산업 컴퓨터비전 실제
Homework #1

2023254006 이선경

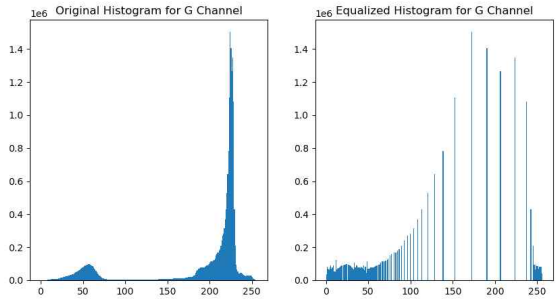
1. 히스토그램 평탄화

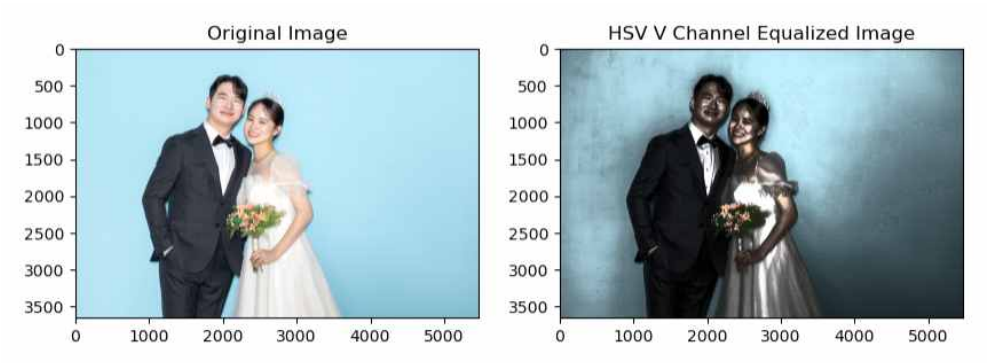
1) 목적 : 히스토그램 평탄화는 이미지의 대비를 향상시켜 세부 사항을 더 명확하게 하기 위한 기법이다.
2) 방법 : RGB 채널 선택과 평탄화, HSV 변환과 V 채널 평탄화
3) 코드

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

image_path = '../data/Wedding.jpg'
img = cv2.imread(image_path)

if img is None:
    print(f'Error: Unable to load image at {image_path}')
else:
    # RGB 채널 분리
    B, G, R = cv2.split(img)

    # 사용자 입력 채널 : 'G' 채널
    channel = G  # 'G' 채널 사용

    # 원본 채널 히스토그램
    plt.figure(figsize=(10, 5))
    plt.subplot( *args: 1, 2, 1)
    plt.hist(channel.ravel(),  bins: 256,  range: [0, 256])
    plt.title('Original Histogram for G Channel')
```

```python
    # 히스토그램 평탄화
    equ = cv2.equalizeHist(channel)

    # 평탄화 후 히스토그램
    plt.subplot( *args: 1, 2, 2)
    plt.hist(equ.ravel(),  bins: 256,  range: [0, 256])
    plt.title('Equalized Histogram for G Channel')
    plt.show()

    # HSV 변환 및 V 채널 평탄화
    img_hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
    h, s, v = cv2.split(img_hsv)
    v_equalized = cv2.equalizeHist(v)
    img_hsv_equalized = cv2.merge((h, s, v_equalized))
    img_equalized = cv2.cvtColor(img_hsv_equalized, cv2.COLOR_HSV2BGR)

    # 원본 및 변환 결과 비교
    plt.figure(figsize=(10, 5))
    plt.subplot( *args: 1, 2, 1)
    plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
    plt.title('Original Image')

    plt.subplot( *args: 1, 2, 2)
    plt.imshow(cv2.cvtColor(img_equalized, cv2.COLOR_BGR2RGB))
    plt.title('HSV V Channel Equalized Image')
    plt.show()
```

4) 결과





결과 분석 : 원본 대비 평탄화된 이미지의 대비가 크게 향상됨. V채널 평탄화로 세부 사항이 더욱 두드러짐

2. 공간 도메인 필터링

1) 목적 : 노이즈가 추가된 이미지에서 노이즈를 제거하고, 원본 이미지의 품질을 복원하기 위해 공간 도메인 필터링 기법을 적용한다.

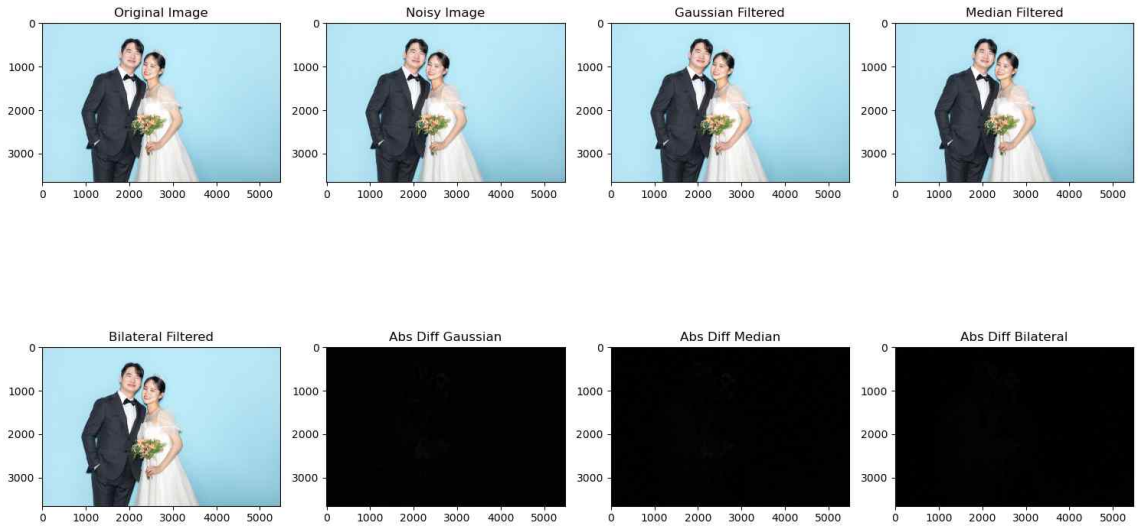
2) 방법 : Gaussain, Median, Bilateral 필터링, 각 필터링 결과 비교

3) 코드

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

image_path = '../data/Wedding.jpg'
img = cv2.imread(image_path)

# 원본 이미지에 Gaussian 노이즈 추가
mean = 0
var = 10
sigma = var ** 0.5
# 이미지에 Gaussian 노이즈를 추가하기 전에 노이즈를 img와 동일한 타입으로 변환
gaussian = np.random.normal(mean, sigma, img.shape).astype(np.float32)  # 노이즈 데이터 타입 변경
noisy_img = cv2.add(img.astype(np.float32), gaussian)  # 두 배열의 타입을 float32로 맞추고 더하기
noisy_img = np.clip(noisy_img, a_min: 0, a_max: 255).astype(np.uint8)  # 결과를 클리핑하여 uint8 타입으로 변환

# 필터 적용
gaussian_filtered = cv2.GaussianBlur(noisy_img, ksize: (5, 5), sigmaX: 1.5)
median_filtered = cv2.medianBlur(noisy_img.astype(np.uint8), ksize: 5)
bilateral_filtered = cv2.bilateralFilter(noisy_img.astype(np.uint8), d: 9, sigmaColor: 75, sigmaSpace: 75)

# 결과 및 차이 계산
abs_diff_gaussian = cv2.absdiff(img, gaussian_filtered)
abs_diff_median = cv2.absdiff(img, median_filtered)
abs_diff_bilateral = cv2.absdiff(img, bilateral_filtered)
```

```python
# 결과 이미지 출력
plt.figure(figsize=(15, 10))

plt.subplot(*args: 2, 4, 1)
plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
plt.title('Original Image')

plt.subplot(*args: 2, 4, 2)
plt.imshow(cv2.cvtColor(noisy_img.astype(np.uint8), cv2.COLOR_BGR2
plt.title('Noisy Image')

plt.subplot(*args: 2, 4, 3)
plt.imshow(cv2.cvtColor(gaussian_filtered, cv2.COLOR_BGR2RGB))
plt.title('Gaussian Filtered')

plt.subplot(*args: 2, 4, 4)
plt.imshow(cv2.cvtColor(median_filtered, cv2.COLOR_BGR2RGB))
plt.title('Median Filtered')

plt.subplot(*args: 2, 4, 5)
plt.imshow(cv2.cvtColor(bilateral_filtered, cv2.COLOR_BGR2RGB))
plt.title('Bilateral Filtered')

plt.subplot(*args: 2, 4, 6)
plt.imshow(cv2.cvtColor(abs_diff_gaussian, cv2.COLOR_BGR2RGB))
plt.title('Abs Diff Gaussian')

plt.subplot(*args: 2, 4, 7)
plt.imshow(cv2.cvtColor(abs_diff_median, cv2.COLOR_BGR2RGB))
plt.title('Abs Diff Median')

plt.subplot(*args: 2, 4, 8)
plt.imshow(cv2.cvtColor(abs_diff_bilateral, cv2.COLOR_BGR2RGB))
plt.title('Abs Diff Bilateral')

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

4) 결과



결과 분석 : 각 필터링 기법에 따른 노이즈 제거 효과와 세부 사항 보존 정도의 차이

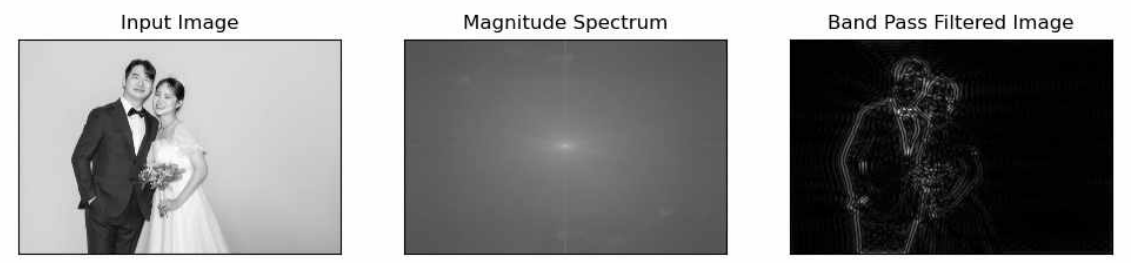
3. 주파수 도메인 필터링

1) 목적 : 이미지에서 원하는 주파수 범위의 성분을 추출하거나 제거하여 이미지의 특정 특성을 강조하거나 약화시키기 위함이다.
2) 방법 : DFT를 사용한 주파수 변환, Band pass 필터의 구현과 적용
3) 코드

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

image_path = '../data/Wedding.jpg'
img = cv2.imread(image_path, flags: 0)  # Grayscale로 로드

# DFT 수행
dft = cv2.dft(np.float32(img), flags=cv2.DFT_COMPLEX_OUTPUT)
dft_shift = np.fft.fftshift(dft)

# 주파수 변환 결과의 진폭 스펙트럼 계산
magnitude_spectrum = 20 * np.log(cv2.magnitude(dft_shift[:, :, 0], dft_shift[:, :, 1]))

# 반지름 값
radius1, radius2 = 30, 80

# Band Pass 필터 마스크 생성 - 두 반지름 사이의 값만 통과
rows, cols = img.shape
crow, ccol = rows // 2, cols // 2
mask = np.zeros( shape: (rows, cols, 2), np.uint8)
cv2.circle(mask, center: (ccol, crow), radius2, color: (1, 1), -1)
cv2.circle(mask, center: (ccol, crow), radius1, color: (0, 0), -1)
```

```python
# 마스크를 DFT 결과에 적용
fshift = dft_shift * mask

# 역 FFT(IFT)를 통해 이미지 복구
f_ishift = np.fft.ifftshift(fshift)
img_back = cv2.idft(f_ishift)
img_back = cv2.magnitude(img_back[:, :, 0], img_back[:, :, 1])

# 결과 이미지 출력
plt.figure(figsize=(12, 6))

plt.subplot(131), plt.imshow(img, cmap='gray')
plt.title('Input Image'), plt.xticks([]), plt.yticks([])

plt.subplot(132), plt.imshow(magnitude_spectrum, cmap='gray')
plt.title('Magnitude Spectrum'), plt.xticks([]), plt.yticks([])

plt.subplot(133), plt.imshow(img_back, cmap='gray')
plt.title('Band Pass Filtered Image'), plt.xticks([]), plt.yticks([])

plt.show()
```

4) 결과



결과 분석 : 필터링을 통해 강조된 이미지의 주파수 성분, 원본 이미지와 필터링된 이미지 사이의 비교

4. 모폴로지 필터

1) 목적 : 이미지의 구조적 요소를 분석하고, 특정 형태의 객체를 분리하거나 강조하기 위해 모폴로지 필터를 사용한다.
2) 방법 : Otsu와 Adaptive 이진화 방법의 적용, Erosion, Dilation, Opening, Closing 연산의 선택과 적용
3) 코드

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

image_path = '../data/Wedding.jpg'
img = cv2.imread(image_path, flags: 0)  # 그레이스케일로 로드

# 이진화 방법 선택: 'otsu' 또는 'adaptive'
binarization_method = 'otsu'  # 예시로 'otsu'를 사용

# Otsu의 이진화
if binarization_method == 'otsu':
    _, bin_img = cv2.threshold(img, thresh: 0, maxval: 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU)
# Adaptive 이진화 (median)
elif binarization_method == 'adaptive':
    bin_img = cv2.adaptiveThreshold(img, maxValue: 255, cv2.ADAPTIVE_THRESH_MEAN_C, cv2.THRESH_BINARY, blockSize: 11, C: 2)

# 모폴로지 연산 선택과 횟수 입력
morphology_operation = 'opening'  # 'erosion', 'dilation', 'opening', 'closing'
iterations = 2  # 연산을 적용할 횟수

# 모폴로지 연산 수행
kernel = np.ones( shape: (3,3), np.uint8)
if morphology_operation == 'erosion':
    result_img = cv2.erode(bin_img, kernel, iterations=iterations)
elif morphology_operation == 'dilation':
    result_img = cv2.dilate(bin_img, kernel, iterations=iterations)
elif morphology_operation == 'opening':
    result_img = cv2.morphologyEx(bin_img, cv2.MORPH_OPEN, kernel, iterations=iterations)
elif morphology_operation == 'closing':
    result_img = cv2.morphologyEx(bin_img, cv2.MORPH_CLOSE, kernel, iterations=iterations)

# 결과 출력
plt.figure(figsize=(15, 5))

plt.subplot(131), plt.imshow(img, cmap='gray'), plt.title('Original')
plt.xticks([]), plt.yticks([])

plt.subplot(132), plt.imshow(bin_img, cmap='gray'), plt.title('Binarization Result')
plt.xticks([]), plt.yticks([])

plt.subplot(133), plt.imshow(result_img, cmap='gray'), plt.title(f'{morphology_operation.capitalize()} Result')
plt.xticks([]), plt.yticks([])

plt.show()
```

4) 결과

결과 분석 : 이진화 방법에 따른 이미지의 변화, 모폴로지 연산에 따른 구조적 특성의 변화 및 개선