

## 4주차 - 화소 점처리(2)

- 감마 보정(Gamma correction)
  - 감마 보정 함수를 이용하여 화소 값을 처리
- 포스터라이징(Posterizing)
  - 디지털 영상의 화소 값을 주어진 경계 값으로 그룹화하여 화소의 소를 감소시키는 처리 방법
- 이진화(Binarization)
  - 임계 값(threshold)보다 큰 화소는 255 그렇지 않은 화소는 0으로 변환

```
In [1]: ### Packages
import cv2
import numpy as np
from matplotlib import pyplot as plt
import os
```

```
In [2]: ### 출력 영상 크기
plt.rcParams["figure.figsize"] = (16,9)
### 한글 표시
plt.rcParams['font.family'] = "Gulim" # 'AppleGothic' in mac
```

### ▶ 감마 보정(Gamma correction)

- 모든 영역에서 유사한 밝기 변화를 느낄 수 있도록 선형화할 수 있는 잘 알려진 방법 중 하나
- CRT (Cathode Ray Tube) 디스플레이 장치
  - 입력되는 영상은 선형적으로 밝기가 변하는 영상
  - CRT 모니터가 입력되는 영상을 빛으로 재현하는 구동 방식의 특성이 비선형적
  - 입력 영상에 대하여 CRT 모니터는 어두운 영역의 밝기 변화가 적도록, 밝은 영역의 밝기 변화가 상대적으로 급격하게 증가하도록 함




- 감마보정 함수
  - 입력 신호  $r$ , 감마보정 값  $\gamma$ , 출력 신호  $s$

$$s = (r)^\gamma$$

- 비트심도 8인 영상에 대한 감마보정 함수의 적용
  - 감마보정 함수를 이용한 비선형 변환(non-linear transformation)

$$s = 255 \times \left( \frac{r}{255} \right)^\gamma$$

- 다양한 감마보정 값에 대한 감마보정 함수 

- 감마보정 전의 CRT 디스플레이를 통한 영상 출력 예

- CRT 디스플레이 특성이 감마 변형 값 2.5에 해당함을 가정 → 감마보정 값 0.4를 적용



```
In [3]: ### 감마보정 값 0.4를 적용
gamma = 0.4
lookUpTable = np.empty((1, 256), np.uint8)
for i in range(256) :
    lookUpTable[0, i] = np.clip(pow (i / 255.0, gamma) * 255.0, 0, 255)
lookUpTable
```

```
Out[3]: array([[ 0, 27, 36, 43, 48, 52, 56, 60, 63, 66, 69, 72, 75,
77, 79, 82, 84, 86, 88, 90, 92, 93, 95, 97, 99, 100,
102, 103, 105, 106, 108, 109, 111, 112, 113, 115, 116, 117, 119,
120, 121, 122, 123, 125, 126, 127, 128, 129, 130, 131, 132, 133,
134, 136, 137, 138, 139, 140, 141, 141, 142, 143, 144, 145, 146,
147, 148, 149, 150, 151, 152, 152, 153, 154, 155, 156, 157, 157,
158, 159, 160, 161, 161, 162, 163, 164, 165, 165, 166, 167, 168,
168, 169, 170, 171, 171, 172, 173, 173, 174, 175, 176, 176, 177,
178, 178, 179, 180, 180, 181, 182, 182, 183, 184, 184, 185, 186,
186, 187, 187, 188, 189, 189, 190, 191, 191, 192, 192, 193, 194,
194, 195, 195, 196, 197, 197, 198, 198, 199, 200, 200, 201, 201,
202, 202, 203, 204, 204, 205, 205, 206, 206, 207, 207, 208, 208,
209, 210, 210, 211, 211, 212, 212, 213, 213, 214, 214, 215, 215,
216, 216, 217, 217, 218, 218, 219, 219, 220, 220, 221, 221, 222,
222, 223, 223, 224, 224, 225, 225, 226, 226, 227, 227, 228, 228,
229, 229, 229, 230, 230, 231, 231, 232, 232, 233, 233, 234, 234,
235, 235, 235, 236, 236, 237, 237, 238, 238, 239, 239, 239, 240,
240, 241, 241, 242, 242, 242, 243, 243, 244, 244, 245, 245, 245,
246, 246, 247, 247, 248, 248, 248, 249, 249, 250, 250, 250, 251,
251, 252, 252, 252, 253, 253, 254, 254, 255]]) dtype=uint8)
```

## ▶ 감마보정

- [https://docs.opencv.org/3.4/d3/dc1/tutorial\\_basic\\_linear\\_transform.html](https://docs.opencv.org/3.4/d3/dc1/tutorial_basic_linear_transform.html)
- 파일: lena.png

```
In [4]: ### 감마보정 함수
def fn_gamma_correction(img, gamma=1):
    lookUpTable = np.empty((1, 256), np.uint8)
    for i in range(256) :
        lookUpTable[0, i] = np.clip(pow (i / 255.0, gamma) * 255.0, 0, 255)
    return cv2.LUT(img, lookUpTable)
```

```
In [5]: ### 영상 읽기
img_lena = cv2.imread(r'D:\Wimage\lena.png')
img_lena.shape
```

```
Out[5]: (512, 512, 3)
```

```
In [6]: ### 감마보정
img_gamma_1 = fn_gamma_correction(img_lena, 0.40)
img_gamma_2 = fn_gamma_correction(img_lena, 0.67)
img_gamma_3 = fn_gamma_correction(img_lena, 1.00)
img_gamma_4 = fn_gamma_correction(img_lena, 1.50)
img_gamma_5 = fn_gamma_correction(img_lena, 2.50)
```

```
In [7]: ### 영상 출력
titles = ["원래 영상", "$Wgamma$=0.40", "$Wgamma$=0.67", "$Wgamma$=1.00", "$Wgamma$=1.50", "$Wgamma$=2.50"]
images = [img_lena, img_gamma_1, img_gamma_2, img_gamma_3, img_gamma_4, img_gamma_5]
for i in range(len(images)):
    img_rgb = cv2.cvtColor(images[i], cv2.COLOR_BGR2RGB)
    plt.subplot(2, 3, i+1)
    plt.imshow(img_rgb)
    plt.title(titles[i])
    plt.axis('off')
plt.show()
```



## ▣ 예제

- 파일: image\_test.jpg

## ▣ 예제

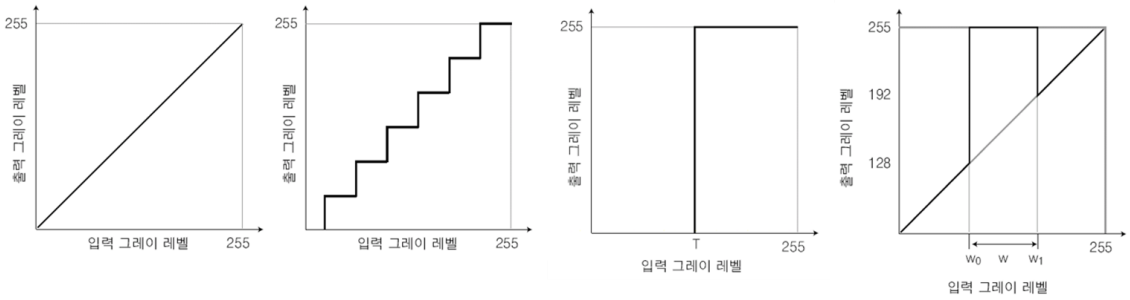
- 파일: PMS.jpg

## ▶ 경계 값을 이용한 처리

- 영상의 화소 값을 주어진 경계 값으로 그룹화하여 화소 값의 수를 감소시키는 처리
  - 포스터라이징(Posterizing)
    - 화소값의 범위를 경계 값으로 축소하는 기법
  - 이진화(Binarization)
    - 임계 값(threshold)보다 큰 화소는 255 그렇지 않은 화소는 0으로 변환

- 범위 강조

- 일정 범위의 화소만 강조하는 변환



```
In [8]: ### 경계 값을 이용한 변환
        levels = 5
        lookUpTable = np.arange(256, dtype='uint8')
        for i in range(levels):
            lookUpTable[(lookUpTable >= i*255/levels) & (lookUpTable < (i+1)*255/levels)] =
            lookUpTable
```

```
Out[8]: array([[ 0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
 63, 63, 63, 63, 63, 63, 63, 63, 63, 63, 63, 63, 63, 63,
 63, 63, 63, 63, 63, 63, 63, 63, 63, 63, 63, 63, 63,
 63, 63, 63, 63, 63, 63, 63, 63, 63, 63, 63, 63, 63,
 63, 63, 63, 63, 63, 63, 63, 63, 63, 63, 63, 127, 127,
127, 127, 127, 127, 127, 127, 127, 127, 127, 127, 127, 127,
127, 127, 127, 127, 127, 127, 127, 127, 127, 127, 127, 127,
127, 127, 127, 127, 127, 127, 127, 127, 127, 127, 127, 127,
127, 127, 127, 127, 127, 127, 127, 127, 127, 127, 191, 191, 191,
191, 191, 191, 191, 191, 191, 191, 191, 191, 191, 191, 191,
191, 191, 191, 191, 191, 191, 191, 191, 191, 191, 191, 191,
191, 191, 191, 191, 191, 191, 191, 191, 191, 191, 191, 191,
191, 191, 191, 191, 191, 191, 191, 191, 191, 255, 255, 255, 255,
255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255,
255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255,
255, 255, 255, 255, 255, 255, 255, 255, 255], dtype=uint8)
```

## ■ 포스터라이징

```
In [9]: ### 포스터라이징 함수
def fn_posterizing(img, levels=2):
    lookUpTable = np.arange(256, dtype='uint8')
    for i in range(levels):
        lookUpTable[(lookUpTable >= i*255/levels) & (lookUpTable < (i+1)*255/levels)] = i
    return cv2.LUT(img, lookUpTable)
```

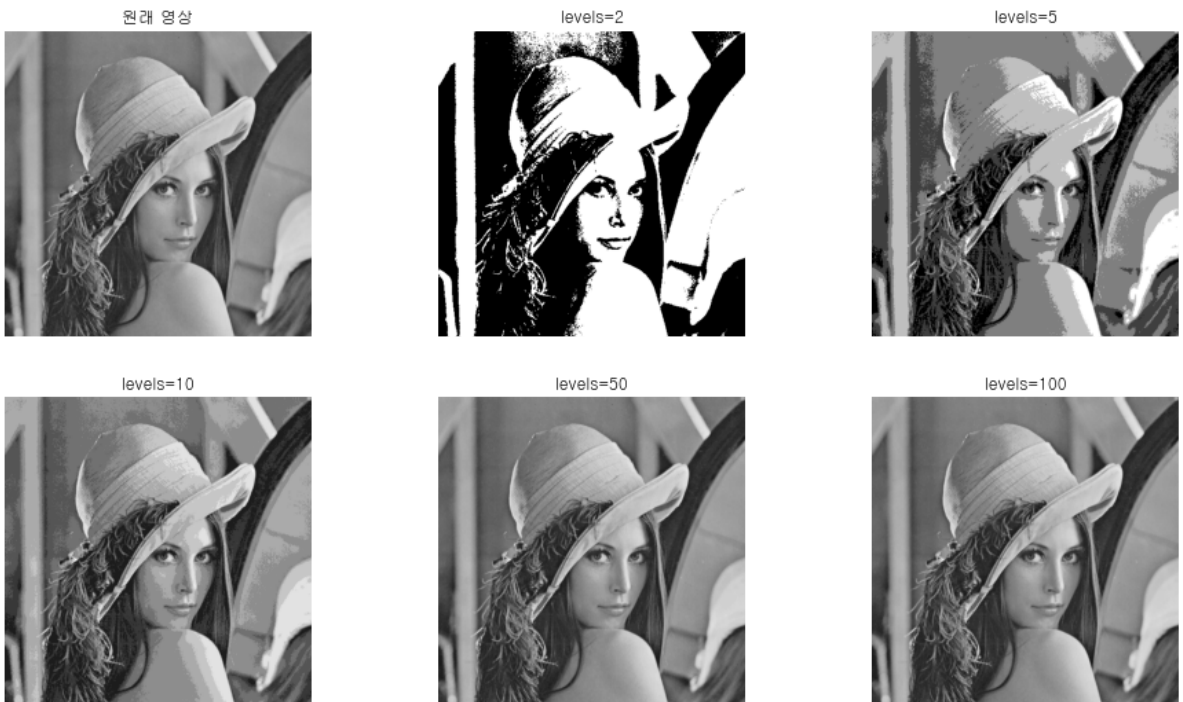
```
In [10]: ### 영상 읽기 - grayscale
img_lena_gray = cv2.imread(r'D:\image\lena.png', 0)
img_lena_gray.shape
```

```
Out[10]: (512, 512)
```

```
In [11]: ### 포스터라이징
img_poster_1 = fn_posterizing(img_lena_gray, 2)
img_poster_2 = fn_posterizing(img_lena_gray, 5)
img_poster_3 = fn_posterizing(img_lena_gray, 10)
```

```
img_poster_4 = fn_posterizing(img_lena_gray, 50)
img_poster_5 = fn_posterizing(img_lena_gray, 100)
```

```
In [12]: ### 영상 출력
titles = ["원래 영상", "levels=2", "levels=5", "levels=10", "levels=50", "levels=100"]
images = [img_lena_gray, img_poster_1, img_poster_2, img_poster_3, img_poster_4, img_poster_5]
for i in range(len(images)):
    img_rgb = cv2.cvtColor(images[i], cv2.COLOR_BGR2RGB)
    plt.subplot(2, 3, i+1)
    plt.imshow(img_rgb)
    plt.title(titles[i])
    plt.axis('off')
plt.show()
```



## ■ 범위 강조

```
In [13]: ### 범위 강조 함수
def fn_highlight(img, min_value, max_value, out_value):
    lookUpTable = np.arange(256, dtype='uint8')
    lookUpTable[(lookUpTable >= min_value) & (lookUpTable < max_value)] = out_value
    return cv2.LUT(img, lookUpTable)
```

```
In [14]: ### 범위 강조
img_highlight_1 = fn_highlight(img_lena_gray, 200, 255, 255)
img_highlight_2 = fn_highlight(img_lena_gray, 0, 150, 0)
img_highlight_3 = fn_highlight(img_lena_gray, 50, 100, 0)
```

```
In [15]: ### 영상 출력
titles = ["원래 영상", "200~255→255", "0~150→0", "50~100→0"]
images = [img_lena_gray, img_highlight_1, img_highlight_2, img_highlight_3]
for i in range(len(images)):
    img_rgb = cv2.cvtColor(images[i], cv2.COLOR_BGR2RGB)
    plt.subplot(2, 2, i+1)
    plt.imshow(img_rgb)
    plt.title(titles[i])
    plt.axis('off')
plt.show()
```

원래 영상



200~255→255



0~150→0



50~100→0



## ■ 이진화(Binarization)

- [https://docs.opencv.org/master/d7/d4d/tutorial\\_py\\_thresholding.html](https://docs.opencv.org/master/d7/d4d/tutorial_py_thresholding.html)
- `cv2.threshold(src, thresh, maxval, type)`
  - `src`: grayscale image
  - `thresh`: 임계값
  - `maxval`: 임계값을 넘을 때 적용할 값
  - `type`: thresholding type

THRESH\_BINARY  
Python: `cv.THRESH_BINARY`

$$\text{dst}(x, y) = \begin{cases} \text{maxval} & \text{if } \text{src}(x, y) > \text{thresh} \\ 0 & \text{otherwise} \end{cases}$$

THRESH\_BINARY\_INV  
Python: `cv.THRESH_BINARY_INV`

$$\text{dst}(x, y) = \begin{cases} 0 & \text{if } \text{src}(x, y) > \text{thresh} \\ \text{maxval} & \text{otherwise} \end{cases}$$

THRESH\_TRUNC  
Python: `cv.THRESH_TRUNC`

$$\text{dst}(x, y) = \begin{cases} \text{threshold} & \text{if } \text{src}(x, y) > \text{thresh} \\ \text{src}(x, y) & \text{otherwise} \end{cases}$$

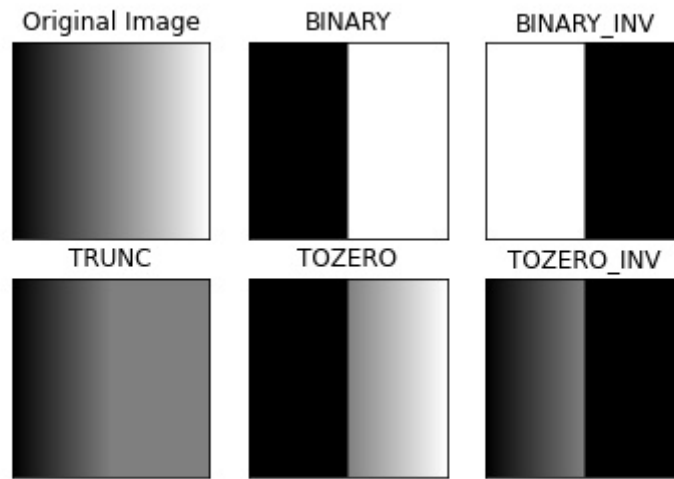
THRESH\_TOZERO  
Python: `cv.THRESH_TOZERO`

$$\text{dst}(x, y) = \begin{cases} \text{src}(x, y) & \text{if } \text{src}(x, y) > \text{thresh} \\ 0 & \text{otherwise} \end{cases}$$

THRESH\_TOZERO\_INV  
Python: `cv.THRESH_TOZERO_INV`

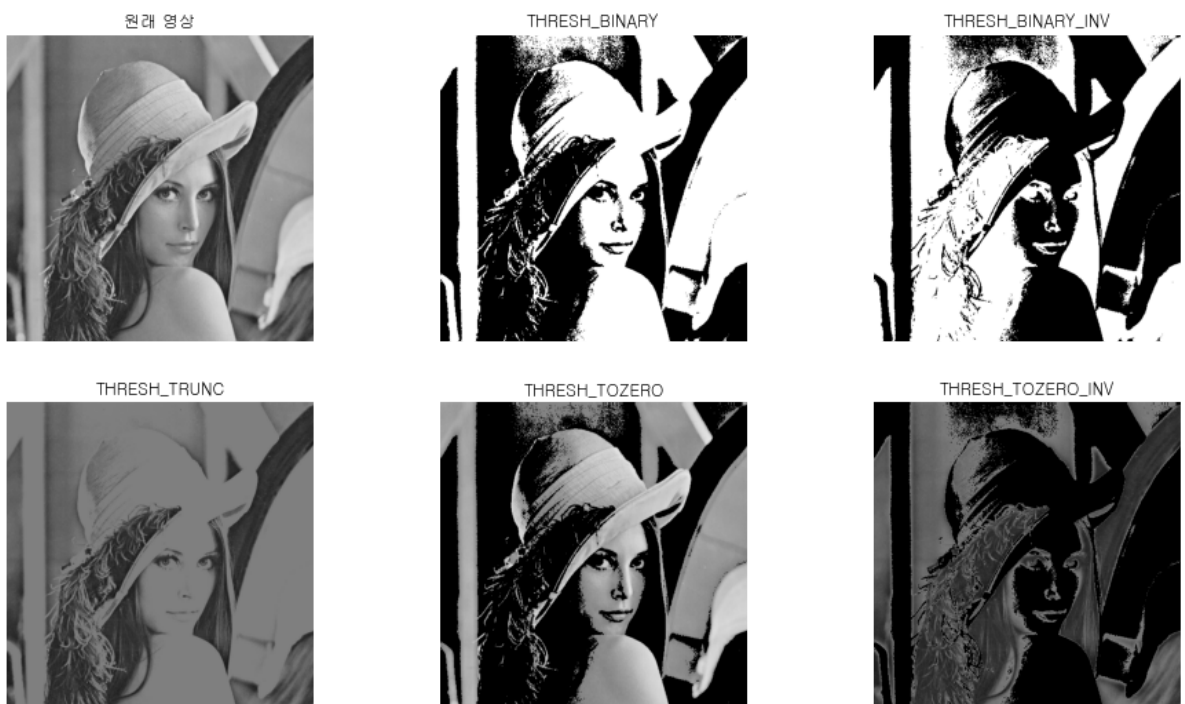
$$\text{dst}(x, y) = \begin{cases} 0 & \text{if } \text{src}(x, y) > \text{thresh} \\ \text{src}(x, y) & \text{otherwise} \end{cases}$$





```
In [16]: ### 이진화 - grayscale
ret, img_thresh_1 = cv2.threshold(img_lena_gray, 127, 255, cv2.THRESH_BINARY)
ret, img_thresh_2 = cv2.threshold(img_lena_gray, 127, 255, cv2.THRESH_BINARY_INV)
ret, img_thresh_3 = cv2.threshold(img_lena_gray, 127, 255, cv2.THRESH_TRUNC)
ret, img_thresh_4 = cv2.threshold(img_lena_gray, 127, 255, cv2.THRESH_TOZERO)
ret, img_thresh_5 = cv2.threshold(img_lena_gray, 127, 255, cv2.THRESH_TOZERO_INV)
```

```
In [17]: ### 영상 출력
titles = ["원래 영상", "THRESH_BINARY", "THRESH_BINARY_INV", "THRESH_TRUNC", "THRESH_TOZERO", "THRESH_TOZERO_INV"]
images = [img_lena_gray, img_thresh_1, img_thresh_2, img_thresh_3, img_thresh_4, img_thresh_5]
for i in range(len(images)):
    img_rgb = cv2.cvtColor(images[i], cv2.COLOR_BGR2RGB)
    plt.subplot(2, 3, i+1)
    plt.imshow(img_rgb)
    plt.title(titles[i])
    plt.axis('off')
plt.show()
```



## ■ 영상 합성

- 파일 (1): bp\_white\_re.jpg
- 파일 (2): img1.jpg

```
In [18]: ### 영상 읽기
img_raw = cv2.imread(r'D:\WimageWbp_white_re.jpg')

### 마스크 - 이진영상
img_gray = cv2.cvtColor(img_raw, cv2.COLOR_BGR2GRAY)
ret, img_binary_inv = cv2.threshold(img_gray, 240, 255, cv2.THRESH_BINARY_INV)

### 논리연산 → 영상 추출
img_bp = cv2.bitwise_and(img_raw, cv2.cvtColor(img_binary_inv, cv2.COLOR_GRAY2BGR))
```

```
In [19]: ### 영상 읽기
img1 = cv2.imread(r'D:\WimageWimg1.jpg')

### 마스크 - 이진영상
ret, img_binary = cv2.threshold(img_gray, 240, 255, cv2.THRESH_BINARY)

### 논리연산 → 영상 추출
img_bg = cv2.bitwise_and(img1, cv2.cvtColor(img_binary, cv2.COLOR_GRAY2BGR))
```

```
In [20]: ### 산술연산 → 영상 합성
img_bp_bg = cv2.add(img_bp, img_bg)
```

```
In [21]: ### 영상 출력
images = [img_raw, img_binary_inv, img_bp,
          img1, img_binary, img_bg,
          img_bp, img_bg, img_bp_bg,]
for i in range(len(images)):
    img_rgb = cv2.cvtColor(images[i], cv2.COLOR_BGR2RGB)
    plt.subplot(3,3,i+1),plt.imshow(img_rgb)
    plt.axis('off')
plt.show()
```

