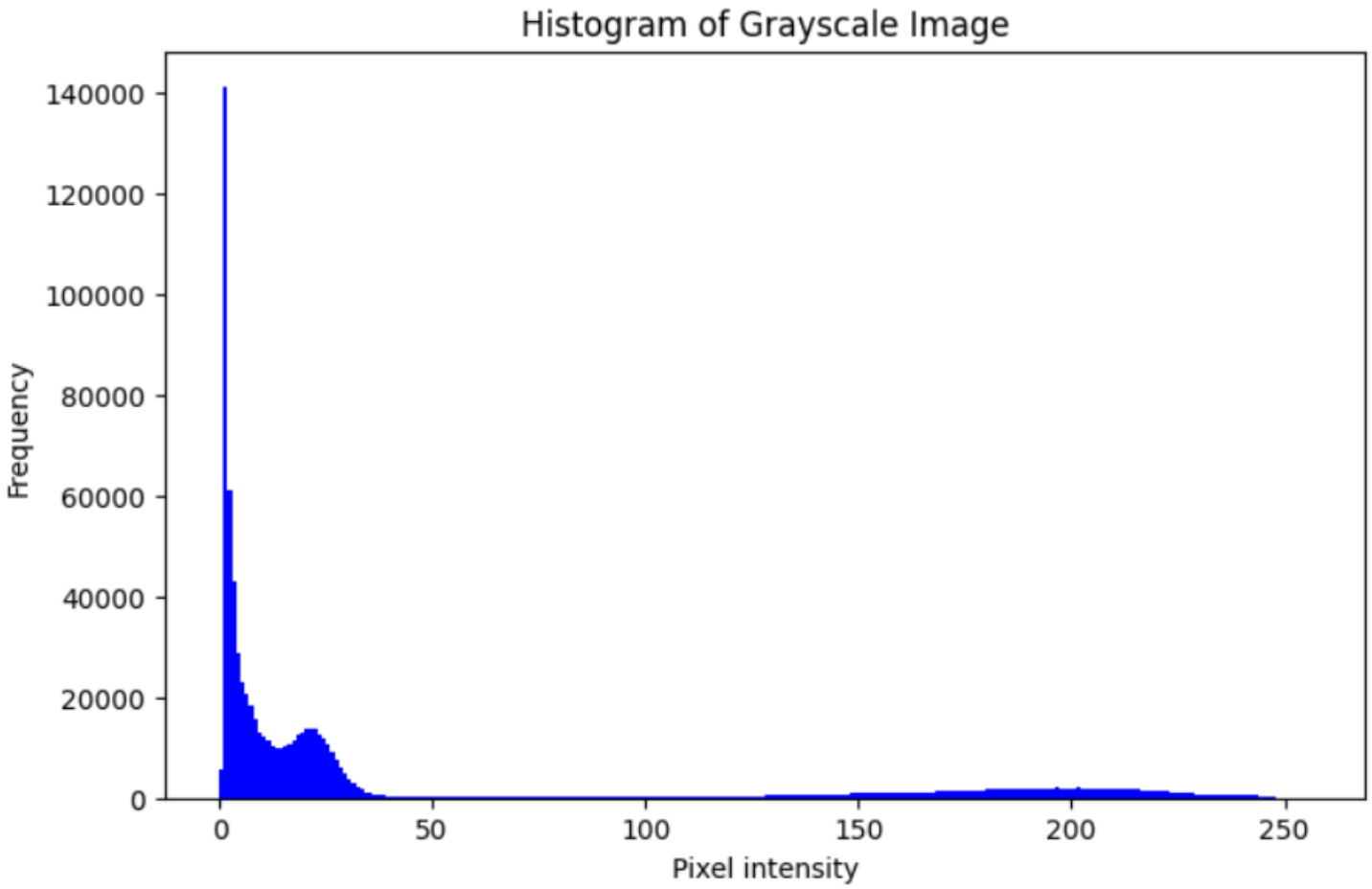
RGB to Grayscale

Indexed to Grayscale

Indexed to RGB

Grayscale to Binary

Inverted Binary Image


Histogram of Grayscale Image
Frequency
140000
120000
100000
80000
60000
40000
20000
0
0
50
100
150
200
250
Pixel intensity

Inverted RGB Image

Blurred RGB Image