산업 컴퓨터비전 실제

Homework#1

산업인공지능학과
2024254008 신희권

1. Histogram Equalization

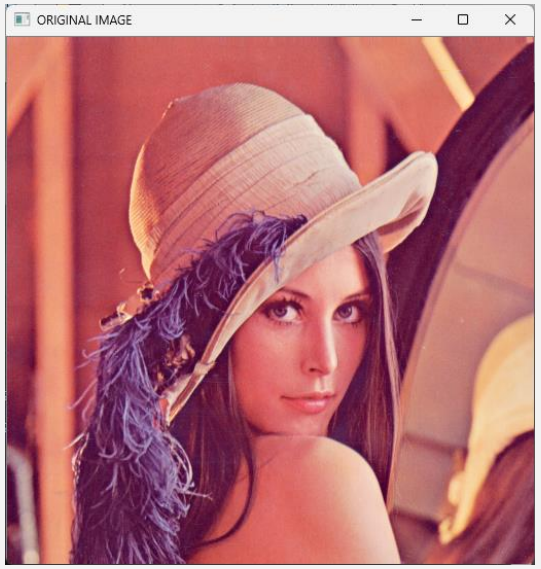
Source Code

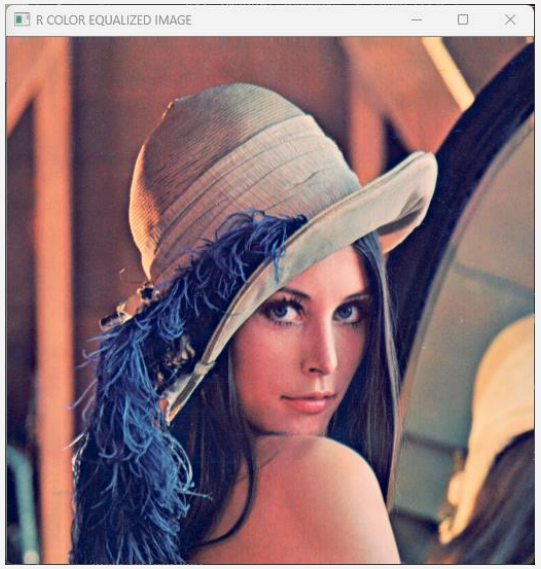
```python
import cv2
import numpy as np
from matplotlib import pyplot as plt

original_image = cv2.imread('./data/Lena.png')
cv2.imshow( winname: 'ORIGINAL IMAGE', original_image)

# 1. 사용자로부터 R, G, B 입력 받기
key = cv2.waitKey(0) & 0xFF


1 usage  HEEKWON
def equalize_histogram(index, data):
    # 3. 히스토그램에 대해서 평탄화 작업 진행
    equalized_channel = cv2.equalizeHist(data)
    equalized_image = original_image.copy()
    equalized_image[:, :, index] = equalized_channel

    draw_histogram( title: f'{chr(key).upper()} COLOR EQUALIZED HISTOGRAM', equalized_channel)

    # 4. 평탄화 이후 영상 출력
    cv2.imshow( winname: f'{chr(key).upper()} COLOR EQUALIZED IMAGE', equalized_image)

    # 5. 출력한 영상에 대해 HSV 컬러 스페이스로 변경
    hsv = cv2.cvtColor(equalized_image, cv2.COLOR_BGR2HSV)

    # 6. V 채널에 대한 평탄화 진행
    hsv[..., 2] = cv2.equalizeHist(hsv[..., 2])
    final_image = cv2.cvtColor(hsv, cv2.COLOR_HSV2BGR)

    draw_histogram( title: 'V EQUALIZED HISTOGRAM', hsv[..., 2])

    # 7. 평탄화 이후 영상 출력
    cv2.imshow( winname: 'V EQUALIZED IMAGE', final_image)
```

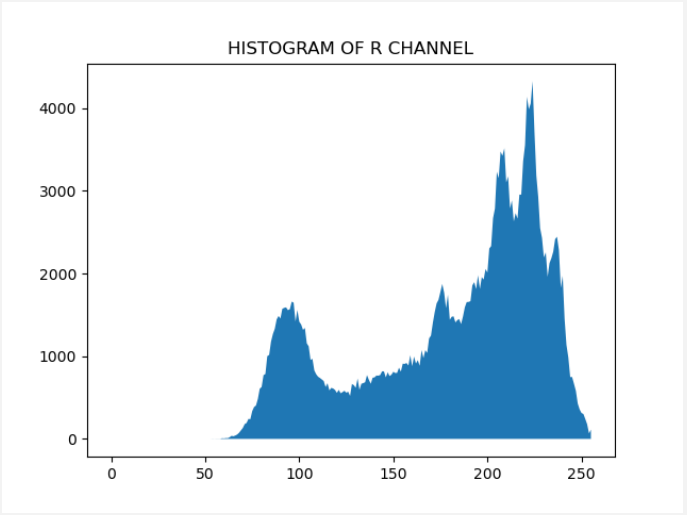
```python
3 usages  HEEKWON
def draw_histogram(title, source_data):
    hist, bins = np.histogram(source_data, bins: 256, range: [0, 256])
    plt.fill_between(range(256), hist, y2: 0)
    plt.title(title)
    plt.show()


if key in [ord('r'), ord('g'), ord('b')]:
    # OpenCV는 BGR 순서로 채널을 관리함
    channel_index = {'b': 0, 'g': 1, 'r': 2}[chr(key)]
    channel_data = original_image[:, :, channel_index]

    # 2. 입력 받은 채널에 대한 히스토그램 그리기
    draw_histogram( title: f'HISTOGRAM OF {chr(key).upper()} CHANNEL', channel_data)

    # 평탄화 함수 호출
    equalize_histogram(channel_index, channel_data)

cv2.waitKey()
cv2.destroyAllWindows()
```

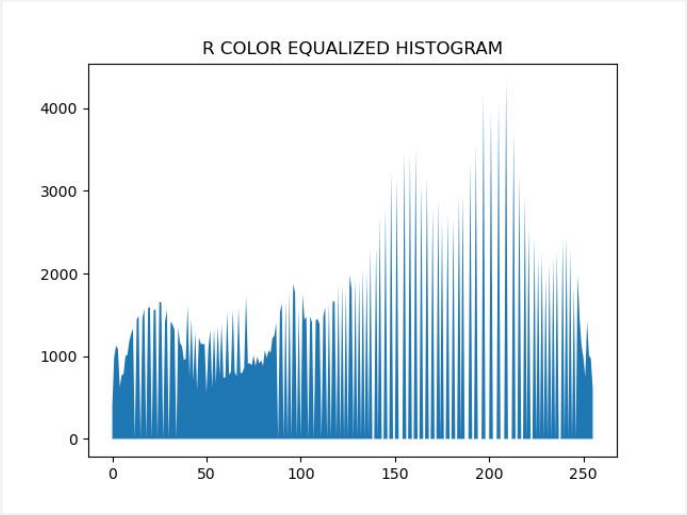
1. Histogram Equalization

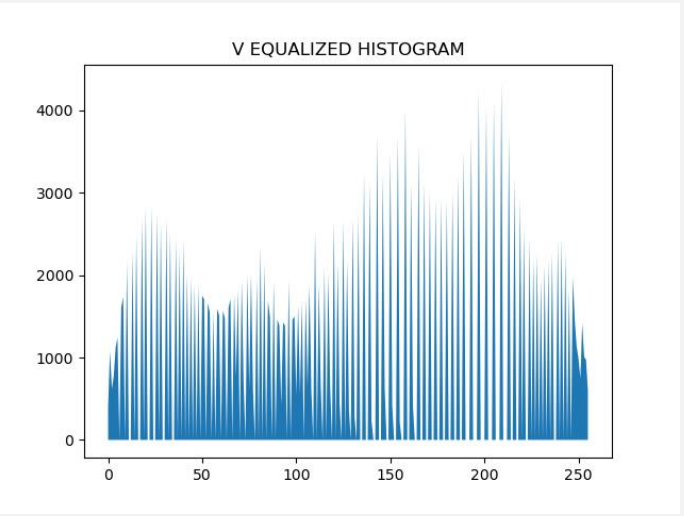
Result(‘R’ 입력)





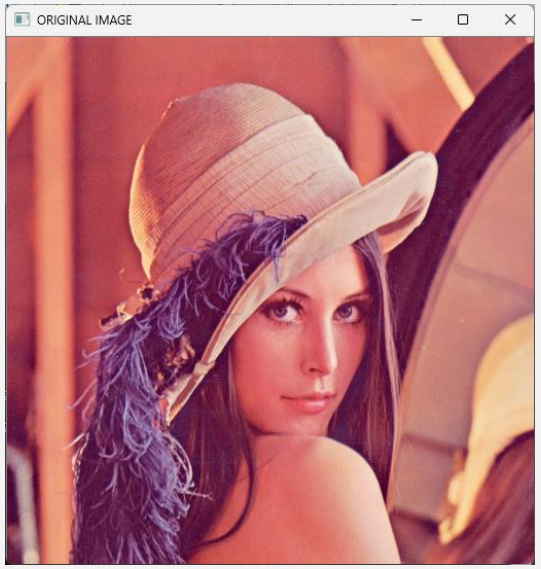


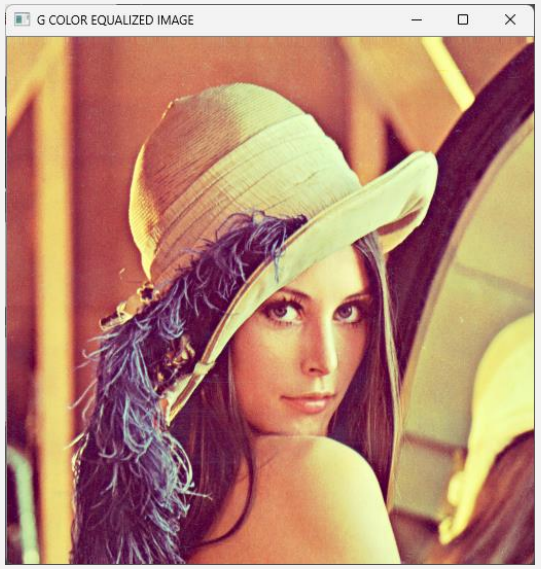


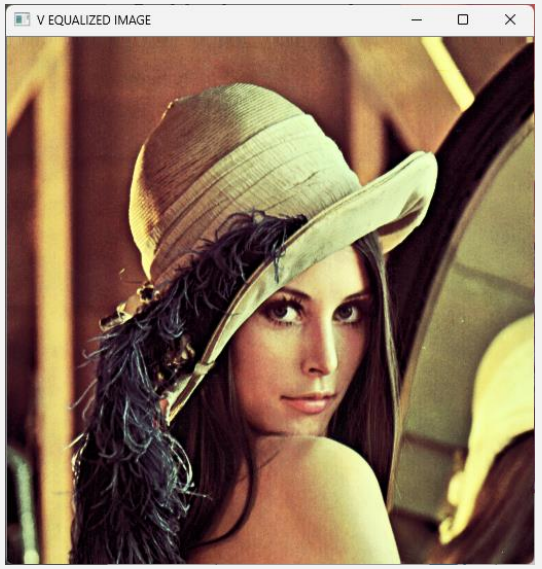


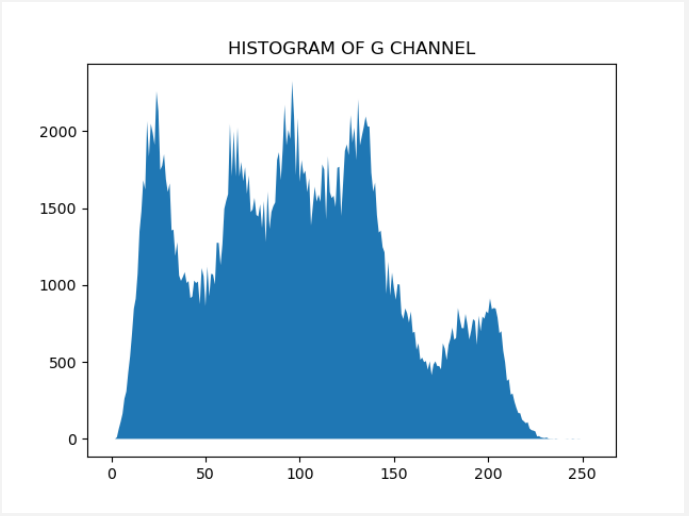
1. Histogram Equalization

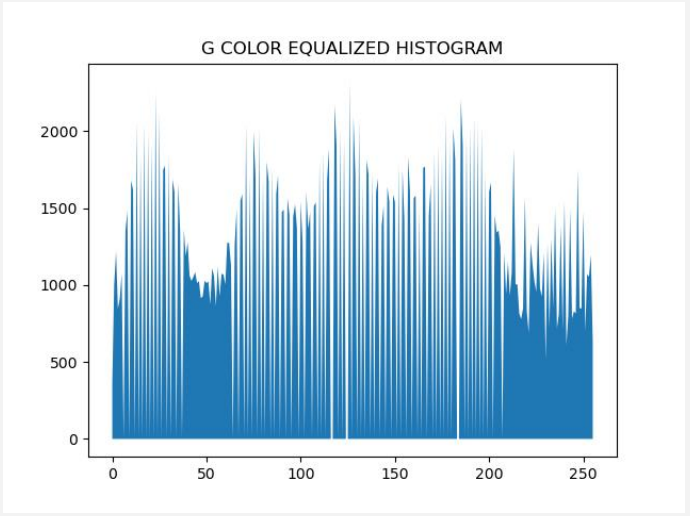
Result('G' 입력)

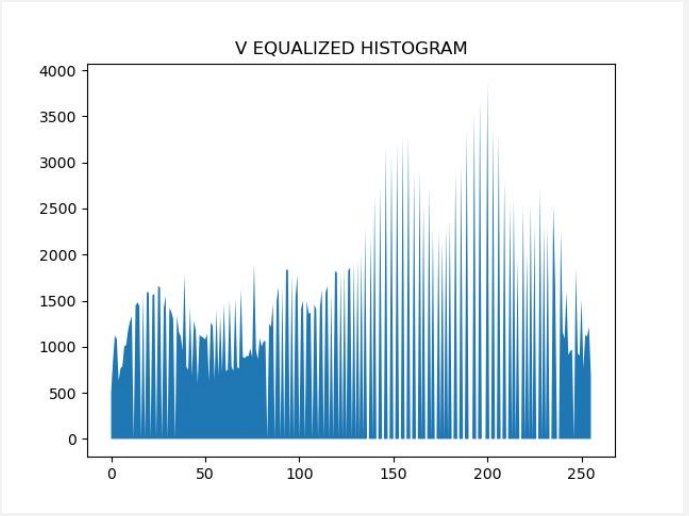






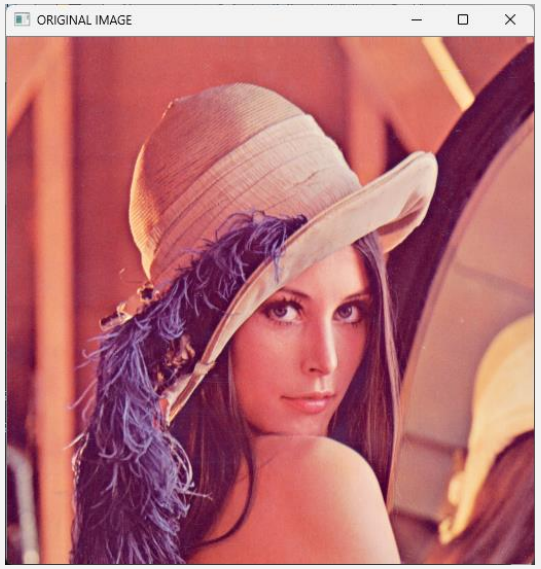


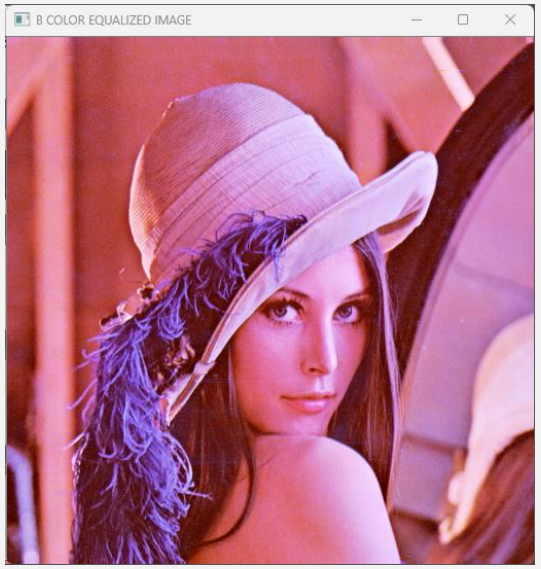


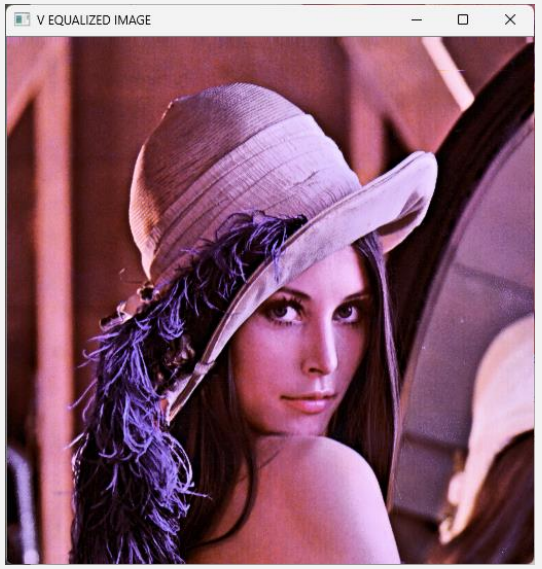


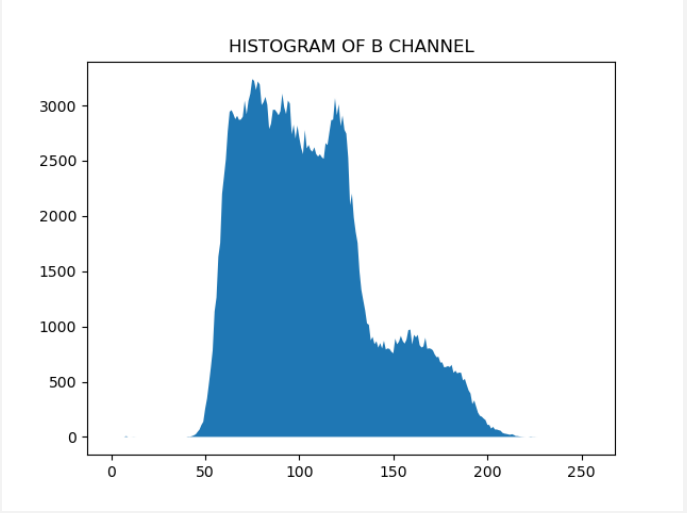
1. Histogram Equalization

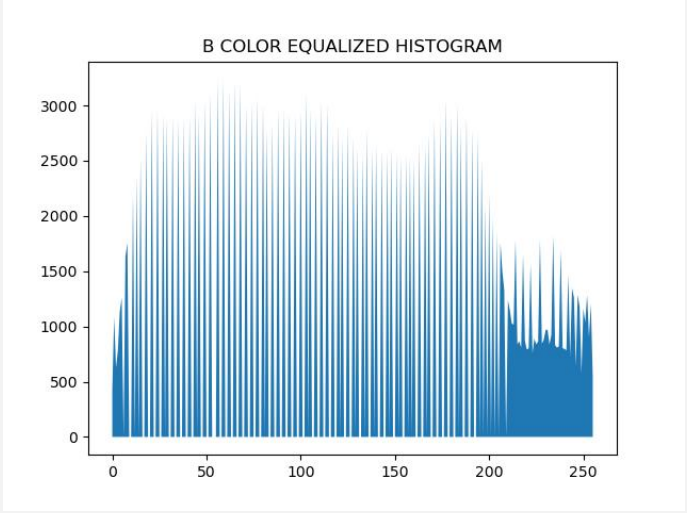
Result(‘B’ 입력)

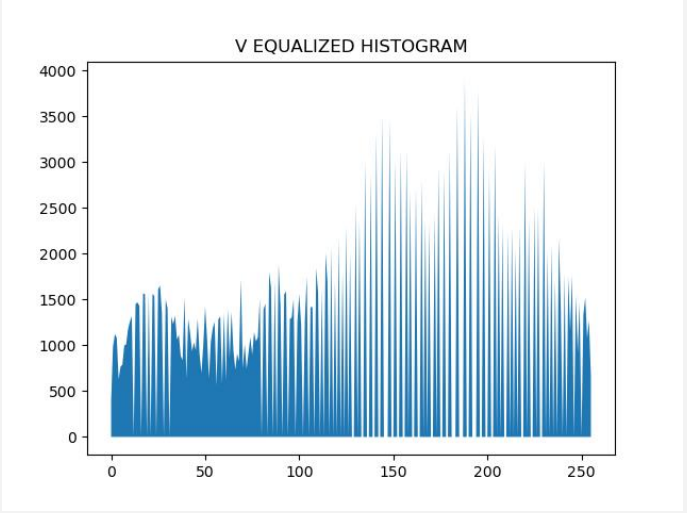












2. Spatial Domain Filtering

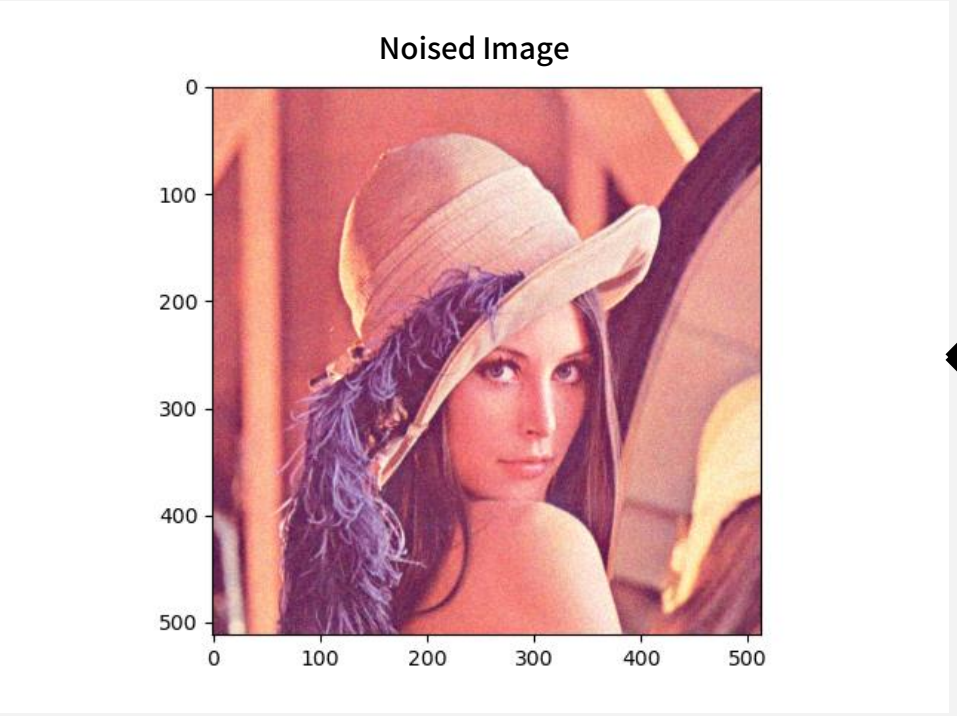
Source Code

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

image = cv2.imread('./data/Lena.png').astype(np.float32) / 255

# 1. 입력 영상에 임의의 노이즈를 입힌다.
noised = (image + 0.2 * np.random.rand(*image.shape).astype(np.float32))
noised = noised.clip(0, 1)
plt.imshow(noised[:, :, [2, 1, 0]])
plt.show()

# 2. Gaussian Filtering 적용 후 결과 출력
gauss_blur = cv2.GaussianBlur(noised, ksize: (7, 7), sigmaX: 0)
plt.imshow(gauss_blur[:, :, [2, 1, 0]])
plt.show()

# 3. Median Filtering 적용 후 결과 출력
median_blur = cv2.medianBlur((noised * 255).astype(np.uint8), ksize: 7)
plt.imshow(median_blur[:, :, [2, 1, 0]])
plt.show()

# 4. Bilateral Filtering 적용 후 결과 출력
bilat = cv2.bilateralFilter(noised, -1, sigmaColor: 0.3, sigmaSpace: 10)
plt.imshow(bilat[:, :, [2, 1, 0]])
plt.show()
```

```python
3 usages  HEEKWON
def calculate_difference_with_absolute(source, target, title):
    difference = cv2.absdiff(source, target)
    plt.imshow(difference[:, :, [2, 1, 0]])
    plt.title(title)
    plt.show()


# 5. 각 결과에 대해 노이즈 입히기 전과 절대값 차이를 취해서 결과 출력
calculate_difference_with_absolute(image, gauss_blur, title: 'Difference after Gaussian Filtering')

median_blur_float = median_blur.astype(np.float32) / 255
calculate_difference_with_absolute(image, median_blur_float, title: 'Difference after Median Filtering')
calculate_difference_with_absolute(image, bilat, title: 'Difference after Bilateral Filtering')
```

2. Spatial Domain Filtering

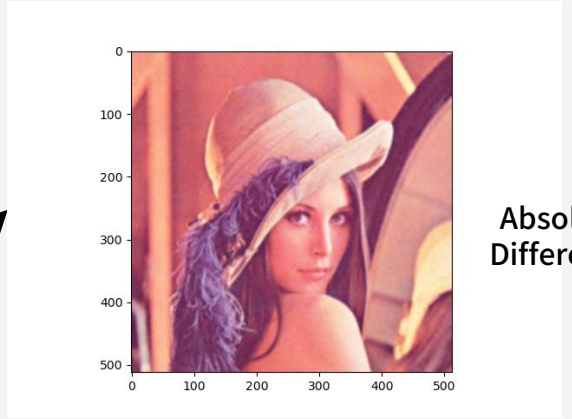
Result



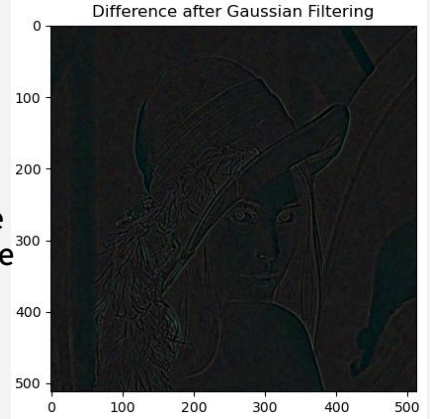
Gaussian Blur

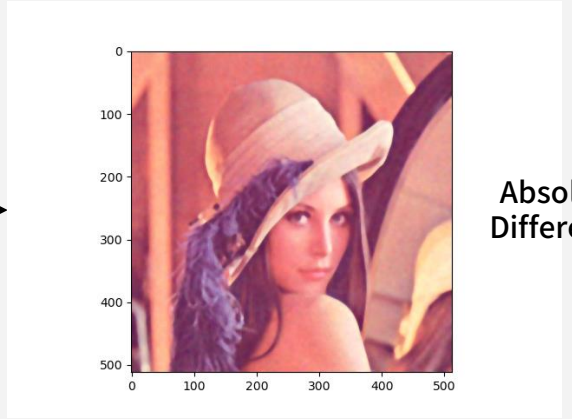
Median Blur

Bilateral Blur

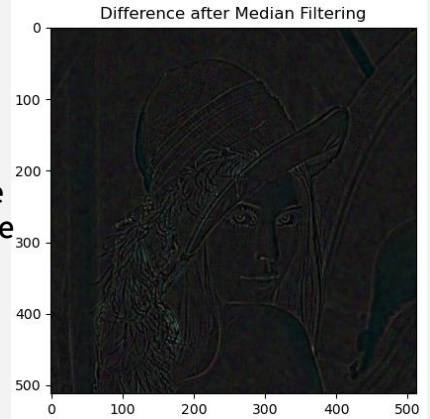


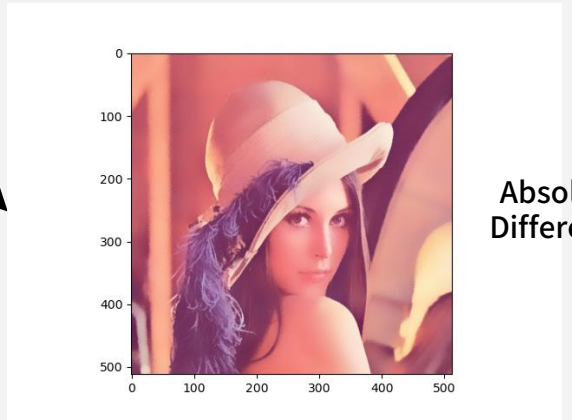
Absolute Difference



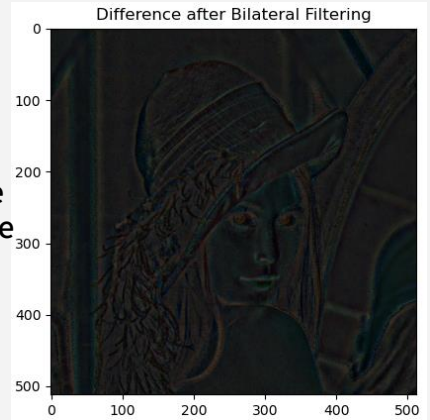


Absolute Difference





Absolute Difference



3. Frequency Domain Filtering

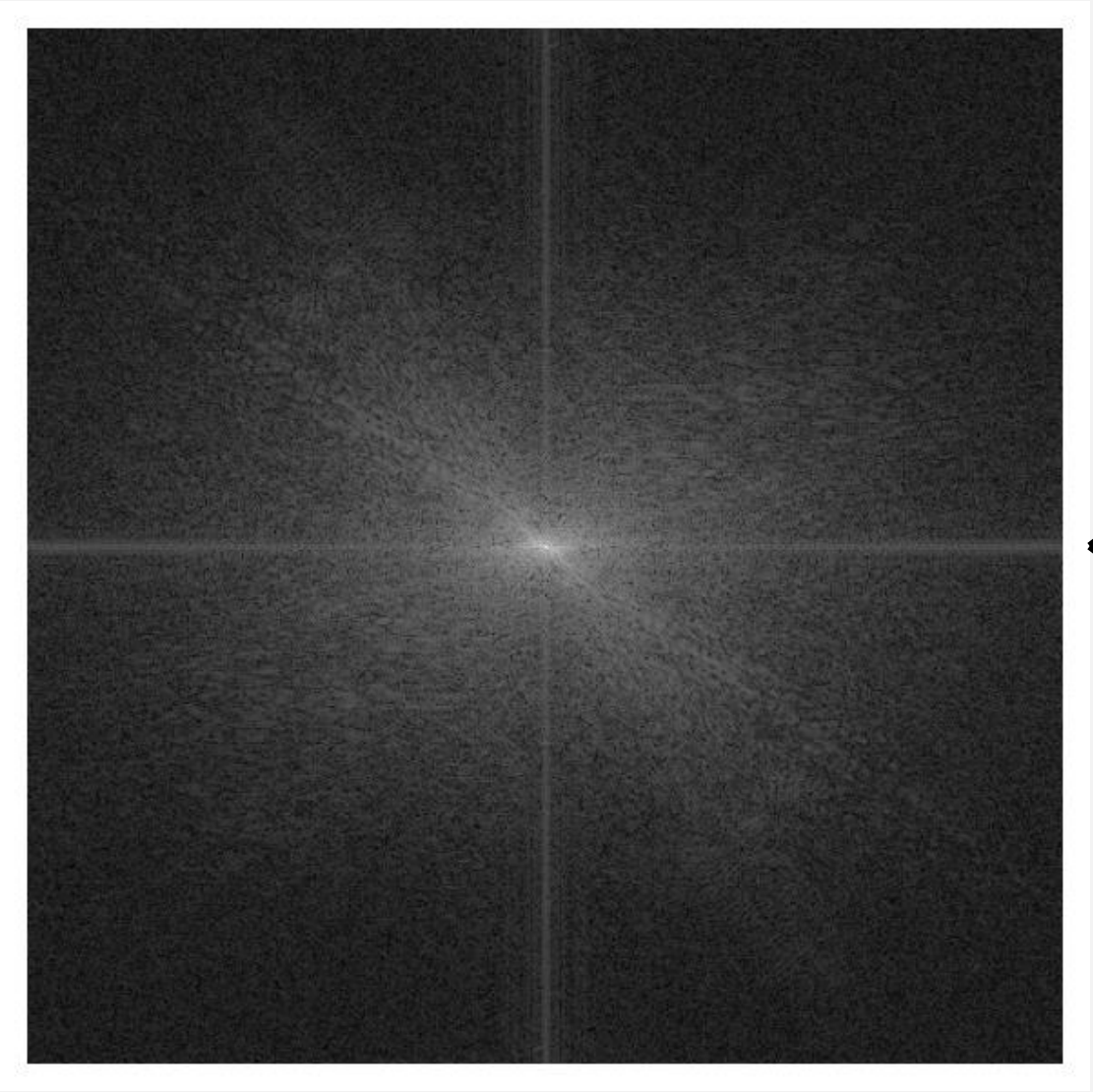
Source Code

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

image = cv2.imread( filename: './data/Lena.png',  flags: 0).astype(np.float32) / 255

# 1. 입력 영상을 DFT를 통해 주파수 도메인으로 변경.
fft = cv2.dft(image, flags=cv2.DFT_COMPLEX_OUTPUT)
shifted = np.fft.fftshift(fft, axes=[0, 1])

magnitude = cv2.magnitude(shifted[:, :, 0], shifted[:, :, 1])
magnitude = np.log(magnitude + 1)

# 2. 주파수 도메인으로 변경한 후 출력
plt.figure(figsize=(6, 6))
plt.axis('off')
plt.imshow(magnitude, cmap='gray')
plt.tight_layout()
plt.show()

# 3. 사용자로부터 반지름 정보 2개 입력 받기.
r_inner = int(input("내부 원의 반지름을 입력하세요: "))
r_outer = int(input("외부 원의 반지름을 입력하세요: "))
```

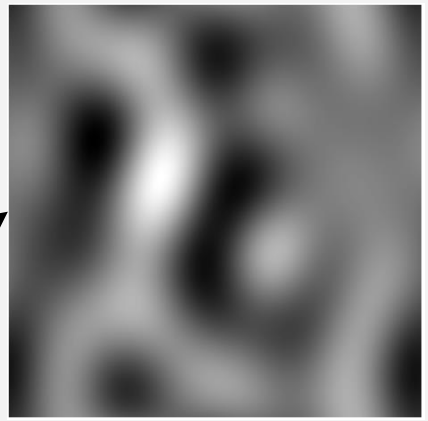
```python
# 4. 영상의 중심을 원의 중심으로 하여 2개의 원 그리기.
rows, cols = image.shape
crow, ccol = rows // 2, cols // 2
mask = np.zeros( shape: (rows, cols, 2), np.uint8)
x, y = np.ogrid[:rows, :cols]
center_distance = (x - crow)**2 + (y - ccol)**2
mask[(center_distance >= r_inner**2) & (center_distance <= r_outer**2)] = 1

# 5. 두 원 사이의 영역을 통과시키는 Band Pass 필터 구현.
filtered = shifted * mask
filtered_shift = np.fft.ifftshift(filtered)
restored = cv2.idft(filtered_shift, flags=cv2.DFT_SCALE | cv2.DFT_REAL_OUTPUT)

# 6. 필터링 결과를 출력.
plt.figure(figsize=(6, 6))
plt.imshow(restored, cmap='gray')
plt.axis('off')
plt.show()
```

3. Frequency Domain Filtering

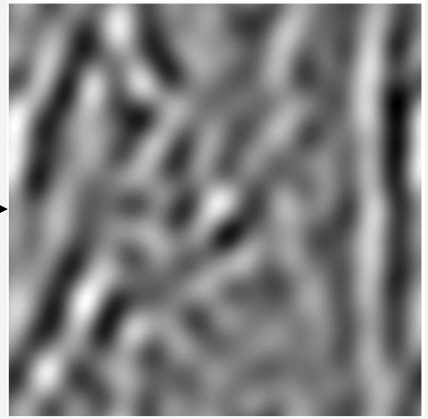
Result



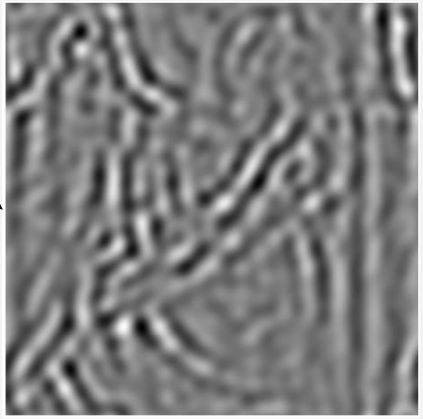
Inner Circle Radius : 2
Outer Circle Radius : 4



Inner Circle Radius : 5
Outer Circle Radius : 10



Inner Circle Radius : 10
Outer Circle Radius : 20



4. Morphological Filter

Source Code

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

image = cv2.imread( filename: './data/Lena.png', flags: 0)

print("이진화 방법을 선택하세요 : ")
print("1 : Otsu's Binarization")
print("2 : Adaptive Threshold")
binary_user_input = input("선택 (1, 2) : ")

# 1. 사용자의 입력을 받아서 Otsu, Median 중 선택
if binary_user_input == '1':
    ret, thresh = cv2.threshold(image, thresh: 0, maxval: 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU)
    title = "Otsu's Binarization"
elif binary_user_input == '2':
    thresh = cv2.adaptiveThreshold(image, maxValue: 255, cv2.ADAPTIVE_THRESH_MEAN_C, cv2.THRESH_BINARY, blockSize: 11, C: 2)
    title = "Adaptive Threshold"

# 2. 선택한 이진화 기법 적용하여 결과 출력
plt.imshow(thresh, cmap='gray')
plt.title(title)
plt.axis('off')
plt.show()

# 3. Erosion, Dilation, Opening, Closing에 대한 선택과 횟수를 입력 받음
print("모폴로지 연산을 선택하세요 : ")
print("1 : Erosion")
print("2 : Dilation")
print("3 : Opening")
print("4 : Closing")
morph_user_input = input("선택 (1, 2, 3, 4) : ")
iterations = int(input("적용 횟수를 입력하세요 : "))
```

```python
kernel = np.ones( shape: (5, 5), np.uint8)

if morph_user_input == '1':
    result = cv2.erode(thresh, kernel, iterations=iterations)
    title = "Erosion"
elif morph_user_input == '2':
    result = cv2.dilate(thresh, kernel, iterations=iterations)
    title = "Dilation"
elif morph_user_input == '3':
    result = cv2.morphologyEx(thresh, cv2.MORPH_OPEN, kernel, iterations=iterations)
    title = "Opening"
elif morph_user_input == "4":
    result = cv2.morphologyEx(thresh, cv2.MORPH_CLOSE, kernel, iterations=iterations)
    title = "Closing"

# 4. 해당 결과 출력
plt.imshow(result, cmap='gray')
plt.title(title)
plt.axis('off')
plt.show()
```

4. Morphological Filter

Result(Otsu Binarization & Erosion)

Erosion : 1

Erosion : 2

Erosion : 3

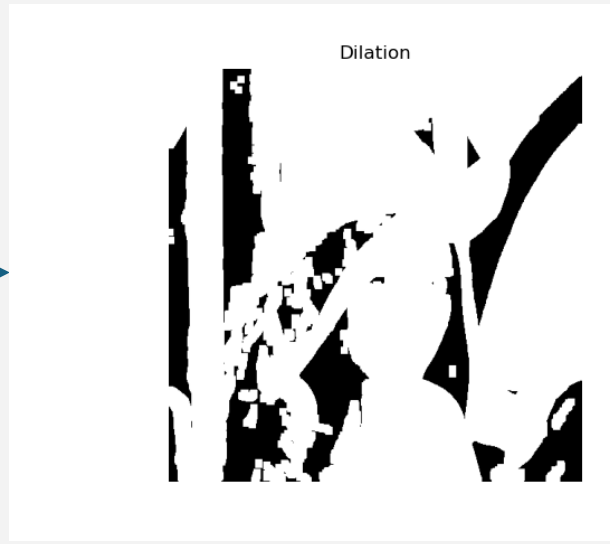
4. Morphological Filter

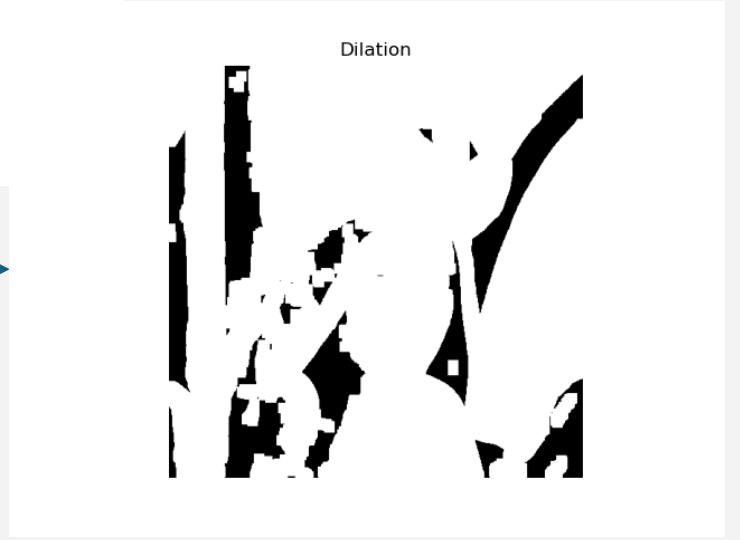
Result(Otsu Binarization & Dilation)

Dilation : 1

Dilation : 2

Dilation : 3





4. Morphological Filter

Result(Otsu Binarization & Opening)

Opening : 1

Opening : 2

Opening : 3

4. Morphological Filter

Result(Otsu Binarization & Closing)

Closing : 1

Closing : 2

Closing : 3

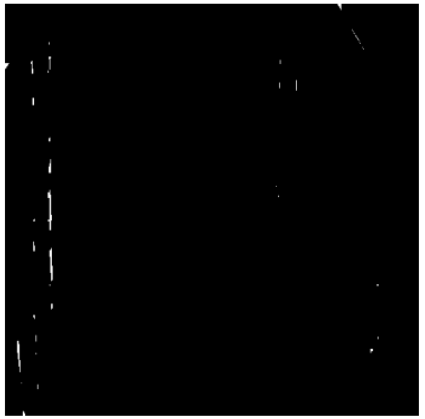
4. Morphological Filter

Result(Adaptive Threshold & Erosion)

Erosion : 1

Erosion : 2

Erosion : 3



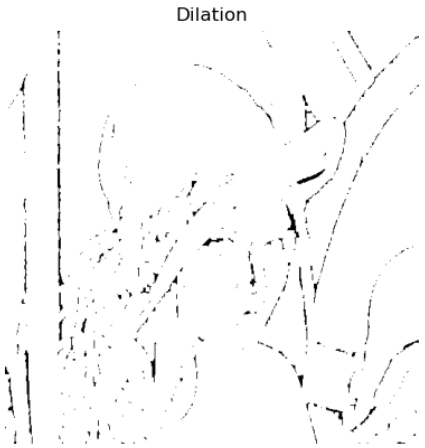
4. Morphological Filter

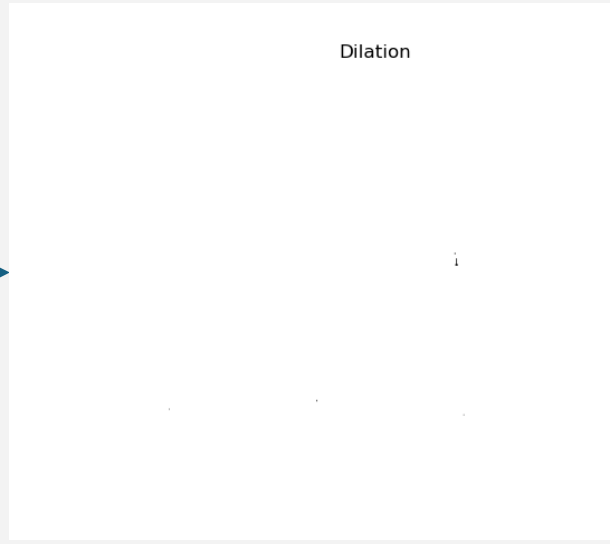
Result(Adaptive Threshold & Dilation)

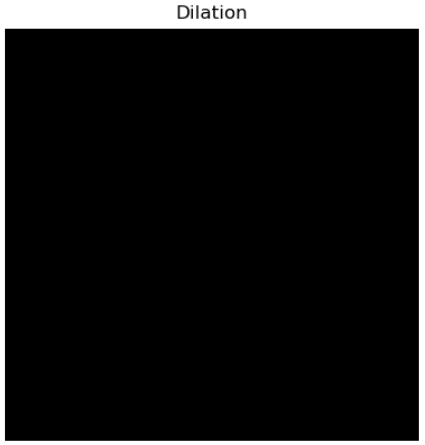
Dilation : 1

Dilation : 2

Dilation : 3







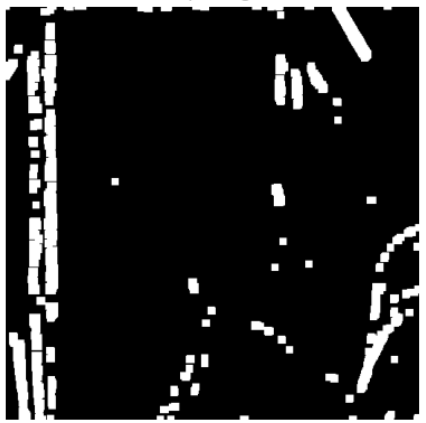
4. Morphological Filter

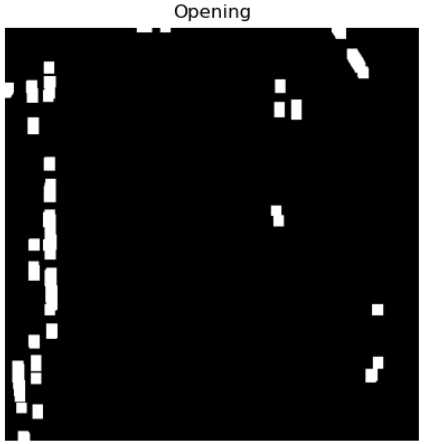
Result(Adaptive Threshold & Opening)

Opening : 1

Opening : 2

Opening : 3





4. Morphological Filter

Result(Adaptive Threshold & Closing)

Closing : 1

Closing : 2

Closing : 3