영상처리 프로그래밍

6주차

소개

- 조교 : 김형훈
  - 연구실 : A1409 (운영체제 연구실)
  - 이메일 : khh8996@naver.com
  - 전화번호 : 010-5316-7953

- 담당 교수 : 김백섭 교수님

목차

- 픽셀 단위 산술 연산
- 룩업 테이블 연산
- 히스토그램
- 실습
- 과제

픽셀 단위 산술 연산

- 상수 값에 의한 산술 연산
  - C는 임의의 상수 값이며, 0 ~ 255 사이의 값으로 선정
  - InImg : 원본 영상, OutImg : 결과 영상

연산	방법	영상 결과
덧셈	OutImg = InImg + C	밝아짐
뺄셈	OutImg = InImg - C	어두워짐
곱셈	OutImg = InImg * C	대비 증가
나눗셈	OutImg = InImg / C	대비 감소

픽셀 단위 산술 연산

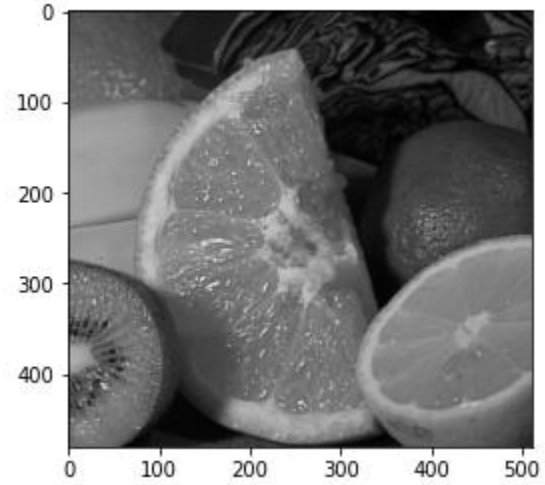
- OpenCV 함수를 이용한 산술 연산
  - OpenCV의 산술 연산 함수를 이용함으로써 픽셀 값의 범위(0 ~ 255)에 제한할 수 있음
    - 산술 연산 수행할 때, 0보다 작은 값이 되면 0으로 치환하고 255보다 큰 값이 되면 255로 치환
    - 영상 값 클램핑 처리
  - 덧셈 : cv.add(src1, src2)
  - 뺄셈 : cv.subtract(src1, src2)
  - 곱셈 : cv.multiply(src1, src2)
  - 나눗셈 : cv.divide(src1, src2)
    - src1 : 이미지 파일 또는 임의의 상수 값
    - src2 : 이미지 파일 또는 임의의 상수 값

픽셀 단위 산술 연산

- 픽셀 단위 산술 연산 - 예시

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

# fruits 이미지 파일 불러옴
img = cv2.imread('./fruits.bmp')

# BGR에서 GRAY로 변경
img_gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

# 이미지 형태 출력 (height, width, channel)
print('Shape: ', img_gray.shape)

# 이미지 출력
plt.imshow(img_gray, cmap='gray')
plt.show()
```

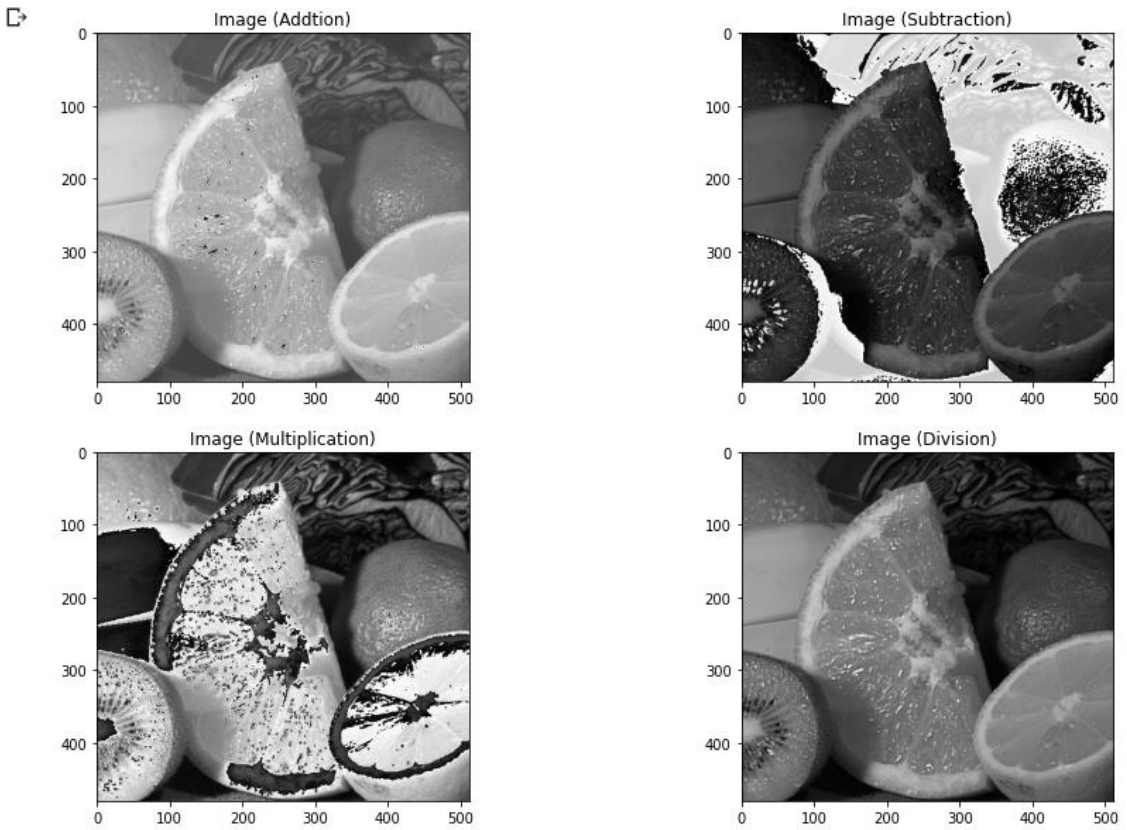
```
Shape:  (480, 512)
```



픽셀 단위 산술 연산

- 픽셀 단위 산술 연산 - 예시 (상수 값에 의한 산술 연산)

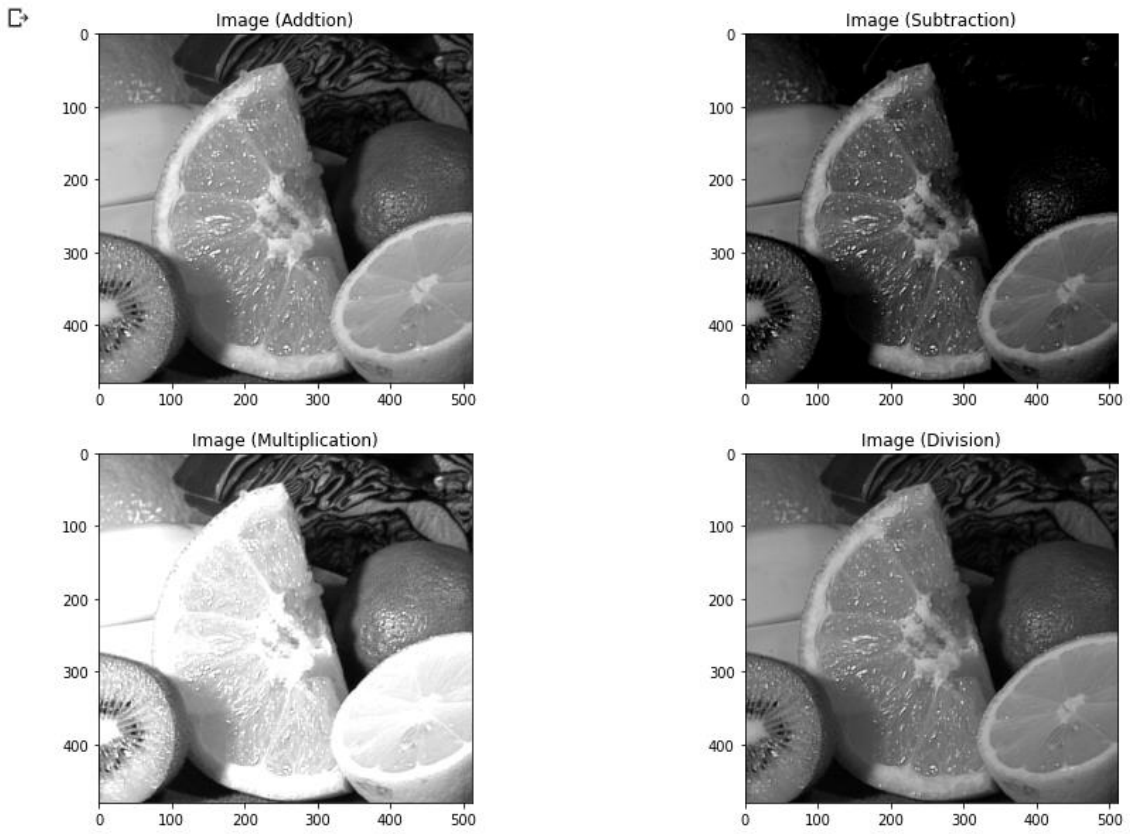
```
fig = plt.figure(figsize=(15, 10))

# 이미지 산술 연산
img_gray_add = img_gray + 60
img_gray_sub = img_gray - 60
img_gray_mul = img_gray * 2
img_gray_div = img_gray / 2

# 이미지 화소 값을 정수로 변환
img_gray_div = img_gray_div.astype(np.int)

# 덧셈
ax1 = fig.add_subplot(221)
ax1.set_title('Image (Addtion)')
ax1.imshow(img_gray_add, cmap='gray')

# 뺄셈
ax2 = fig.add_subplot(222)
ax2.set_title('Image (Subtraction)')
ax2.imshow(img_gray_sub, cmap='gray')

# 곱셈
ax3 = fig.add_subplot(223)
ax3.set_title('Image (Multiplication)')
ax3.imshow(img_gray_mul, cmap='gray')

# 나눗셈
ax4 = fig.add_subplot(224)
ax4.set_title('Image (Division)')
ax4.imshow(img_gray_div, cmap='gray')

plt.show()
```



픽셀 단위 산술 연산

- 픽셀 단위 산술 연산 - 예시 (OpenCV 함수를 이용한 산술 연산)

```
fig = plt.figure(figsize=(15, 10))

# 이미지 산술 연산
img_gray_add = cv2.add(img_gray, 60)
img_gray_sub = cv2.subtract(img_gray, 60)
img_gray_mul = cv2.multiply(img_gray, 2)
img_gray_div = cv2.divide(img_gray, 2)

# 이미지 화소 값을 정수로 변환
img_gray_div = img_gray_div.astype(np.int)

# 덧셈
ax1 = fig.add_subplot(221)
ax1.set_title('Image (Addtion)')
ax1.imshow(img_gray_add, cmap='gray')

# 뺄셈
ax2 = fig.add_subplot(222)
ax2.set_title('Image (Subtraction)')
ax2.imshow(img_gray_sub, cmap='gray')

# 곱셈
ax3 = fig.add_subplot(223)
ax3.set_title('Image (Multiplication)')
ax3.imshow(img_gray_mul, cmap='gray')

# 나눗셈
ax4 = fig.add_subplot(224)
ax4.set_title('Image (Division)')
ax4.imshow(img_gray_div, cmap='gray')

plt.show()
```



룩업 테이블 연산

- 룩업 테이블 (Lookup Table(LUT)) 연산
  - 룩업 테이블 연산을 수행하기 위해 룩업 테이블 생성
  - LUT()를 이용하여 룩업 테이블 연산 수행
    - cv2.LUT(image, table)
      - image : 이미지 파일
      - table : 룩업 테이블

룩업 테이블 생성

```
unsigned char LUT[256];

for (int i = 0; i < 256; i++) {
    int temp = (int)(i * value);
    LUT[i] = temp > 255 ? 255 : temp
}
```

룩업 테이블 연산

```
for (int i = 0; i < height; i++) {
    for (int j = 0; j < width; j++) {
        OutImg[i][j] = LUT[InImg[i][j]];
    }
}
```

룩업 테이블 연산

- 룩업 테이블 (Lookup Table(LUT)) 연산

- C Programming

```c
unsigned char LUT[256];

for (int i = 0; i < 256; i++) {
    int temp = (int)(i * value);
    LUT[i] = temp > 255 ? 255 : temp
}

for (int i = 0; i < height; i++) {
    for (int j = 0; j < width; j++) {
        OutImg[i][j] = LUT[InImg[i][j]];
    }
}
```

- Python Programming

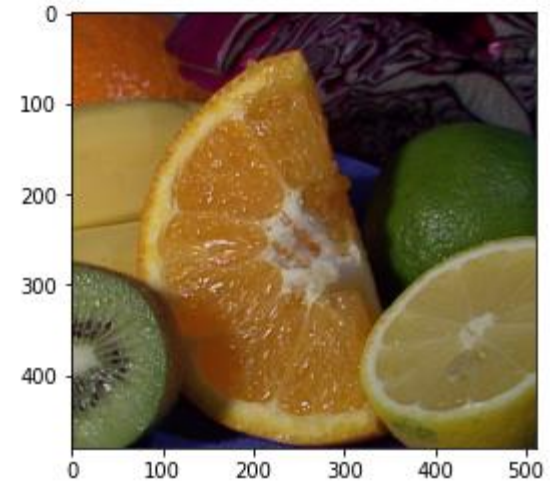
```python
table = np.zeros(256).astype(np.int)

for i in range(256):
    temp = int(i * value)
    table[i] = 255 if temp > 255 else temp

OutImg = cv2.LUT(Inimg, table)
```

룩업 테이블 연산

- 룩업 테이블 연산 - 예시

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

# fruits 이미지 파일 불러옴
img = cv2.imread('./fruits.bmp')

# BGR에서 RGB로 변경
img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

# 이미지 형태 출력 (height, width, channel)
print('Shape: ', img_rgb.shape)

# 이미지 출력
plt.imshow(img_rgb)
plt.show()
```

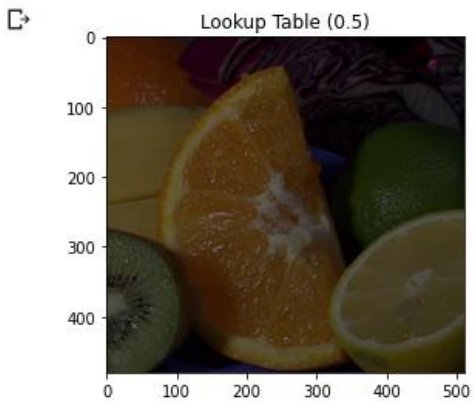
```
Shape:  (480, 512, 3)
```



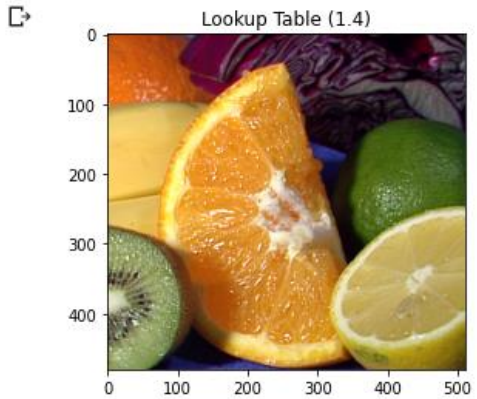
룩업 테이블 연산

- 룩업 테이블 연산 - 예시

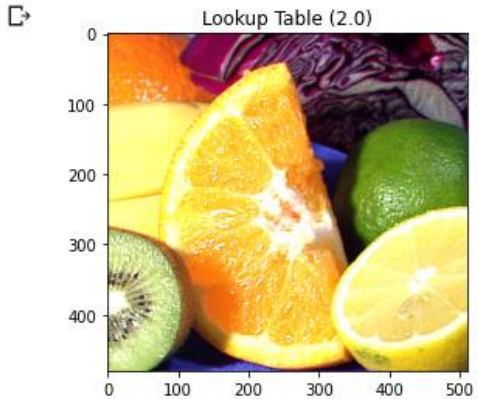
```python
# 모든 요소가 0이며 256 크기의 1차원 배열 생성
table = np.zeros(256).astype(np.int)

# 룩업 테이블 생성
for i in range(256):
  temp = int(i * 0.5)
  table[i] = 255 if temp > 255 else temp

# 룩업 테이블 연산
img_lut = cv2.LUT(img_rgb, table)

plt.title('Lookup Table (0.5)')
plt.imshow(img_lut)
plt.show()
```




```python
# 모든 요소가 0이며 256 크기의 1차원 배열 생성
table = np.zeros(256).astype(np.int)

# 룩업 테이블 생성
for i in range(256):
  temp = int(i * 1.4)
  table[i] = 255 if temp > 255 else temp

# 룩업 테이블 연산
img_lut = cv2.LUT(img_rgb, table)

plt.title('Lookup Table (1.4)')
plt.imshow(img_lut)
plt.show()
```




```python
# 모든 요소가 0이며 256 크기의 1차원 배열 생성
table = np.zeros(256).astype(np.int)

# 룩업 테이블 생성
for i in range(256):
  temp = int(i * 2.0)
  table[i] = 255 if temp > 255 else temp

# 룩업 테이블 연산
img_lut = cv2.LUT(img_rgb, table)

plt.title('Lookup Table (2.0)')
plt.imshow(img_lut)
plt.show()
```



룩업 테이블 연산

- 룩업 테이블 (Lookup Table(LUT)) 연산
  - 감마(γ) 상관관계에 의한 영상 변환

- C Programming

```c
unsigned char LUT[256];

for (int i = 0; i < 256; i++) {
    LUT[i] = pow((double)(i)/255.0, 1.0/GAMMA) * 255;
}

for (int i = 0; i < height; i++) {
    for (int j = 0; j < width; j++) {
        OutImg[i][j] = LUT[InImg[i][j]];
    }
}
```

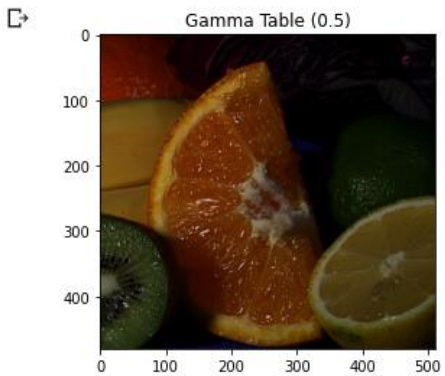
- Python Programming

```python
table = np.zeros(256).astype(np.int)

for i in range(256):
    table[i] = ((i / 255.0) ** (1.0 / GAMMA)) * 255

OutImg = cv2.LUT(Inimg, table)
```

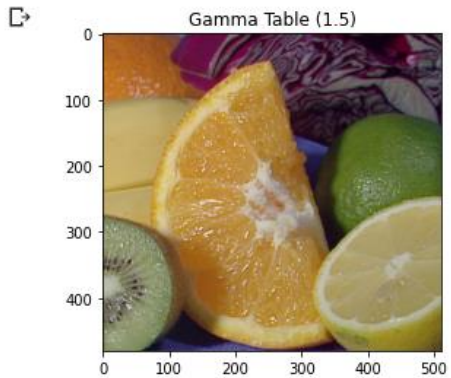
룩업 테이블 연산

- 룩업 테이블 연산 - 예시 (감마 테이블)

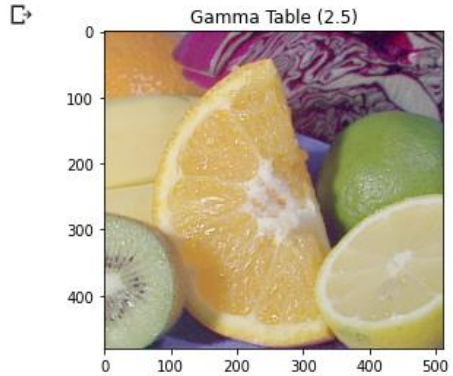
```
gamma = 0.5

# 모든 요소가 0이며 256 크기의 1차원 배열 생성
table = np.zeros(256).astype(np.int)

# 감마 테이블 생성
for i in range(256):
  table[i] = ((i / 255.0) ** (1.0 / gamma)) * 255

# 감마 테이블을 이용한 룩업 테이블 연산
img_gamma = cv2.LUT(img_rgb, table)

plt.title('Gamma Table (0.5)')
plt.imshow(img_gamma)
plt.show()
```




```
gamma = 1.5

# 모든 요소가 0이며 256 크기의 1차원 배열 생성
table = np.zeros(256).astype(np.int)

# 감마 테이블 생성
for i in range(256):
  table[i] = ((i / 255.0) ** (1.0 / gamma)) * 255

# 감마 테이블을 이용한 룩업 테이블 연산
img_gamma = cv2.LUT(img_rgb, table)

plt.title('Gamma Table (1.5)')
plt.imshow(img_gamma)
plt.show()
```




```
gamma = 2.5

# 모든 요소가 0이며 256 크기의 1차원 배열 생성
table = np.zeros(256).astype(np.int)

# 감마 테이블 생성
for i in range(256):
  table[i] = ((i / 255.0) ** (1.0 / gamma)) * 255

# 감마 테이블을 이용한 룩업 테이블 연산
img_gamma = cv2.LUT(img_rgb, table)

plt.title('Gamma Table (2.5)')
plt.imshow(img_gamma)
plt.show()
```



룩업 테이블 연산

- 룩업 테이블 (Lookup Table(LUT)) 연산
  - 포스터라이징(Posterizing) 변환
    - C Programming

```c
unsigned char LUT[256];
double step = 256 / level;
double inc = 255 / (level - 1);

for (int i = 0; i < 256; i++) {
    LUT[i] = int(i / step) * inc;
}

for (int i = 0; i < height; i++) {
    for (int j = 0; j < width; j++) {
        OutImg[i][j] = LUT[InImg[i][j]];
    }
}
```

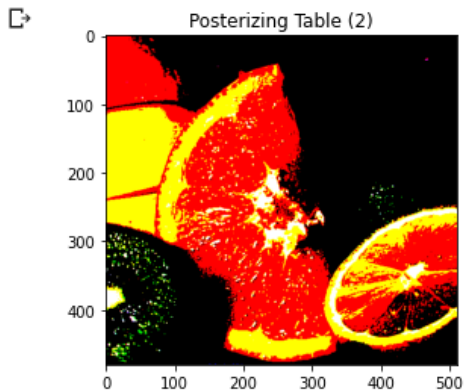
    - Python Programming

```python
table = np.zeros(256).astype(np.int)
step = 256 / level
inc = 255 / (level - 1)

for i in range(256):
    table[i] = int(i / step) * inc

OutImg = cv2.LUT(Inimg, table)
```

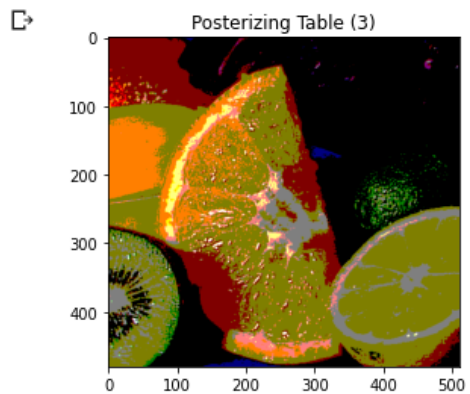
룩업 테이블 연산

- 룩업 테이블 연산 - 예시 (포스터라이징)

```
level = 2

step = 256 / level
inc = 255 / (level - 1)

# 모든 요소가 0이며 256 크기의 1차원 배열 생성
table = np.zeros(256).astype(np.int)

# 포스터라이징 테이블 생성
for i in range(256):
  table[i] = int(i/step) * inc

# 포스터라이징 테이블을 이용한 룩업 테이블 연산
img_posterizing = cv2.LUT(img_rgb, table)

plt.title('Posterizing Table (2)')
plt.imshow(img_posterizing)
plt.show()
```




```
level = 3

step = 256 / level
inc = 255 / (level - 1)

# 모든 요소가 0이며 256 크기의 1차원 배열 생성
table = np.zeros(256).astype(np.int)

# 포스터라이징 테이블 생성
for i in range(256):
  table[i] = int(i/step) * inc

# 포스터라이징 테이블을 이용한 룩업 테이블 연산
img_posterizing = cv2.LUT(img_rgb, table)

plt.title('Posterizing Table (3)')
plt.imshow(img_posterizing)
plt.show()
```



히스토그램

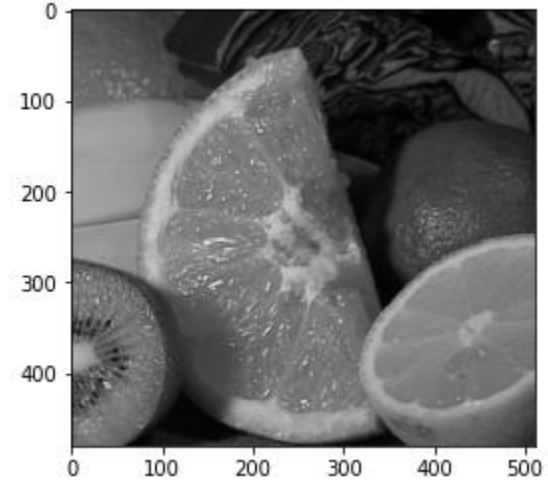
- 히스토그램 (Histrogram)
  - 이미지의 밝기의 분포를 그래프로 표현한 방식
  - 이미지 전체의 밝기 분포와 채도를 알 수 있음
  - cv2.calcHist(image, channel, mask, histSize, ranges)
    - image : 이미지 파일. [img]처럼 리스트로 감싸서 표현
    - channel : 이미지 파일의 채널. 1채널: [0], 2채널: [0, 1], 3채널: [0, 1, 2]
    - mask : 이미지의 분석 영역. None이면 전체 영역
    - histSize : 히스토그램 그래프의 x 축의 간격. [256]
    - ranges : 각 픽셀이 가질 수 있는 범위. [0, 256]

히스토그램

- 히스토그램 - 예시 (Gray)

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

# fruits 이미지 파일 불러옴
img = cv2.imread('./fruits.bmp')

# BGR에서 GRAY로 변경
img_gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

# 이미지 형태 출력 (height, width, channel)
print('Shape: ', img_gray.shape)

# 이미지 출력
plt.imshow(img_gray, cmap='gray')
plt.show()
```

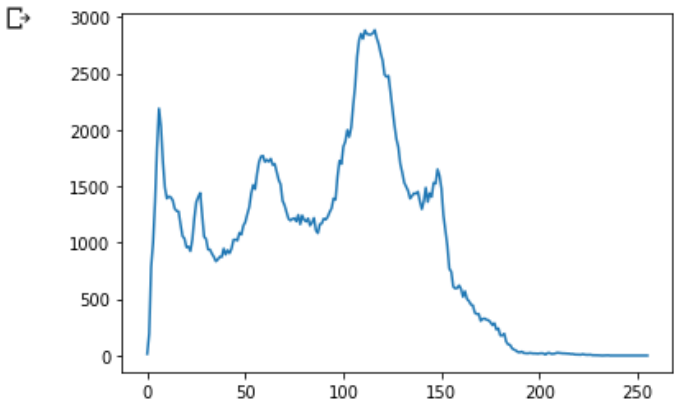
```
Shape:  (480, 512)
```



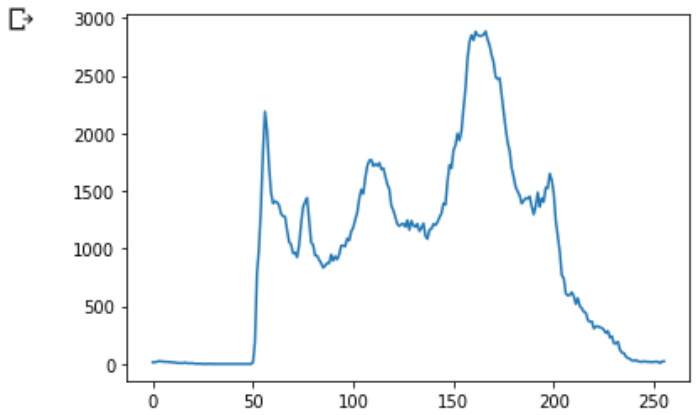
히스토그램

- 히스토그램 - 예시 (Gray)

```
# 히스토그램 계산 및 그리기
hist = cv2.calcHist([img_gray], [0], None, [256], [0, 256])
plt.plot(hist)
plt.show()
```




```
# 이미지 각 화소에 50씩 더함
img_add = img_gray + 50

# 히스토그램 계산 및 그리기
hist = cv2.calcHist([img_add], [0], None, [256], [0, 256])
plt.plot(hist)
plt.show()
```

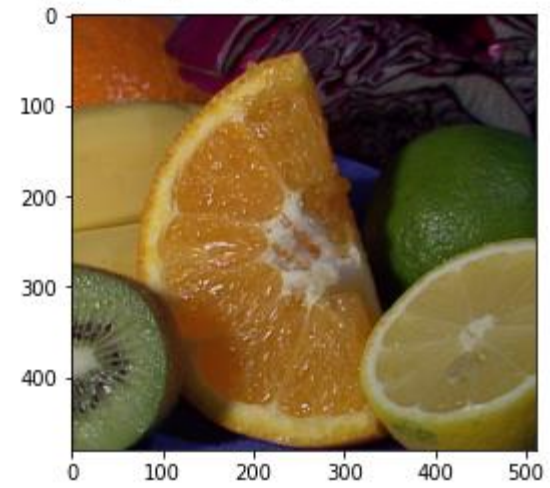


히스토그램

- 히스토그램 - 예시 (RGB)

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

# fruits 이미지 파일 불러옴
img = cv2.imread('./fruits.bmp')

# BGR에서 RGB로 변경
img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

# 이미지 형태 출력 (height, width, channel)
print('Shape: ', img_rgb.shape)

# 이미지 출력
plt.imshow(img_rgb)
plt.show()
```

```
Shape:  (480, 512, 3)
```

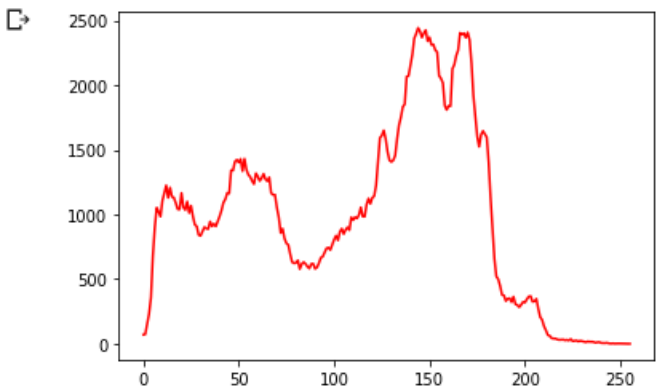


히스토그램

- 히스토그램 - 예시 (RGB)

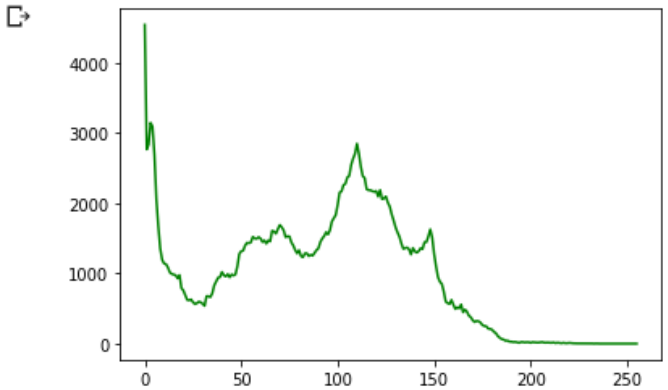
- Red

```
# 이미지 채널 분리 (RGB)
r, g, b = cv2.split(img_rgb)

# 컬러 히스토그램 계산 및 그리기
hist = cv2.calcHist([r], [0], None, [256], [0, 256])
plt.plot(hist, color='red')
plt.show()

plt.show()
```



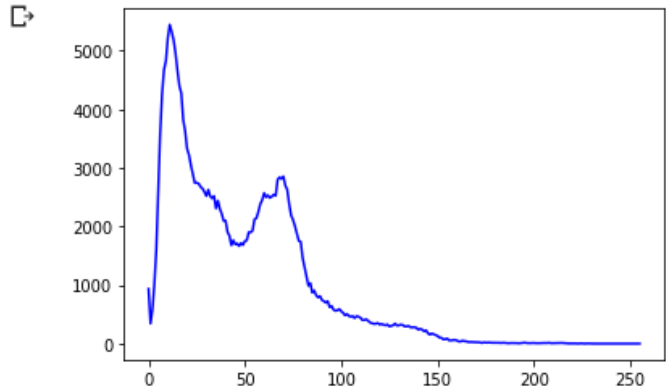
- Green

```
# 이미지 채널 분리 (RGB)
r, g, b = cv2.split(img_rgb)

# 컬러 히스토그램 계산 및 그리기
hist = cv2.calcHist([g], [0], None, [256], [0, 256])
plt.plot(hist, color='green')
plt.show()

plt.show()
```



- Blue

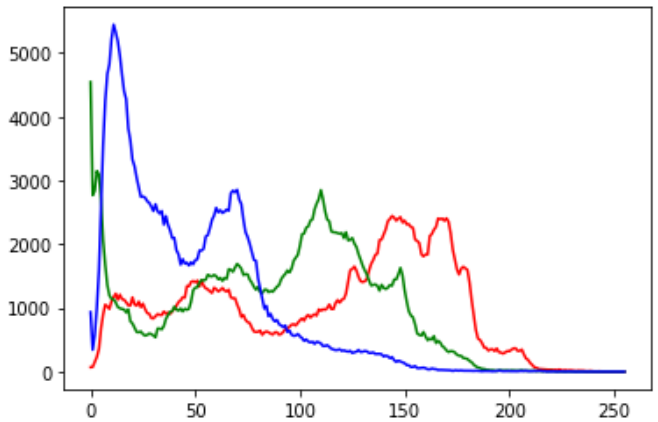
```
# 이미지 채널 분리 (RGB)
r, g, b = cv2.split(img_rgb)

# 컬러 히스토그램 계산 및 그리기
hist = cv2.calcHist([b], [0], None, [256], [0, 256])
plt.plot(hist, color='blue')
plt.show()

plt.show()
```



히스토그램

- 히스토그램 - 예시 (RGB)

```
# 이미지 채널 분리 (RGB)
channels = cv2.split(img_rgb)
colors = ['red', 'green', 'blue']

# 컬러 히스토그램 계산 및 그리기
for (ch, co) in zip(channels, colors):
  hist = cv2.calcHist([ch], [0], None, [256], [0, 256])
  plt.plot(hist, color=co)

plt.show()
```



- zip()
  - 동일한 개수로 이루어진 자료형을 묶어 줌
  - 예시

```
alphabet = ['a', 'b', 'c', 'd', 'e']
number = [1, 2, 3, 4, 5]

for a, n in zip(alphabet, number):
  print(a, n)
```

```
a 1
b 2
c 3
d 4
e 5
```

히스토그램

- 히스토그램 (Histrogram)
  - 스트레칭 (stretching)
    - OpenCV에서는 정규화 (normalize) 라고 부름
    - cv2.normalize(src, dst, alpha, beta, type_flag)
      - src : normalize 이전 데이터
      - dst : normalize 이후 데이터
      - alpha : normalize 구간 1
      - beta : normalize 구간 2
      - type_flag : 알고리즘 선택 옵션
        - cv2.NORM_MINMAX : alpha와 beta 구간으로 normalize
        - cv2.NORM_L1 : 전체 합으로 나누기
        - cv2.NORM_L2 : 단위 벡터로 normalize
        - cv2.NORM_INF : 최대 값으로 나누기

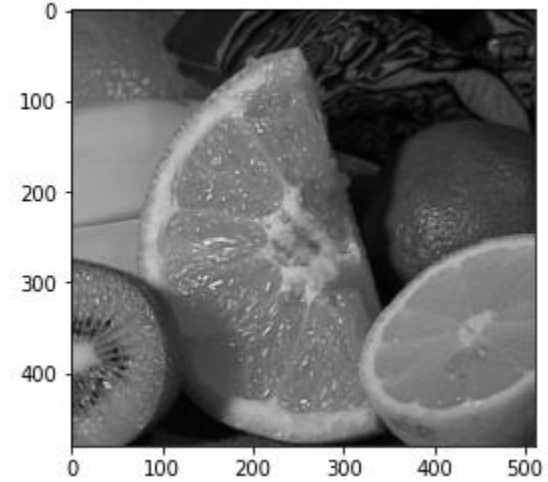
- 히스토그램 (Histrogram)
  - 평활화 (Equalization)
    - cv2.equalizeHist(src)
      - src : 이미지 파일

히스토그램

- 히스토그램 - 예시 (정규화)

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

# fruits 이미지 파일 불러옴
img = cv2.imread('./fruits.bmp')

# BGR에서 GRAY로 변경
img_gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

# 이미지 형태 출력 (height, width, channel)
print('Shape: ', img_gray.shape)

# 이미지 출력
plt.imshow(img_gray, cmap='gray')
plt.show()
```

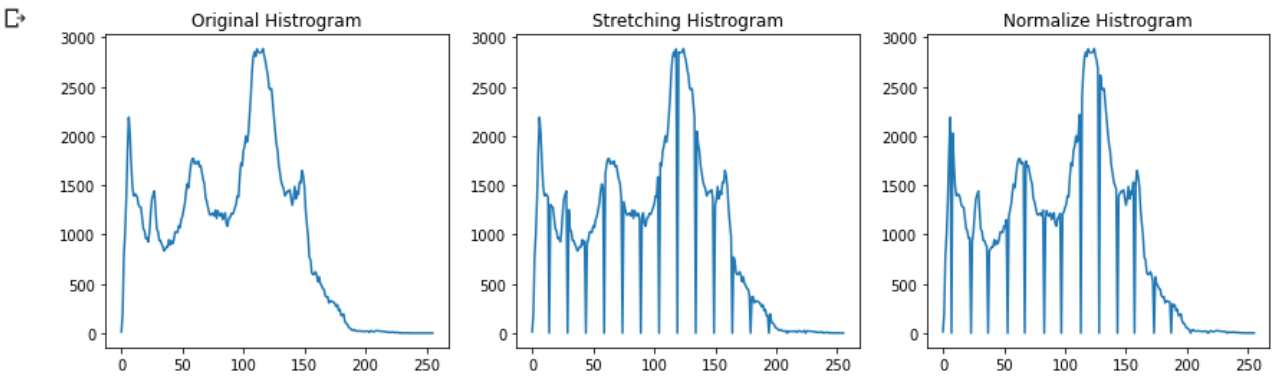
```
Shape:  (480, 512)
```



히스토그램

- 히스토그램 - 예시 (정규화)

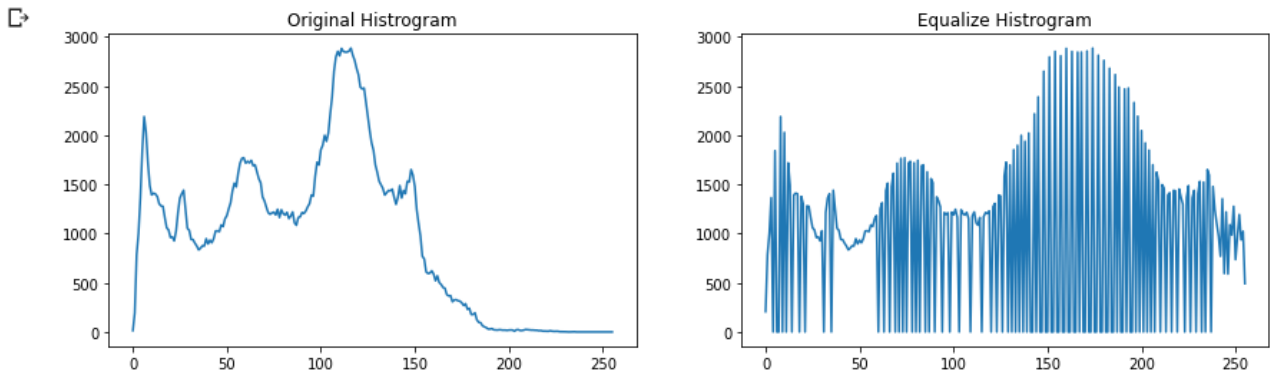
```python
# 직접 연산한 스트레칭
img_f = img_gray.astype(np.float)
img_normalize_1 = ((img_f - img_f.min())) * 255 / (img_f.max() - img_f.min())
img_normalize_1 = img_normalize_1.astype(np.uint8)

# OpenCV normalize 함수를 이용한 정규화
img_normalize_2 = cv2.normalize(img_gray, None, 0, 255, cv2.NORM_MINMAX)

# 히스토그램 계산 및 그리기
hist = cv2.calcHist([img_gray], [0], None, [256], [0, 256])
hist_normalize_1 = cv2.calcHist([img_normalize_1], [0], None, [256], [0, 256])
hist_normalize_2 = cv2.calcHist([img_normalize_2], [0], None, [256], [0, 256])

title = ['Original Histrogram', 'Stretching Histrogram', 'Normalize Histrogram']
histrogram = [hist, hist_normalize_1, hist_normalize_2]

idx = 1
for t, h in zip(title, histrogram):
  plt.subplot(1, 3, idx)
  plt.title(t)
  plt.plot(h)
  idx += 1

plt.subplots_adjust(right=2)
plt.show()
```



히스토그램

- 히스토그램 - 예시 (평활화)

```
# OpenCV equalizeHist 함수를 이용한 평활화
img_equalize = cv2.equalizeHist(img_gray)

# 히스토그램 계산 및 그리기
hist = cv2.calcHist([img_gray], [0], None, [256], [0, 256])
hist_equalize = cv2.calcHist([img_equalize], [0], None, [256], [0, 256])

title = ['Original Histrogram', 'Equalize Histrogram']
histrogram = [hist, hist_equalize]

idx = 1
for t, h in zip(title, histrogram):
  plt.subplot(1, 2, idx)
  plt.title(t)
  plt.plot(h)
  idx += 1

plt.subplots_adjust(right=2)
plt.show()
```



실습

- 실습은 예시를 따라 하는 것으로 대체함

과제

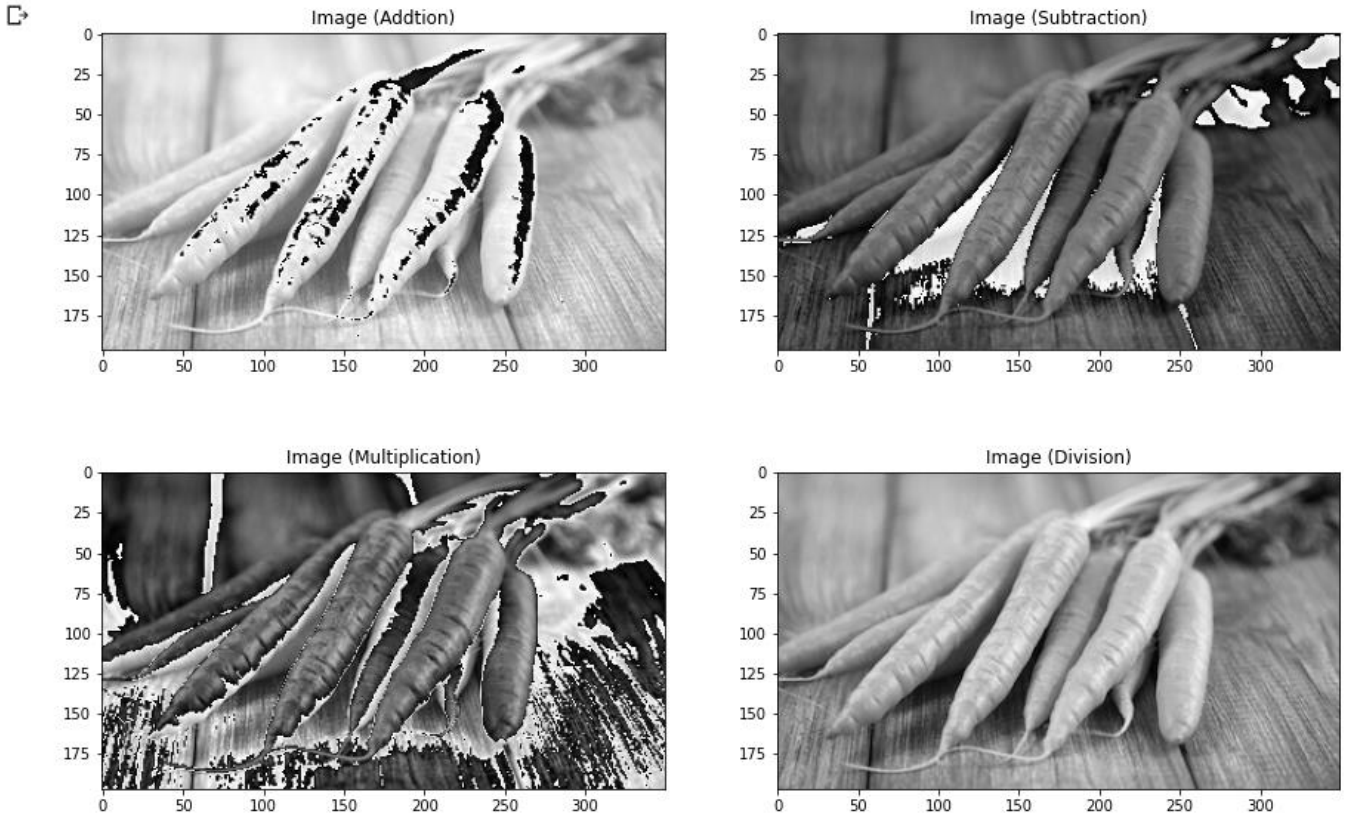
- 소스 코드와 결과를 캡쳐하여 문서(워드, 한글)에 정리한 후, 스마트 캠퍼스에 제출
  - 제출 형식 : 교과목명_주차_학번_이름.docx or .hwp (제출 형식 안 맞을 시 감점)
  - 기간 내에 제출하지 못할 경우 이메일로 제출 (감점)
  - 주석 작성 필수 (주석 없을 시 감점)
  - 부정 행위 적발 시 감점 (컨닝)

과제1

- 이미지 파일의 픽셀 단위로 산술 연산을 수행하시오.
  - 상수 값에 의한 산술 연산과 OpenCV에서 제공하는 산술 연산 함수(cv2.add, …)를 모두 사용
  - 임의의 상수 값 -> 덧셈 : 70, 뺄셈 : 70, 곱셈 : 1.5, 나눗셈 : 1.5
  - 스마트 캠퍼스에서 이미지 파일 (carrot.bmp) 다운로드

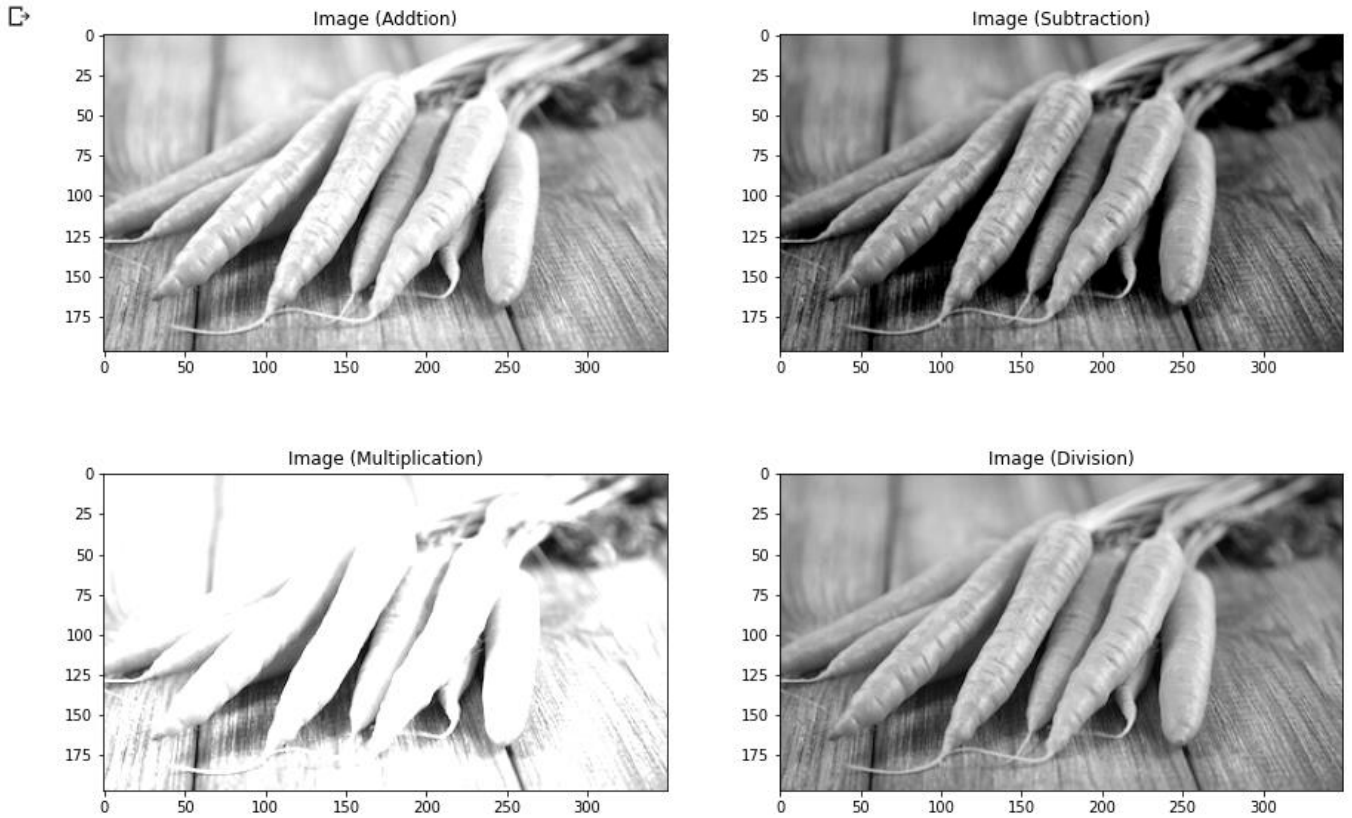
과제1

- 이미지 파일의 픽셀 단위로 산술 연산을 수행하시오. (상수 값에 의한 산술 연산)



과제1

- 이미지 파일의 픽셀 단위로 산술 연산을 수행하시오. (OpenCV 함수를 이용한 산술 연산)



과제2

- 룩업 테이블 연산을 수행하시오.
  - 스마트 캠퍼스에서 이미지 파일 (Lichtenstein.bmp) 다운로드
  - 룩업 테이블 생성 후, 연산 수행 (value : 0.5, 1.5, 2.0)
  - 포스터라이징 테이블 생성 후, 연산 수행 (level : 3, 6)

- 룩업 테이블

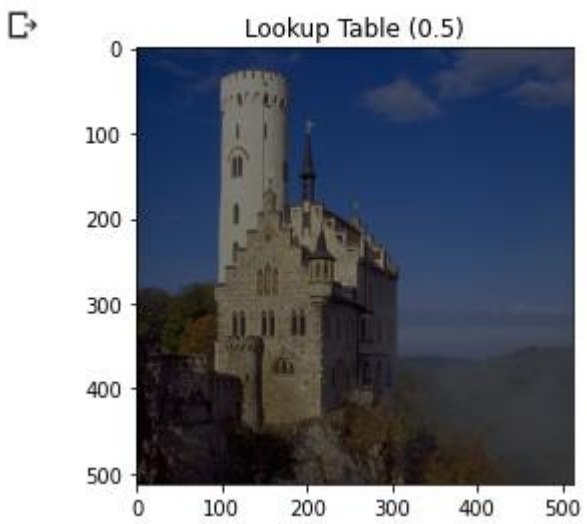
```
table = np.zeros(256).astype(np.int)

for i in range(256):
    temp = int(i * value)
    table[i] = 255 if temp > 255 else temp

OutImg = cv2.LUT(Inimg, table)
```

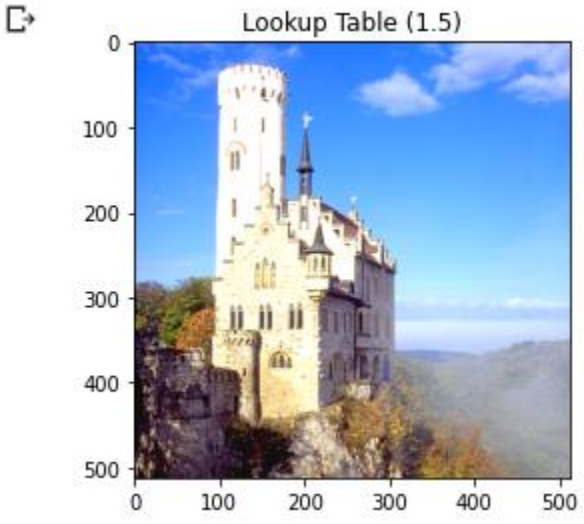
- 포스터 라이징

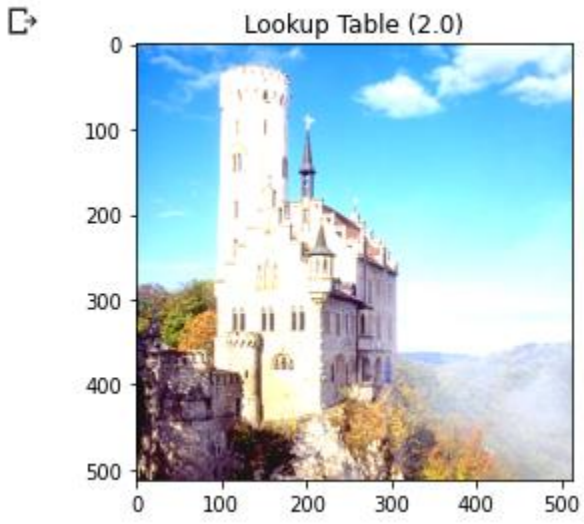
```
table = np.zeros(256).astype(np.int)
step = 256 / level
inc = 255 / (level - 1)

for i in range(256):
    table[i] = int(i / step) * inc

OutImg = cv2.LUT(Inimg, table)
```

과제2

- 룩업 테이블 연산을 수행하시오.

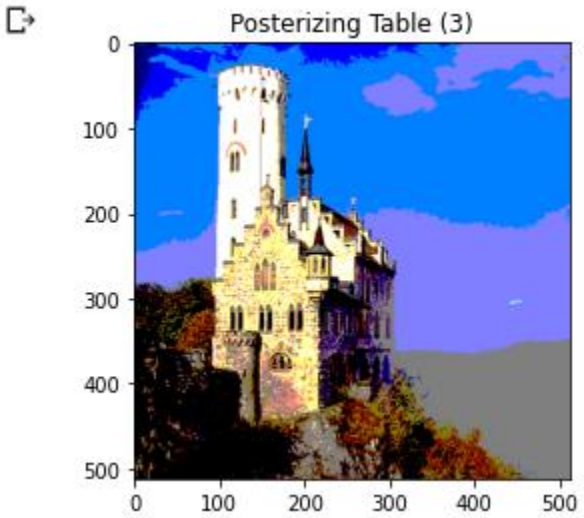


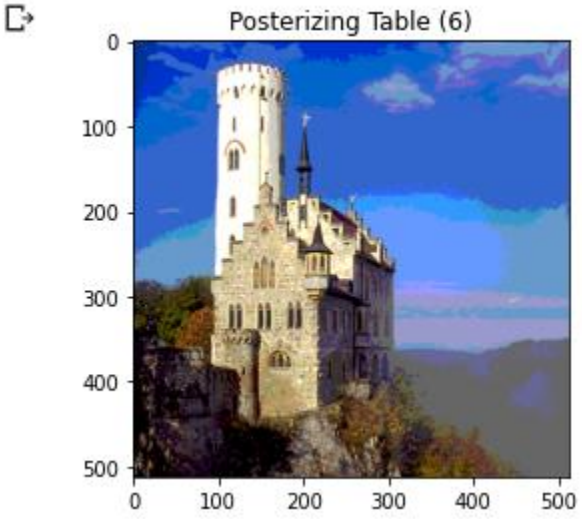




과제2

- 룩업 테이블 연산을 수행하시오.





과제3

- 히스토그램을 그리시오.
  - 이미지 파일 (carrot.bmp) 이용
  - 이미지 파일의 1채널(Gray) 히스토그램과 3채널(RGB) 히스토그램을 그리시오.
  - 이미지 파일의 히스토그램 정규화(normalize)와 히스토그램 평활화(equalizeHist)를 그리시오.

감사합니다