CSE 3113 / CSE 3214

INTRODUCTION TO DIGITAL IMAGE PROCESSING

SPRING 2024

Homework 2 Report

Alker AYHAN 210316063

Cengizhan BAYRAM – 200316066

Submission Date: 29 April 2024

Programming Environment

Write your Pyhton/Matlab/Octave version and additional packages if needed.

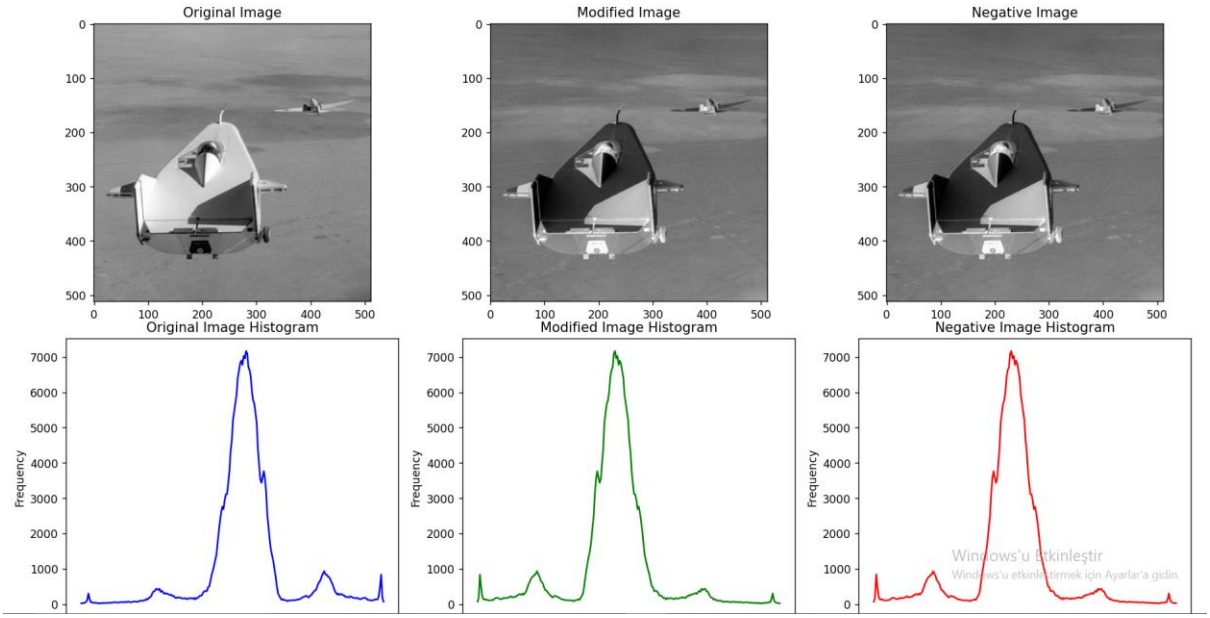
OpenCV sürümü: 4.9.0

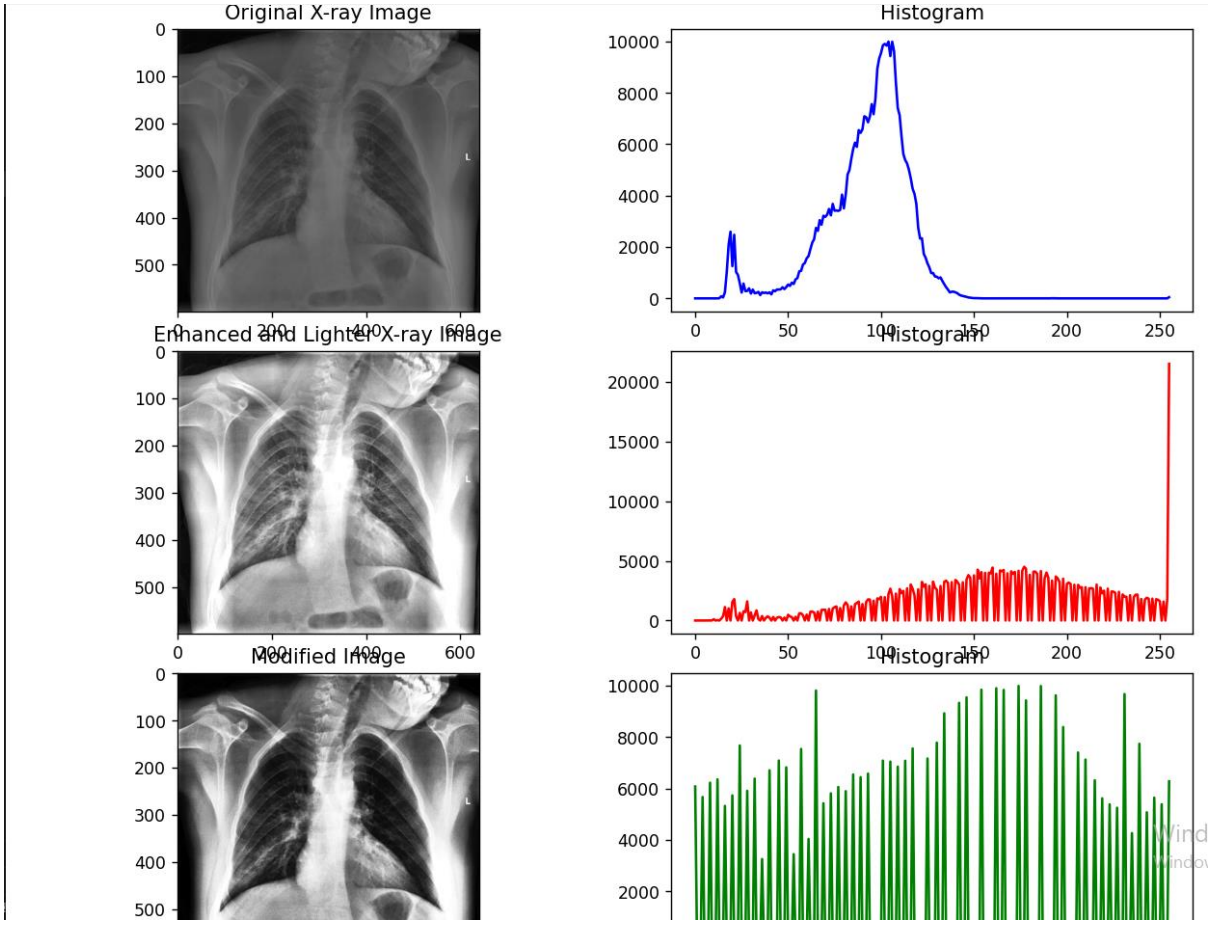
Python 3.11.5

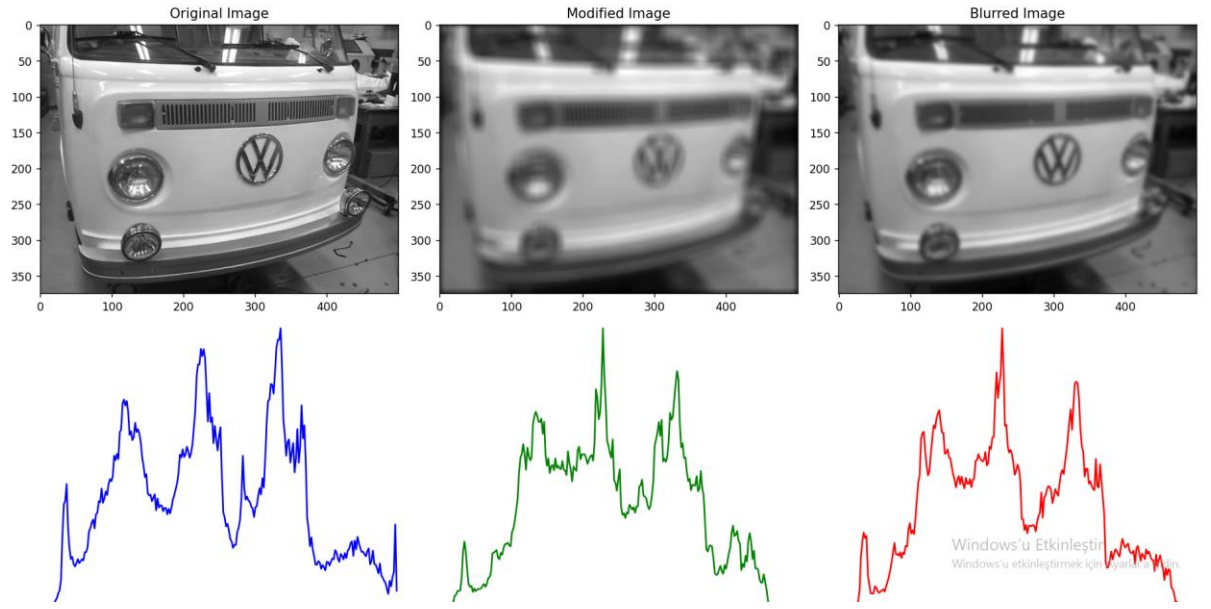
Matplotlib 3.7.2

Results

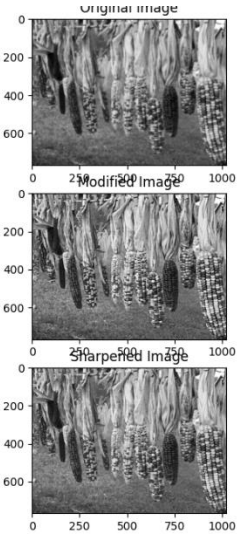
Give your code outputs here. Your code should produce <u>at least</u> the following two figures <u>for each image</u>.

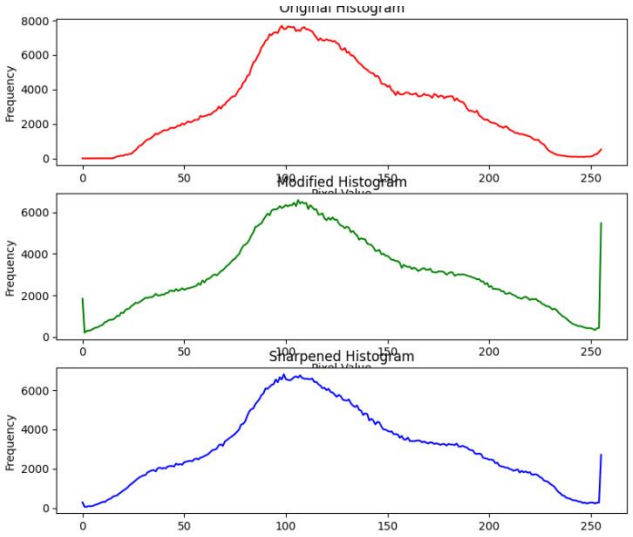


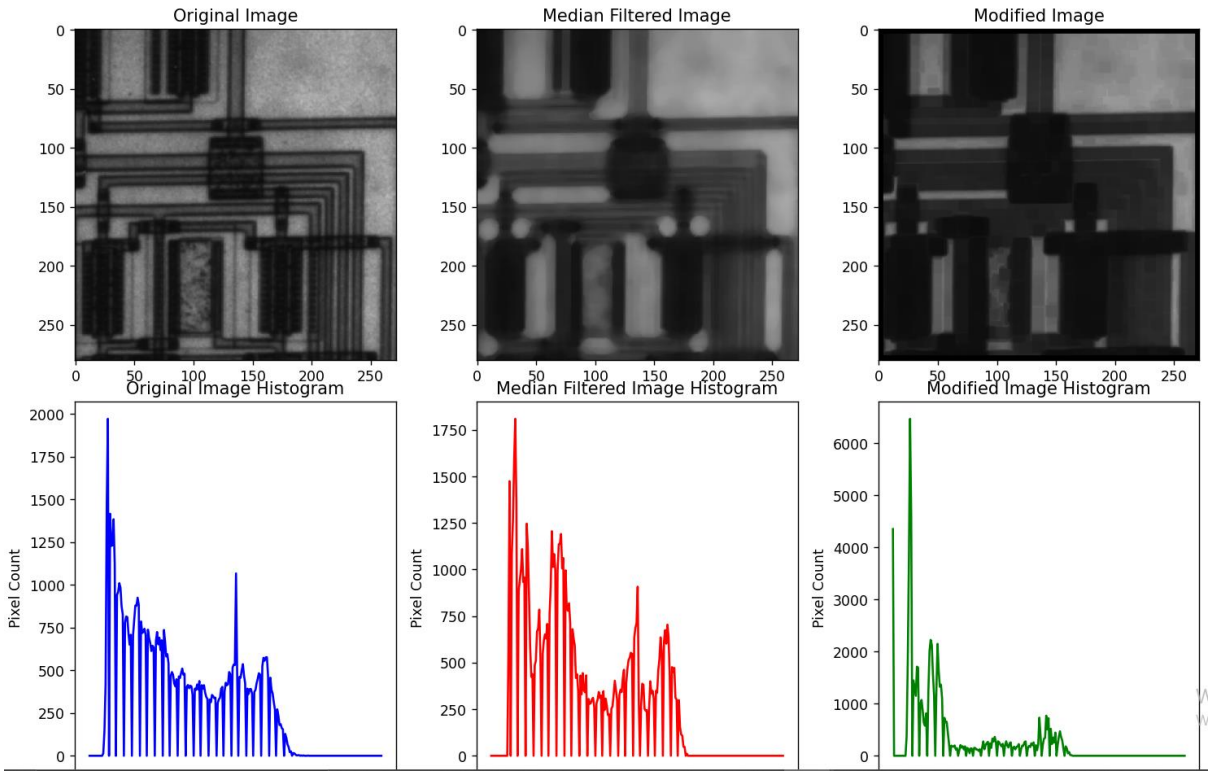


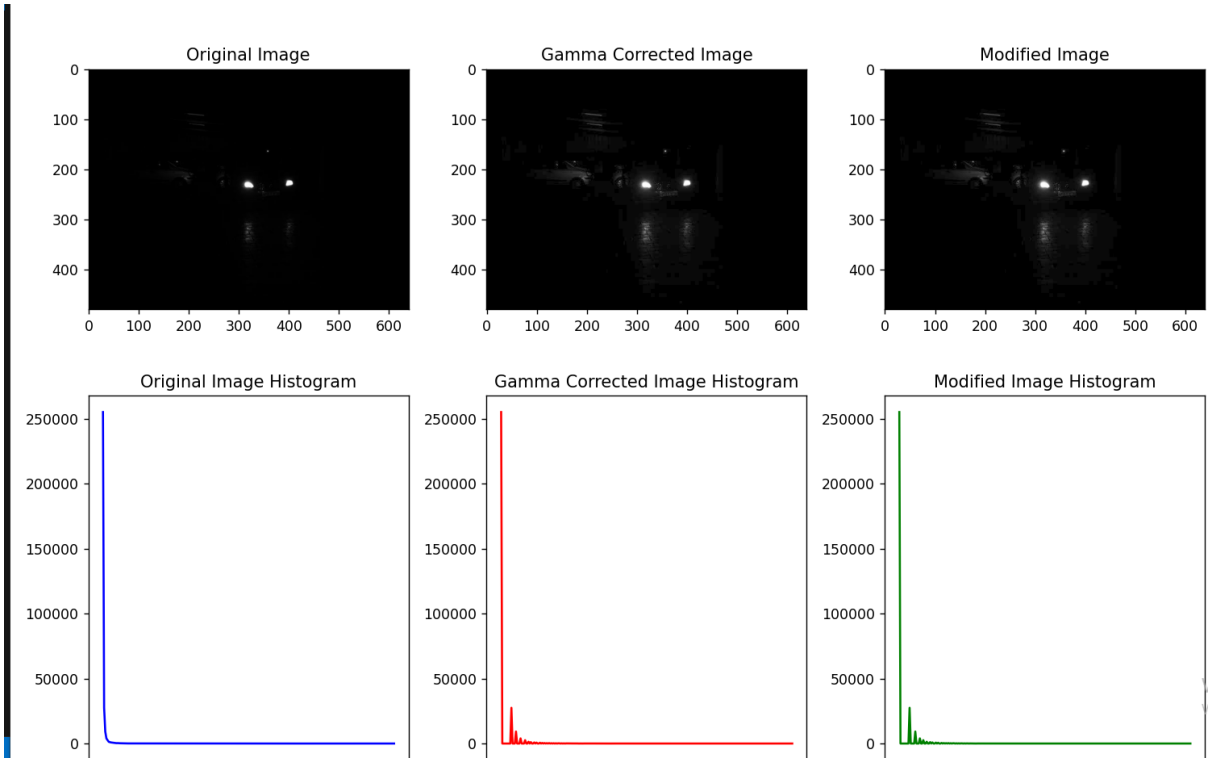


I cant solve this problem but other images are true


Original Image
Modified Image
Sharpened Image


Original Histogram
Modified Histogram
Sharpened Histogram
Frequency
Pixel Value





Discussion

This is the main part of your assignment. Justify your predictions and selected parameters by answering the following questions for each image (A – F).

- *What spatial domain technique was used to modify the source image and why?*
- *What are the parameters (if any) of the applied method?*
- *Have you obtained exactly the same output or an approximation? (You can compare the images by calculating the sum of the pointwise differences.)*

Reflections

Write a few sentences about what you have learnt, the difficulties you have faced, etc.

Source Code

Copy and paste your source codes here. (Note that you must also submit your source codes, in executable code files (.py, .ipynb, .m, or .mlx). Otherwise, your code will not be graded.)

A- When we examined the photos in 'A', it was clearly evident that reverse colors were used. Therefore, we decided to create the modified photo using the reverse colors method.
We created the modified photo by using code that takes the inverse of the pixels in the image.

Source Code(Python) For A:

```
import cv2
import matplotlib.pyplot as plt

# Load the original and modified images
original_image = cv2.imread(r'C:\Users\cengh\Desktop\repos\idip\A_original.png', cv2.IMREAD_GRAYSCALE)
modified_image = cv2.imread(r'C:\Users\cengh\Desktop\repos\idip\A_modified.png', cv2.IMREAD_GRAYSCALE)

# Check if the images are loaded successfully
if original_image is None or modified_image is None:
    print("Error: Could not load the image")
    exit()
```

```
# Calculate the negative image
negative_image = 255 - original_image

# Calculate histograms
original_hist = cv2.calcHist([original_image], [0], None, [256], [0, 256])
modified_hist = cv2.calcHist([modified_image], [0], None, [256], [0, 256])
negative_hist = cv2.calcHist([negative_image], [0], None, [256], [0, 256])

# Plot histograms
plt.figure(figsize=(15, 10))

# Display original image
plt.subplot(2, 3, 1)
plt.imshow(original_image, cmap='gray')
plt.title('Original Image')

# Display modified image
plt.subplot(2, 3, 2)
plt.imshow(modified_image, cmap='gray')
plt.title('Modified Image')

plt.subplot(2, 3, 4)
plt.plot(original_hist, color='blue')
plt.title('Original Image Histogram')
plt.xlabel('Pixel Value')
plt.ylabel('Frequency')

plt.subplot(2, 3, 5)
plt.plot(modified_hist, color='green')
plt.title('Modified Image Histogram')
plt.xlabel('Pixel Value')
plt.ylabel('Frequency')

# Display negative image
plt.subplot(2, 3, 3)
plt.imshow(negative_image, cmap='gray')
plt.title('Negative Image')

# Negative image histogram
plt.subplot(2, 3, 6)
plt.plot(negative_hist, color='red')
```

```
plt.title('Negative Image Histogram')
plt.xlabel('Pixel Value')
plt.ylabel('Frequency')

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

B- **Upon observing that the procedure applied to the images in B increased the details in the MRI images, we inferred that this could be achieved using histogram stretching and thus implemented it.**

Source Code B (Python):

```
import cv2
import numpy as np
import matplotlib.pyplot as plt

def enhance_xray(image, brightness_factor):
    """
    Enhances the given X-ray image using CLAHE and adjusts brightness.
    """
    # Convert image to grayscale if necessary
    if len(image.shape) > 2:
        gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    else:
        gray = image

    # Apply CLAHE (Contrast Limited Adaptive Histogram Equalization)
    clahe = cv2.createCLAHE(clipLimit=2.0, tileGridSize=(8, 8))
    enhanced = clahe.apply(gray)

    # Adjust brightness by scaling pixel values
    enhanced_lighter = cv2.addWeighted(enhanced, brightness_factor,
np.zeros_like(enhanced), 0, 0)

    return enhanced_lighter

if __name__ == "__main__":
    # Load the X-ray image and the modified image
    xray_image_path = r'C:\Users\cengh\Desktop\repos\idip\B_original.png'
    modified_image_path = r'C:\Users\cengh\Desktop\repos\idip\B_modified.png'

    xray_image = cv2.imread(xray_image_path, cv2.IMREAD_GRAYSCALE)
    modified_image = cv2.imread(modified_image_path, cv2.IMREAD_GRAYSCALE)
```

```
# Enhance the X-ray image and make it lighter
brightness_factor = 1.5  # Increase brightness by a factor of 1.5 (adjust as needed)
enhanced_xray_lighter = enhance_xray(xray_image, brightness_factor)

# Calculate histograms
original_hist = cv2.calcHist([xray_image], [0], None, [256], [0, 256])
enhanced_hist = cv2.calcHist([enhanced_xray_lighter], [0], None, [256], [0, 256])
modified_hist = cv2.calcHist([modified_image], [0], None, [256], [0, 256])

# Plot histograms
fig, axs = plt.subplots(3, 2, figsize=(10, 15))

# Display original X-ray image and its histogram
axs[0, 0].imshow(xray_image, cmap='gray')
axs[0, 0].set_title('Original X-ray Image')
axs[0, 1].plot(original_hist, color='blue')
axs[0, 1].set_title('Histogram')

# Display enhanced X-ray image and its histogram
axs[1, 0].imshow(enhanced_xray_lighter, cmap='gray')
axs[1, 0].set_title('Enhanced and Lighter X-ray Image')
axs[1, 1].plot(enhanced_hist, color='red')
axs[1, 1].set_title('Histogram')

# Display modified image and its histogram
axs[2, 0].imshow(modified_image, cmap='gray')
axs[2, 0].set_title('Modified Image')
axs[2, 1].plot(modified_hist, color='green')
axs[2, 1].set_title('Histogram')

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

C- When we examined the photos , we noticed that blurring had been applied and we decided to use Gaussian method.

Source Code C (Python):

```
import cv2

import matplotlib.pyplot as plt


def plot_histogram(ax, image, color):
```

```
    hist = cv2.calcHist([image], [0], None, [256], [0, 256])
    ax.plot(hist, color=color)
    ax.set_xlim([0, 256])
    ax.set_xlabel('Pixel Value')
    ax.set_ylabel('Frequency')

if __name__ == "__main__":
    # Load the original, blurred, and modified images
    original_path = r'C:\Users\cengh\Desktop\repos\idip\C_original.png'
    modified_path = r'C:\Users\cengh\Desktop\repos\idip\C_modified.png'

    original_image = cv2.imread(original_path, cv2.IMREAD_GRAYSCALE)
    blurred_image = cv2.GaussianBlur(original_image, (15, 15), 0)
    modified_image = cv2.imread(modified_path, cv2.IMREAD_GRAYSCALE)

    # Plot images and histograms
    fig, axs = plt.subplots(2, 3, figsize=(15, 10))

    # Display original image
    axs[0, 0].imshow(original_image, cmap='gray')
    axs[0, 0].set_title('Original Image')
    axs[1, 0].axis('off')
    plot_histogram(axs[1, 0], original_image, color='blue')

    # Display modified image
    axs[0, 1].imshow(modified_image, cmap='gray')
    axs[0, 1].set_title('Modified Image')
```

```
axs[1, 1].axis('off')
plot_histogram(axs[1, 1], modified_image, color='green')

# Display blurred image
axs[0, 2].imshow(blurred_image, cmap='gray')
axs[0, 2].set_title('Blurred Image')
axs[1, 2].axis('off')
plot_histogram(axs[1, 2], blurred_image, color='red')

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

D- When we examined the photographs, we noticed that sharpening had been applied. Consequently, we concluded that we also needed to apply a sharpening filter and did so.

Source Code D (Python):

```
import cv2
import numpy as np
import matplotlib.pyplot as plt

def plot_histogram(ax, image, title, color='blue'):
    hist = cv2.calcHist([image], [0], None, [256], [0, 256])
    ax.plot(hist, color=color)
    ax.set_title(title + ' Histogram')
    ax.set_xlabel('Pixel Value')
    ax.set_ylabel('Frequency')

# Load the original and modified images
```

```
original_image = cv2.imread(r'C:\Users\cengh\Desktop\repos\idip\D_original.png', cv2.IMREAD_GRAYSCALE)

modified_image = cv2.imread(r'C:\Users\cengh\Desktop\repos\idip\D_modified.png', cv2.IMREAD_GRAYSCALE)


# Check if the images are loaded successfully

if original_image is None or modified_image is None:

    print("Error: Could not load the image")

    exit()


# Sharpen the original image

kernel = np.array([[-1, -1, -1],

                   [-1,  9, -1],

                   [-1, -1, -1]])

sharpened_image = cv2.filter2D(original_image, -1, kernel)


# Create subplots

fig, axs = plt.subplots(3, 2, figsize=(15, 15))


# Display original image and its histogram

axs[0, 0].imshow(original_image, cmap='gray')

axs[0, 0].set_title('Original Image')

plot_histogram(axs[0, 1], original_image, 'Original', color='red')


# Display modified image and its histogram

axs[1, 0].imshow(modified_image, cmap='gray')

axs[1, 0].set_title('Modified Image')

plot_histogram(axs[1, 1], modified_image, 'Modified', color='green')
```

```
# Display sharpened image and its histogram
axs[2, 0].imshow(sharpened_image, cmap='gray')
axs[2, 0].set_title('Sharpened Image')
plot_histogram(axs[2, 1], sharpened_image, 'Sharpened', color='blue')

# Adjust spacing and show plot
plt.tight_layout()
plt.show()
```

E- When we examined the photos in 'E', we observed that the noise had been removed. For this, we used the median method, setting the scale to 9, which allowed us to obtain a close image

Source Code E (Python):

```
import cv2
import numpy as np
from matplotlib import pyplot as plt

# Dosya yolu
original_file_path = r"C:\Users\cengh\Desktop\repos\idip\E_original.png"
modified_file_path = r"C:\Users\cengh\Desktop\repos\idip\E_modified.png"

# Fotoğrafları oku
original_image = cv2.imread(original_file_path)
modified_image = cv2.imread(modified_file_path)

# Median filtresi uygula
median_filtered_image = cv2.medianBlur(original_image, 11)  # 11x11 kernel boyutu
```

```
# Orjinal görüntü histogramını hesapla
hist_original = cv2.calcHist([original_image], [0], None, [256], [0, 256])

# Median filtresi uygulanmış görüntü histogramını hesapla
hist_median_filtered = cv2.calcHist([median_filtered_image], [0], None, [256], [0, 256])

# Modifiye edilmiş görüntü histogramını hesapla
hist_modified = cv2.calcHist([modified_image], [0], None, [256], [0, 256])

# Orijinal ve median filtresi uygulanmış görüntüleri göster
plt.figure(figsize=(12, 10))

plt.subplot(2, 3, 1)
plt.imshow(cv2.cvtColor(original_image, cv2.COLOR_BGR2RGB))
plt.title('Original Image')

plt.subplot(2, 3, 2)
plt.imshow(cv2.cvtColor(median_filtered_image, cv2.COLOR_BGR2RGB))
plt.title('Median Filtered Image')

plt.subplot(2, 3, 3)
plt.imshow(cv2.cvtColor(modified_image, cv2.COLOR_BGR2RGB))
plt.title('Modified Image')

# Orijinal görüntü histogramını göster
plt.subplot(2, 3, 4)
```

```
plt.plot(hist_original, color='blue')

plt.xlabel('Intensity')

plt.ylabel('Pixel Count')

plt.title('Original Image Histogram')


# Median filtresi uygulanmış görüntü histogramını göster

plt.subplot(2, 3, 5)

plt.plot(hist_median_filtered, color='red')

plt.xlabel('Intensity')

plt.ylabel('Pixel Count')

plt.title('Median Filtered Image Histogram')


# Modifiye edilmiş görüntü histogramını göster

plt.subplot(2, 3, 6)

plt.plot(hist_modified, color='green')

plt.xlabel('Intensity')

plt.ylabel('Pixel Count')

plt.title('Modified Image Histogram')


plt.tight_layout()

plt.show()
```

F- When looking at 'F', it was evident that the details in the photograph had become more visible and enhanced. We thought that we could use Power Law for this purpose, and indeed, we utilized it.

Source Code F (Python):

```
import cv2
```

```
import numpy as np
from matplotlib import pyplot as plt

# File paths
original_file_path = r"C:\Users\cengh\Desktop\repos\idip\F_original.png"
modified_file_path = r"C:\Users\cengh\Desktop\repos\idip\F_modified.png"

# Read the images
original_image = cv2.imread(original_file_path)
modified_image = cv2.imread(modified_file_path)

# Convert the images to grayscale
original_gray = cv2.cvtColor(original_image, cv2.COLOR_BGR2GRAY)
modified_gray = cv2.cvtColor(modified_image, cv2.COLOR_BGR2GRAY)

# Normalize the original image
original_normalized = original_gray / 255.0

# Apply gamma correction to the original image
gamma = 0.6  # You can adjust the gamma value
gamma_corrected = np.power(original_normalized, gamma)

# Convert the gamma-corrected image back to the range 0-255
gamma_corrected = np.uint8(gamma_corrected * 255)

# Calculate histograms of the original, gamma-corrected, and modified images
hist_original = cv2.calcHist([original_gray], [0], None, [256], [0, 256])
```

```
hist_gamma_corrected = cv2.calcHist([gamma_corrected], [0], None, [256], [0, 256])
hist_modified = cv2.calcHist([modified_gray], [0], None, [256], [0, 256])

# Show the result
plt.figure(figsize=(12, 8))

# Display original image and its histogram
plt.subplot(2, 3, 1)
plt.imshow(original_gray, cmap='gray')
plt.title('Original Image')

plt.subplot(2, 3, 4)
plt.plot(hist_original, color='blue')
plt.title('Original Image Histogram')

# Display gamma-corrected image and its histogram
plt.subplot(2, 3, 2)
plt.imshow(gamma_corrected, cmap='gray')
plt.title('Gamma Corrected Image')

plt.subplot(2, 3, 5)
plt.plot(hist_gamma_corrected, color='red')
plt.title('Gamma Corrected Image Histogram')

# Display modified image and its histogram
plt.subplot(2, 3, 3)
plt.imshow(modified_gray, cmap='gray')
```

```
plt.title('Modified Image')


plt.subplot(2, 3, 6)
plt.plot(hist_modified, color='green')
plt.title('Modified Image Histogram')


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