교육부 Ministry of Education

NRF 한국연구재단 National Research Foundation of Korea

영상 화소단위 처리

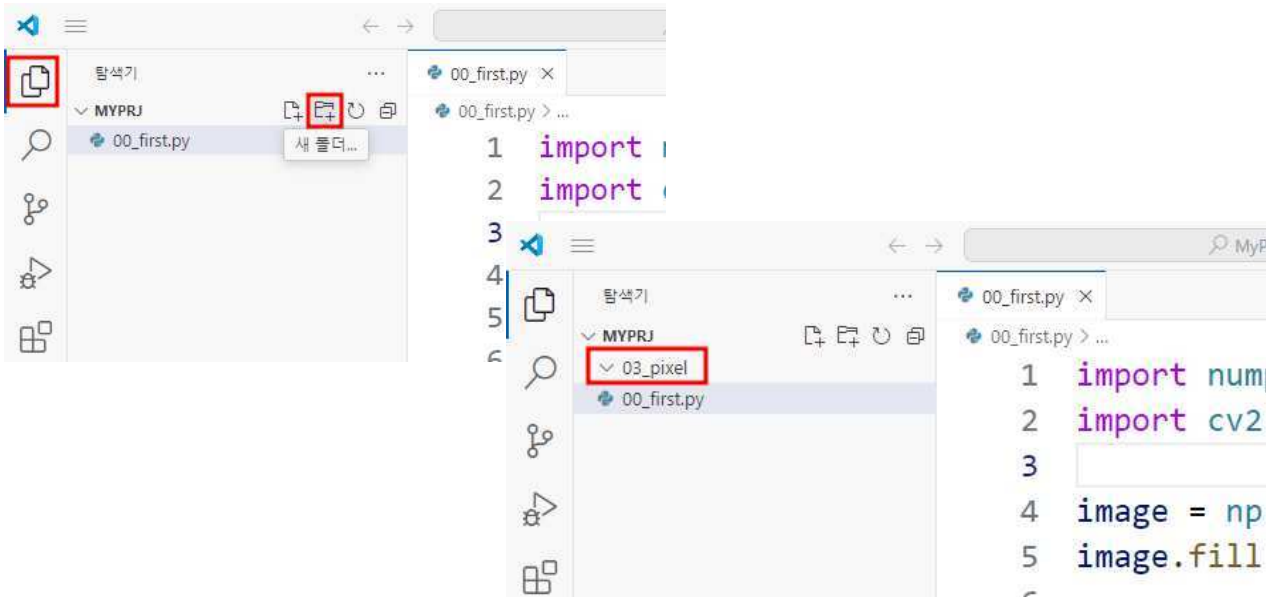
전남대학교 CHONNAM NATIONAL UNIVERSITY | 성균관대학교 SUNGKYUNKWAN UNIVERSITY | KNU 경북대학교 | 전주대학교 JEONJU UNIVERSITY | 서울시립대학교 UNIVERSITY OF SEOUL | 서울과학기술대학교 | 영진전문대학교 YEUNGJIN UNIVERSITY

Contents

- 화소 접근 방법
- 그림파일 읽기/저장하기
- 화소 밝기 변환
- 영상 합성
- 명암 대비

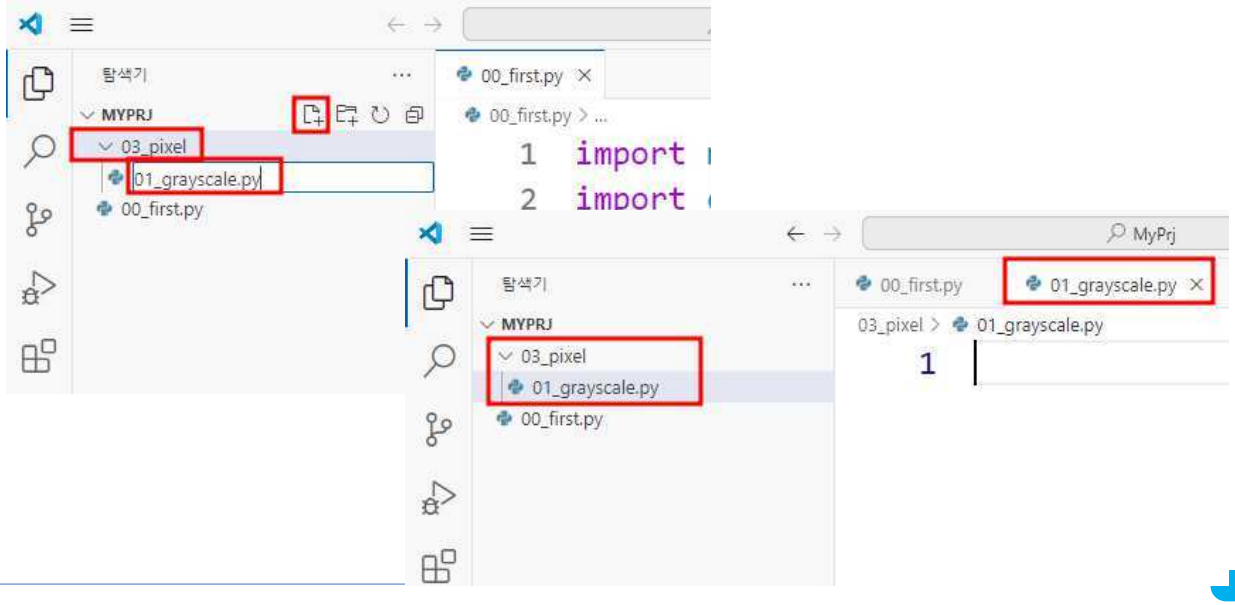
화소 접근 방법

- 화소처리를 위한 새 폴더 추가
  - [탐색기]를 선택 후 [새 폴더] 아이콘을 클릭한 후
  - 03_pixel이라고 폴더 이름을 넣으면 폴더가 새로 생성됨



화소 접근 방법

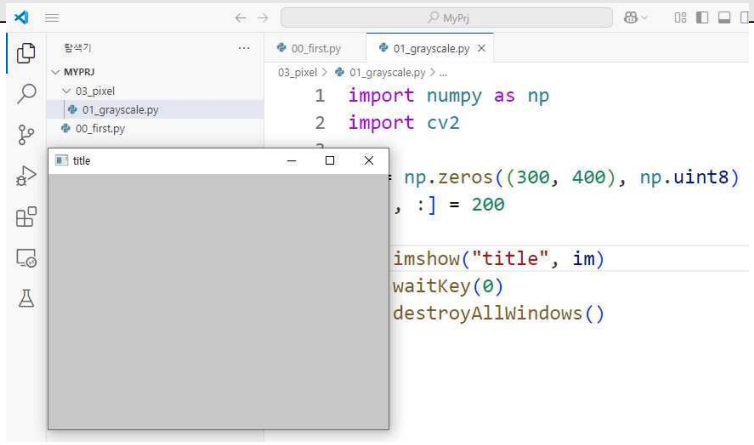
- 03_pixel 폴더에 새 파일 추가
  - [탐색기]를 선택 후 03_pixel 폴더를 선택한 상태에서
  - [새 파일] 아이콘을 클릭한 후
  - 01_grayscale.py이라고 파일 이름을 넣으면 파일이 새로 생성됨



화소 접근 방법

- 회색음영 이미지 생성 : 01_grayscale.py

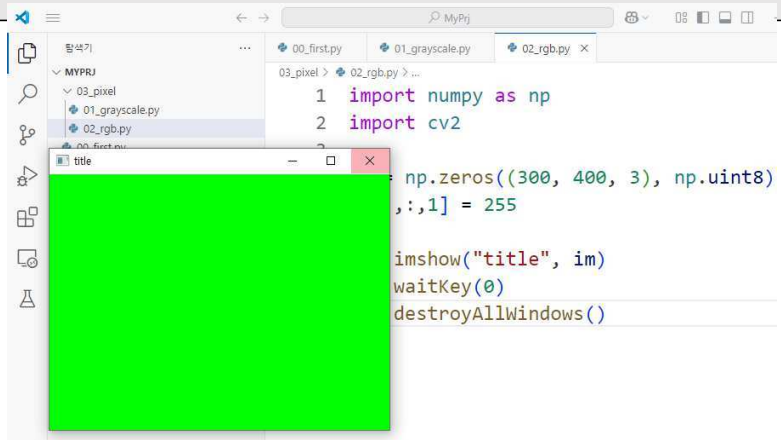
```
import numpy as np
import cv2

im = np.zeros((300, 400), np.uint8)
im[:, :] = 200

cv2.imshow("title", im)
cv2.waitKey(0)
cv2.destroyAllWindows()
```



화소 접근 방법

- RGB 컬러 이미지 생성 : 02_rgb.py

```
import numpy as np
import cv2

im = np.zeros((300, 400, 3), np.uint8)
im[:,:,1] = 255

cv2.imshow("title", im)
cv2.waitKey(0)
cv2.destroyAllWindows()
```



화소 접근 방법

- 01_grayscale.py 수정

```
import numpy as np
Import cv2

im = np.zeros((300, 400), np.uint8)
im[:, :] = 200

#1)

cv2.imshow("title", im)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

- (50, 50) 위치 화소를 0으로 채움 : #1) 위치 아래에 코드 추가
  - 직접 접근
    - im[50, 50] = 0
  - item() 및 itemset() 사용
    - im.itemset((50, 50), 0)

그림 파일 읽기/저장하기

- cv2.imread(filename[, flags])
  - 영상 파일로부터 영상을 읽어 numpy.array로 반환
  - filename : 저장된 영상파일명
  - flags : 읽어온 영상을 numpy.array로 저장할 때 컬러 타입 결정 상수

flags	값	설명
cv2.IMREAD_UNCHANGED	-1	입력 파일에 정의된 타입의 영상을 그대로 반환
cv2.IMREAD_GRAYSCLE	0	그레이스케일 영상으로 변환하여 반환
cv2.IMREAD_COLOR	1	컬러영상으로 변환하여 반환
cv2.IMREAD_ANYDEPTH	2	입력 파일에 정의된 깊이에 따라 16비트/32비트로 변환하여 반환, 기본은 8비트 영상으로 반환
cv2.IMREAD_ANYCOLOR	4	입력파일에 정의된 타입의 영상으로 반환

그림 파일 읽기/저장하기

- 그림파일 읽기 : 03_read.py

```
import numpy as np
import cv2

im1 = cv2.imread('./lena.jpg', cv2.IMREAD_UNCHANGED)
im2 = cv2.imread('./lena.jpg', cv2.IMREAD_GRAYSCALE)
im3 = cv2.imread('./lena.jpg', cv2.IMREAD_COLOR)

print(im1.shape, im2.shape, im3.shape)
cv2.imshow("title1", im1)
cv2.imshow("title2", im2)
cv2.imshow("title3", im3)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

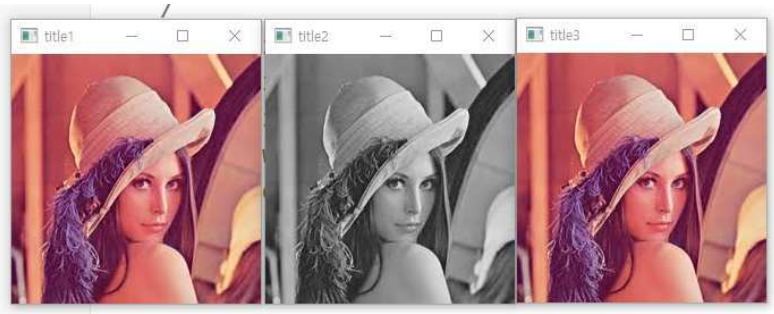
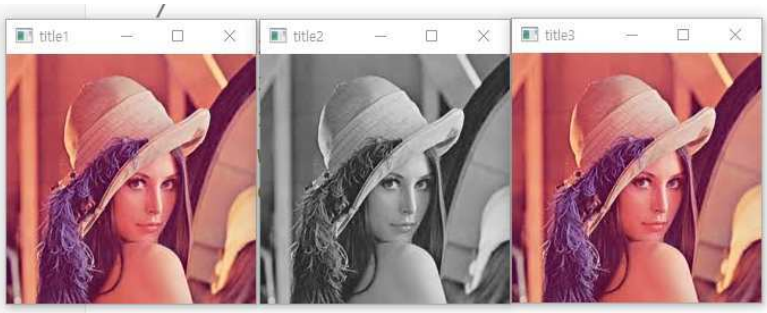


그림 파일 읽기/저장하기

- 그림파일 읽기 : 03_read.py



  - im1 : 색상 변환 없이 반환
    - shape : (220, 220, 3)
  - im2 : 컬러영상을 그레이스케일로 반환
    - shape : (220, 220)
  - im3 : 컬러영상을 컬러로 반환
    - shape : (220, 220, 3)

그림 파일 읽기/저장하기

- 그림 파일 읽기 : 03_read.py 수정

```
import numpy as np
import cv2

im1 = cv2.imread('./lena_g.jpg', cv2.IMREAD_UNCHANGED)
im2 = cv2.imread('./lena_g.jpg', cv2.IMREAD_GRAYSCALE)
im3 = cv2.imread('./lena_g.jpg', cv2.IMREAD_COLOR)

print(im1.shape, im2.shape, im3.shape)
cv2.imshow("title1", im1)
cv2.imshow("title2", im2)
cv2.imshow("title3", im3)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

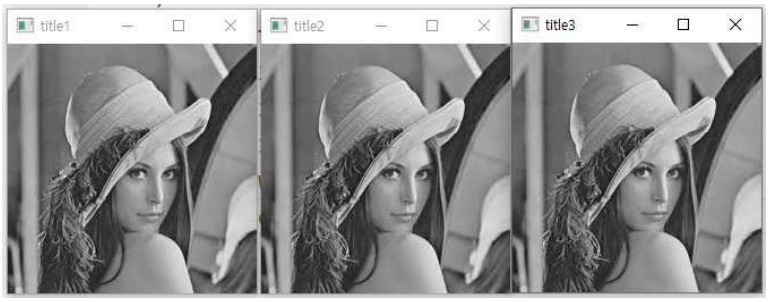
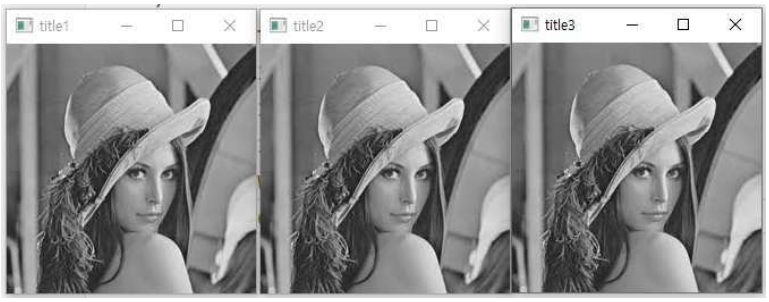


그림 파일 읽기/저장하기

- 그림 파일 읽기 : 03_read.py 수정



  - im1 : 색상 변환 없이 반환
    - shape : (220, 220)
  - im2 : 그레이영상을 그레이스케일로 반환
    - shape : (220, 220)
  - im3 : 그레이영상을 컬러로 반환
    - shape : (220, 220, 3)
    - 본래 그레이 영상이라 컬러로 보이지는 않음

그림 파일 읽기/저장하기

- 03_read.py 수정

```
import numpy as np
import cv2

im1 = cv2.imread('./lena_g.jpg', cv2.IMREAD_UNCHANGED)
im2 = cv2.imread('./lena_g.jpg', cv2.IMREAD_GRAYSCALE)
im3 = cv2.imread('./lena_.jpg', cv2.IMREAD_COLOR)

if im1 is None or im2 is None or im3 is None:
    raise Exception('파일 읽기 오류')

print(im1.shape, im2.shape, im3.shape)
cv2.imshow("title1", im1)
cv2.imshow("title2", im2)
cv2.imshow("title3", im3)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

  - lena_.jpg 파일이 존재하지 않아 에러가 발생
  - 예외처리를 지정해서 에러가 발생하지 않도록 코드 추가

그림 파일 읽기/저장하기

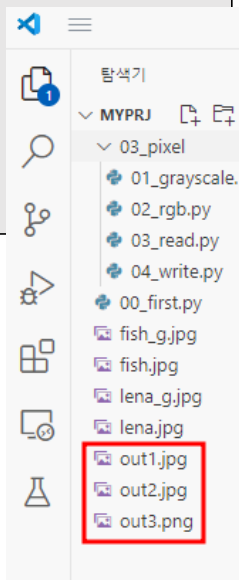
- cv2.imwrite(filename, img[, params])
  - 행렬(영상)을 파일로 저장
  - filename의 확장자에 따라서 JPG, BMP, PNG, TIF, PPM 등의 포맷으로 저장
  - filename : 저장할 영상 파일명
  - img : 저장하고자 하는 행렬(영상)
  - params : 압축 방식에 사용되는 인수 쌍

paramid	값 (기본값)	설명
cv2.IMWRITE_JPEG_QUALITY	0-100(95)	JPG 파일 화질, 높은 값일수록 화질이 좋음
cv2.IMWRITE_PNG_COMPRESSION	0-9(3)	PNG 파일 압축 레벨, 높은 값일수록 용량은 적어지고, 압축 시간이 길어짐
cv2.IMWRITE_PXM_BINARY	0 or 1(1)	PPM, PGM 파일의 이진 포맷 설정

그림 파일 읽기/저장하기

- 그림파일 쓰기 : 04_write.py

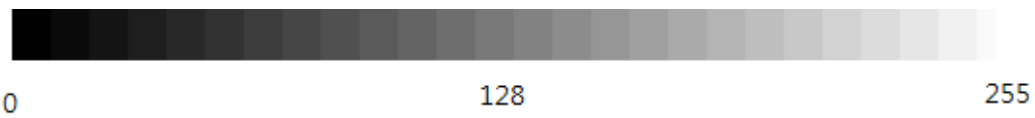
```python
import numpy as np
import cv2

im = cv2.imread('./lena.jpg', cv2.IMREAD_COLOR)

cv2.imwrite('./out1.jpg', im)
cv2.imwrite('./out2.jpg', im, [cv2.IMWRITE_JPEG_QUALITY, 10])
cv2.imwrite('./out3.png', im, [cv2.IMWRITE_PNG_COMPRESSION, 9])

cv2.imshow("title", im)
cv2.waitKey(0)
cv2.destroyAllWindows()
```



- out1.jpg : jpg 포맷으로 화질 95로 저장
- out2.jpg : jpg 포맷으로 화질 10로 저장되어 화질이 나쁨
- out3.png : png 포맷으로 압축률이 가장 높게 저장

화소 밝기 변환

- 그레이 스케일 영상
  - 컬러를 표현하지 못하고 영상의 명암을 표현함
  - 화소를 1바이트(8bits)로 표현
  - 화소는 0~255 사이의 값을 가짐
  - 0은 검은색, 255는 흰색을 의미하며
  - 0과 255사이의 값들은 진한 회색에서 연한 회색으로 표현됨



화소 밝기 변환

- 화소 밝기를 변경하는 방식
  - saturation 방식(OpenCV 함수 사용)
    - im2 = cv2.add(im, 100)
    - im2 = cv2.subtract(im, 100)
    - 연산결과가 255보다 크게 되면 255로 할당, 0보다 작게 되면 0으로 할당
  - modulo 방식(numpy array에 직접 연산)
    - im2 = im + 100
    - im2 = im - 100
    - 연산결과가 0보다 작거나 225보다 크게 되면 엉뚱한 값으로 할당됨

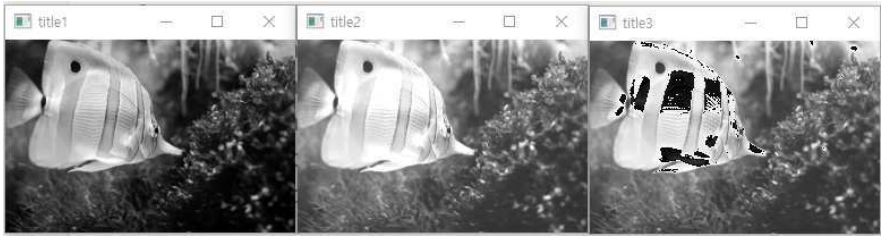
화소 밝기 변환

- 밝기 변환 : 05_bright.py

```
import numpy as np
import cv2

im1 = cv2.imread('./fish.jpg', cv2.IMREAD_GRAYSCALE)
if im1 is None: raise Exception('파일 읽기 오류')

im2 = cv2.add(im1, 50)
im3 = im1 + 50

cv2.imshow("title1", im1)
cv2.imshow("title2", im2)
cv2.imshow("title3", im3)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

화소 밝기 변환

- 밝기 변환 : 05_bright.py



  - im2는 50을 더한 화소값이 255를 넘으면 255로 할당
    - 흰색이 많아짐
  - im3는 50을 더한 화소값이 255를 넘으면 합에서 255를 뺀 값으로 할당
    - 밝은 부분이 어두워짐

화소 밝기 변환

- 밝기 변환 : 05_bright.py

```
import numpy as np
import cv2

im1 = cv2.imread('./fish.jpg', cv2.IMREAD_GRAYSCALE)
if im1 is None: raise Exception('파일 읽기 오류')

im2 = cv2.subtract(im1, 50)
im3 = im1 - 50

cv2.imshow("title1", im1)
cv2.imshow("title2", im2)
cv2.imshow("title3", im3)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

화소 밝기 변환

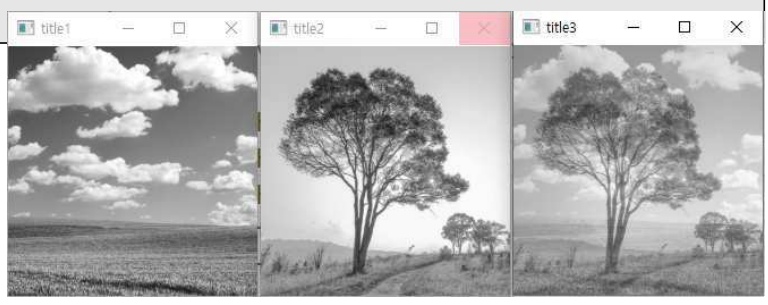
- 밝기 변환 : 05_bright.py

  - im2는 50을 뺀 화소값이 0보다 작으면 0으로 할당
    - 검정색이 많아짐
  - im3는 50을 뺀 화소값이 0보다 작으면 차에 255를 더한 값으로 할당
    - 어두웠던 부분이 밝아짐

영상합성

- OpenCV 함수를 통한 합성 : 06_merge1.py

```
import numpy as np
import cv2

im1 = cv2.imread('./ground.png', cv2.IMREAD_GRAYSCALE)
im2 = cv2.imread('./tree2.png', cv2.IMREAD_GRAYSCALE)

im3 = cv2.addWeighted(im1, 0.4, im2, 0.6, 0)

cv2.imshow("title1", im1)
cv2.imshow("title2", im2)
cv2.imshow("title3", im3)
cv2.waitKey(0)
cv2.destroyAllWindows()
```



영상합성

- 행렬 연산을 통한 합성 : 06_merge2.py

```
import numpy as np
import cv2

im1 = cv2.imread('./ground.png', cv2.IMREAD_GRAYSCALE)
im2 = cv2.imread('./tree2.png', cv2.IMREAD_GRAYSCALE)

im3 = cv2.add(im1 * 0.4 + im2 * 0.6)
im3 = np.clip(im3, 0, 255).astype('uint8')

cv2.imshow("title1", im1)
cv2.imshow("title2", im2)
cv2.imshow("title3", im3)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

영상합성

- 행렬 연산을 통한 합성 : 06_merge2.py

  - cv2.add()에서 0.4, 0.6의 실수값이 있어 float로 반환
  - np.clip()으로 0~255사이값으로 할당하고, uint8로 반환

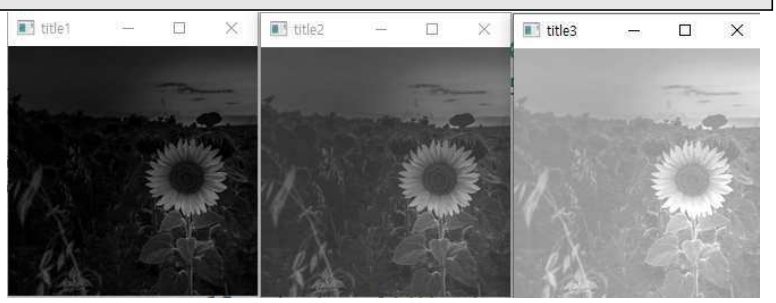
명암대비

- 명암 대비
  - 대비(contrast) : 이미지의 밝은 부분과 어두운 부분 사이의 차이
  - 픽셀 값 직접 조정:
    - dst = cv2.convertScaleAbs(image, alpha=alpha, beta=beta)
    - alpha는 대비를 조절하는 값
  - cv2.addWeighted() 함수 사용:
    - dst = cv2.addWeighted(src, alpha, src, 0, beta)
    - alpha는 대비를, beta는 밝기를 조절
  - cv2.scaleAdd() 함수 사용:
    - dst = cv2. scaleAdd(src, alpha, src)
    - alpha는 대비를 조절
  - 히스토그램 평활화:
    - equalized_image = cv2.equalizeHist(gray_image)
    - 이미지의 intensity 값을 재분배하여 대비를 향상
  - 적응형 히스토그램 평활화(CLAHE)
    - 이미지를 작은 블록으로 나누어 각 블록별로 히스토그램 평활화를 적용
    - 전체적인 대비 향상과 함께 지역적 특성 보존

명암대비

- 명암 증가 : 07_contrast1.py

```
import numpy as np
import cv2

im1 = cv2.imread('./low.png', cv2.IMREAD_GRAYSCALE)

im2 = cv2.add(im1, 50)
im3 = cv2.add(im1, 150)

cv2.imshow("title1", im1)
cv2.imshow("title2", im2)
cv2.imshow("title3", im3)
cv2.waitKey(0)
cv2.destroyAllWindows()
```



명암대비

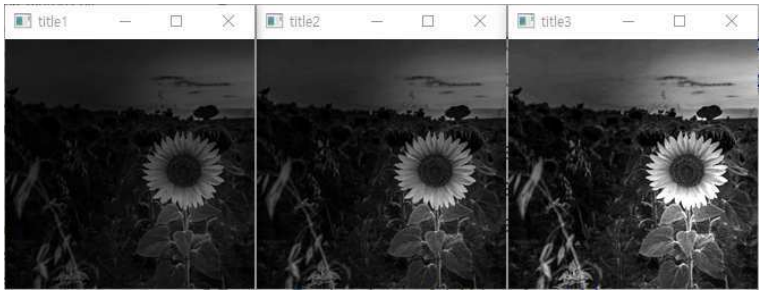
- cv2.convertScaleAbs() 함수
  - 이미지의 명암대비(contrast)와 밝기(brightness)를 조절하는 데 사용
  - dst = cv2.convertScaleAbs(src, alpha=alpha, beta=beta)
  - 매개변수:
    - src: 입력 이미지 (numpy 배열)
    - alpha: 대비를 조절하는 스케일 팩터 (기본값 1)
    - beta: 밝기를 조절하는 값 (기본값 0)
  - 동작 원리:
    - 각 픽셀에 대해 다음 연산을 수행
    - dst(x,y) = saturate(|src(x,y) * alpha + beta|)
    - saturate 함수는 결과값을 [0, 255] 범위로 제한

명암대비

- cv2.convertScaleAbs() 함수 : 07_constrast2.py

```
import numpy as np
import cv2

im1 = cv2.imread('./low.png', cv2.IMREAD_GRAYSCALE)

im2 = cv2.convertScaleAbs(im1, alpha=1.5, beta=0)
im3 = cv2.convertScaleAbs(im1, alpha=2.5, beta=0)

cv2.imshow("title1", im1)
cv2.imshow("title2", im2)
cv2.imshow("title3", im3)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

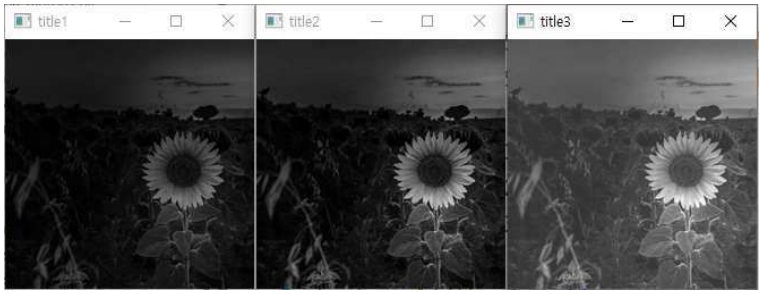
- 대비 1.5배 증가
- 대비 2.5배 증가



명암대비

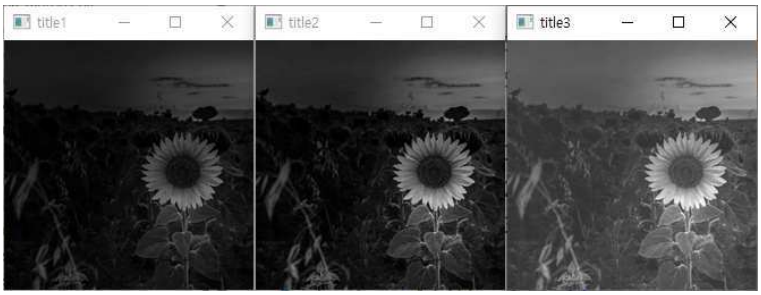
- cv2.convertScaleAbs() 함수 : 07_contrast2.py

```
import numpy as np
import cv2

im1 = cv2.imread('./low.png', cv2.IMREAD_GRAYSCALE)

im2 = cv2.convertScaleAbs(im1, alpha=1.5, beta=0)
im3 = cv2.convertScaleAbs(im1, alpha=1.5, beta=50)

cv2.imshow("title1", im1)
cv2.imshow("title2", im2)
cv2.imshow("title3", im3)
cv2.waitKey(0)
cv2.destroyAllWindows()
```



명암대비

- cv2.convertScaleAbs() 함수 : 07_contrast2.py



  - im2 = cv2.convertScaleAbs(im1, alpha=1.5, beta=0)
    - 대비 1.5배 증가
  - im3 = cv2.convertScaleAbs(im1, alpha=1.5, beta=50)
    - 대비 1.5배 증가 + 밝기 50 증가

명암대비

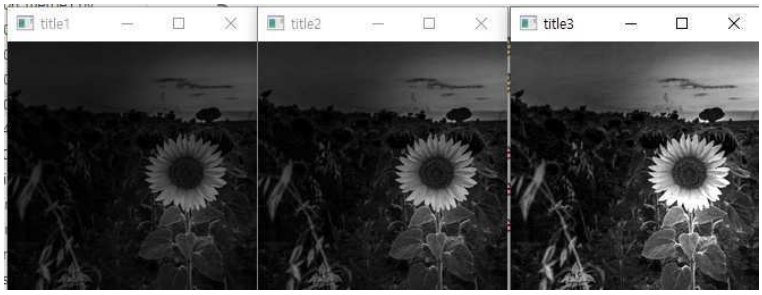
- cv2.addWeighted() 함수
  - 두 이미지를 가중치를 적용해 합성(블렌딩)하는 데 사용
  - dst = cv2.addWeighted(src1, alpha, src2, beta, gamma)
  - 매개변수:
    - src1, src2: 합성할 두 입력 이미지 (같은 크기여야 함)
    - alpha: 첫 번째 이미지(src1)에 적용할 가중치
    - beta: 두 번째 이미지(src2)에 적용할 가중치
    - gamma: 결과 이미지에 더할 스칼라 값 (일반적으로 0)
  - 계산 : $dst(I)=saturate(src1(I)\cdot alpha + src2(I)\cdot beta + gamma)$
  - 특징
    - 두 이미지의 각 픽셀을 가중치에 따라 합성하여 자연스럽게 하나의 이미지로 결합
    - alpha와 beta의 합이 1이 되도록 설정하는 것이 일반적
    - 이미지 간 전환 효과나 투명도 조절에 사용

명암대비

- cv2. addWeighted() 함수 : 07_contrast3.py

```
import numpy as np
import cv2

im1 = cv2.imread('./low.png', cv2.IMREAD_GRAYSCALE)

im2 = cv2.addWeighted(im1, 1.5, im1, 0, 0)
im3 = cv2.addWeighted(im1, 2.5, im1, 0, 0)

cv2.imshow("title1", im1)
cv2.imshow("title2", im2)
cv2.imshow("title3", im3)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

- 대비 1.5배 증가
- 대비 2.5배 증가



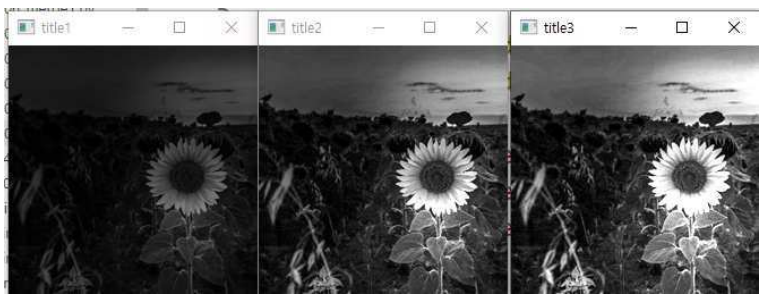
명암대비

- cv2.ScaleAdd() 함수
  - 두 이미지 중 하나의 이미지에 가중치를 적용해 합성(블렌딩)
  - dst = cv2.scaleAdd(src1, scale, src2)
  - 매개변수:
    - src1: 첫 번째 입력 이미지
    - scale: src1에 적용할 스케일 값
    - src2: 두 번째 입력 이미지
    - dst: 결과 이미지
  - 계산: dst(I) = scale * src1(I) + src2(I)
  - 용도:
    - 이미지의 대비(contrast)를 조절하는 데 사용

명암대비

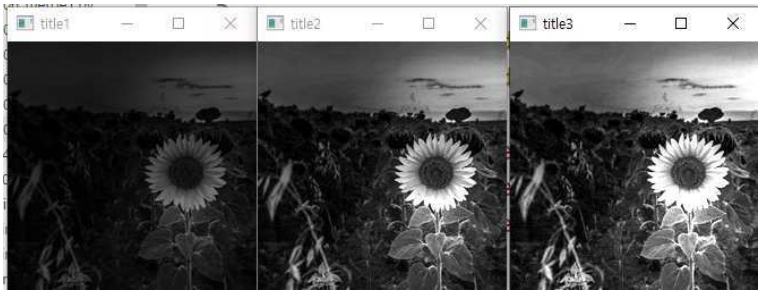
- cv2. ScaleAdd() 함수 : 07_contrast4.py

```
import numpy as np
import cv2

im1 = cv2.imread('./low.png', cv2.IMREAD_GRAYSCALE)

im2 = cv2.scaleAdd(im1, 1.5, im1)
im3 = cv2.scaleAdd(im1, 2.5, im1)

cv2.imshow("title1", im1)
cv2.imshow("title2", im2)
cv2.imshow("title3", im3)
cv2.waitKey(0)
cv2.destroyAllWindows()
```



명암대비

- cv2. ScaleAdd() 함수 : 07_contrast4.py



  - im2 = cv2.scaleAdd(im1, 1.5, im1)
    - 대비 1.5배 증가
  - im3 = cv2.scaleAdd(im1, 2.5, im1)
    - 대비 2.5배 증가

명암대비

- 히스토그램 평활화(Histogram Equalization)
  - 이미지의 명암 대비를 향상시키는 기법
  - 이미지의 밝기 분포를 균등하게 재분배하여 대비를 개선
  - 동작 원리
    - 이미지의 히스토그램을 계산
    - 누적 분포 함수(CDF)를 계산
    - CDF를 기반으로 픽셀 값을 재매핑
  - 효과
    - 전반적인 명암 대비가 높아짐
    - 어두운 영역의 디테일이 개선
    - 특정 밝기 범위에 치우친 픽셀 값들을 넓은 범위로 분포시킴
  - 장점
    - 클리핑 기법과 달리, 효과가 더 넓은 화소값 범위에 걸쳐 고르게 적용
    - 특정 화소값 범위의 형체 정보를 유지하면서 전반적인 명암 대비를 향상
  - 적용 상황
    - 이미지가 너무 어둡거나 밝아 디테일이 손실된 경우에 유용
    - 안개 낀 사진과 같이 명암값이 특정 영역에 몰려있는 경우에 효과적

명암대비

- 히스토그램 평활화 직접해보기

3	2	4	2	3
2	4	3	2	3
4	5	5	3	5
4	2	4	3	4
4	2	5	3	3

화소값	0	1	2	3	4	5	6	7
빈도수	0	0	6	8	7	4	0	0
누적빈도수	0	0	6	14	21	25	25	25
정규화누적합	0.00	0.00	0.24	0.56	0.84	1.00	1.00	1.00
평활화결과	0	0	2	4	6	7	7	7

4	2	6	2	4
2	6	4	2	4
6	7	7	4	7
6	2	6	4	6
6	2	7	4	4

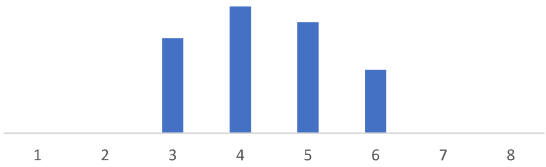
화소값	0	1	2	3	4	5	6	7
빈도수	0	0	6	0	8	0	7	4

명암대비

- 히스토그램 평활화 직접해보기

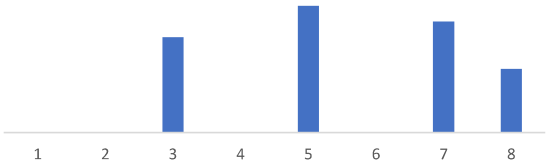
3	2	4	2	3
2	4	3	2	3
4	5	5	3	5
4	2	4	3	4
4	2	5	3	3

화소값	0	1	2	3	4	5	6	7
빈도수	0	0	6	8	7	4	0	0



4	2	6	2	4
2	6	4	2	4
6	7	7	4	7
6	2	6	4	6
6	2	7	4	4

화소값	0	1	2	3	4	5	6	7
빈도수	0	0	6	0	8	0	7	4



명암대비

- cv2.equalizeHist() 함수
  - 히스토그램 평활화를 수행하는 함수
  - dst = cv2.equalizeHist(src)
  - 매개변수
    - src: 입력 영상 (8비트 단일 채널 이미지)
    - dst: 히스토그램 평활화가 적용된 결과 영상
  - 동작 원리
    - 입력 영상의 히스토그램을 계산
    - 히스토그램의 누적 분포를 계산
    - 누적 분포를 정규화하여 픽셀 값을 재매핑
  - 효과
    - 영상의 명암 대비를 향상
    - 어두운 영역의 디테일을 개선

명암대비

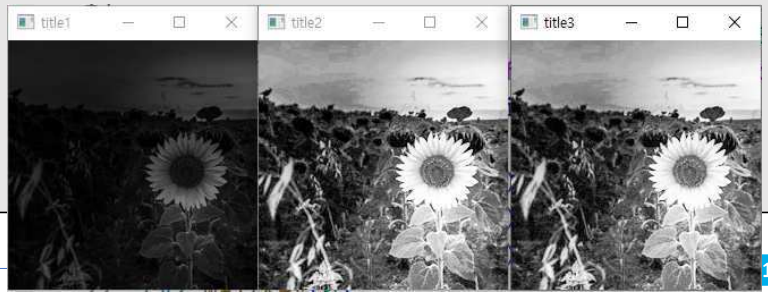
- cv2.equalizeHist() 함수 : 07_contrast5_hist1.py

```
import numpy as np
import cv2

im1 = cv2.imread('./low.png', cv2.IMREAD_GRAYSCALE)

im2 = cv2.equalizeHist(im1)

cv2.imshow("title1", im1)
cv2.imshow("title2", im2)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

명암대비

- numpy 이용 직접 코드 구현 : 07_contrast5_hist2.py

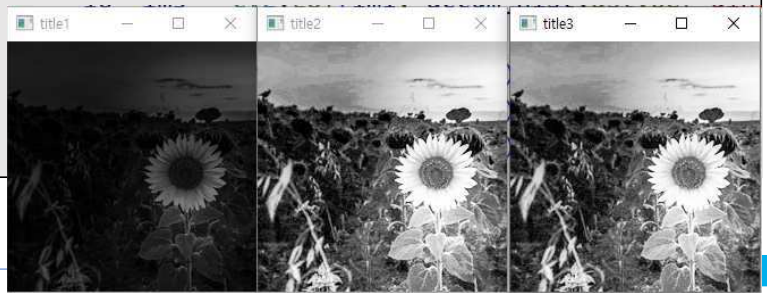
```
import numpy as np
import cv2

im1 = cv2.imread('./low.png', cv2.IMREAD_GRAYSCALE)
im2 = cv2.equalizeHist(im1)

bins, ranges = [256], [0, 256]
hist = cv2.calcHist([im1], [0], None, bins, ranges)
accum_hist = np.zeros(hist.shape[:2], np.float32)
accum_hist[0] = hist[0]
for i in range(1, hist.shape[0]):
    accum_hist[i] = accum_hist[i - 1] + hist[i]

accum_hist = (accum_hist / sum(hist)) * 255
im3 = [[accum_hist[val] for val in row] for row in im1]
im3 = np.array(im3, np.uint8)

cv2.imshow("title1", im1)
cv2.imshow("title2", im2)
cv2.imshow("title3", im3)
cv2.waitKey(0)
cv2.destroyAllWindows()
```



명암대비

- cv2.LUT() 이용 직접 코드 구현 : 07_contrast5_hist3.py

```
import numpy as np
import cv2

im1 = cv2.imread('./low.png', cv2.IMREAD_GRAYSCALE)
im2 = cv2.equalizeHist(im1)

bins, ranges = [256], [0, 256]
hist = cv2.calcHist([im1], [0], None, bins, ranges)
accum_hist = np.zeros(hist.shape[:2], np.float32)
accum_hist[0] = hist[0]
for i in range(1, hist.shape[0]):
    accum_hist[i] = accum_hist[i - 1] + hist[i]

accum_hist = (accum_hist / sum(hist)) * 255
im3 = cv2.LUT(im1, accum_hist.astype("uint8"))

cv2.imshow("title1", im1)
cv2.imshow("title2", im2)
cv2.imshow("title3", im3)
cv2.waitKey(0)
cv2.destroyAllWindows()
```



교육부 Ministry of Education

NRF 한국연구재단 National Research Foundation of Korea

thank you

본 과제(결과물)는 교육부와 한국연구재단의 재원으로 지원을 받아 수행된
디지털신기술인재양성 혁신공유대학사업의 연구결과입니다.

COSS 전남대학교 CHONNAM NATIONAL UNIVERSITY 성균관대학교 SUNGKYUNKWAN UNIVERSITY KNU 경북대학교 전주대학교 JEONJU UNIVERSITY 서울시립대학교 UNIVERSITY OF SEOUL 서울과학기술대학교 영진전문대학교 YEUNGJIN UNIVERSITY