[IP] 컴퓨터융합학부_202102699_정민경_2주차_과제 (Histogram equalization)

1. 구현 코드

- my_calcHist (src)

```
def my_calcHist(src):
    ##############################
    # TODO                       #
    # my_calcHist완성            #
    # src : input image          #
    # hist : src의 히스토그램    #
    ##############################

    array = []
    for i in src_:
        for j in range(len(i)):
            array.append(i[j])

    hist = [0.0] * 256

    for i in array:
        hist[i] = hist[i] + 1

    a = np.array(hist)
    return a
```

2차원 배열을 1차원으로 쭉 늘린 후 histogram을 저장할 'hist' 배열에 각각의 값이 몇 개 나왔는지 저장해야하므로 값을 + 해준 것이 아니라 횟수를 저장해주고, 그렇게 만든 hist를 np.array를 사용해 배열로 만들어 return 해주었다.
(hist에 저장방법 : 2가 나온다면 hist[2] += 1 로 구현해 2가 나온 횟수를 hist 에 저장)

- my_normalize_hist (hist, pixel_num)

```
def my_normalize_hist(hist, pixel_num):
    ##########################################################
    # TODO                                                   #
    # my_normalize_hist완성                                  #
    # hist : 히스토그램                                      #
    # pixel_num : image의 전체 픽셀 수                       #
    # normalized_hist : 히스토그램값을 총 픽셀수로 나눔      #
    ##########################################################

    normalized_hist = np.zeros(hist.shape)
    for i in range(len(hist)):
        normalized_hist[i] = float(hist[i]) / float(pixel_num)

    return np.array(normalized_hist)
```

normalized_hist에 각각의 hist를 총 픽셀수 (pixel_num) 으로 나눠 저장한다.
(normalized_hist = hist / pixel_num, 이 때 normalized_hist 의 값은 float type)

- my_PDF2CDF (pdf)

```
def my_PDF2CDF(pdf):
    ###########################################################
    # TODO                                                    #
    # my_PDF2CDF완성                                          #
    # pdf : normalized_hist                                   #
    # cdf : pdf의 누적                                        #
    ###########################################################
    cdf = np.zeros(pdf.shape)
    cdf[0] = 0 + pdf[0]
    for i in range(1, len(pdf)):
        cdf[i] = cdf[i-1] + pdf[i]
    return np.array(cdf)
```

normalized 된 histogram (= pdf) 를 누적시켜 cdf 배열에 저장시켜주었다.
i번째 cdf에는 이 전까지 누적된 pdf와 i번째 pdf를 더해야하므로, i-1번째 cdf와 i번째 pdf를 더해주며 저장해주었고, 이러한 cdf는 1차원 배열의 형태가 되어야 하므로 np.array(cdf) 로 return 해주었다.

- my_denormalize (normalized, gray_level)

```
def my_denormalize(normalized, gray_level):
    ###########################################################
    # TODO                                                    #
    # my_denormalize완성                                      #
    # normalized : 누적된pdf값(cdf)                           #
    # gray_level : max_gray_level                             #
    # denormalized : normalized와 gray_level을 곱함           #
    ###########################################################
    denormalized = np.zeros(normalized.shape)
    for i in range(len(normalized)):
        denormalized[i] = normalized[i] * gray_level
    return np.array(denormalized)
```

확률로 바꿨던 값을 다시 gray level 로 바꾸기 위해 normalized와 gray_level 을 곱해 denormalized 배열에 저장한다. return 값은 1차원 배열이므로 np.array(denormalized)를 통해 1차원 배열을 return 한다.

- my_calcHist_equalization (denormalized, hist)

```
def my_calcHist_equalization(denormalized, hist):
    ###################################################################
    # TODO                                                            #
    # my_calcHist_equalization완성                                    #
    # denormalized : output gray_level(정수값으로 변경된 gray_level)   #
    # hist : 히스토그램                                               #
    # hist_equal : equalization된 히스토그램                          #
    ###################################################################
    hist_equal = np.zeros(hist.shape)

    for i in range(len(hist)):
        hist_equal[denormalized[i]] = hist_equal[denormalized[i]] + int(hist[i])

    return np.array(hist_equal)
```

지금까지의 과정으로 구한 output gray_level을 가지고 equalization을 수행하는 함수이다. output gray_level을 각각의 index로 보고, 그 index에 각각의 histogram을 더해서 저장하면 된다. 따라서 equalization 하는 배열의 index는 denormalized이고, 그 index 위치에 기존 hist의 값들을 더해 저장하면 된다.

위의 my_calcHist_equalization 함수에서 for 문 안의 코드는 hist 길이만큼 돌면서 각각의 index를 기존의 존재하던 값(초깃값 : 0)과 누적하면서 저장하도록 구현했다.

- my_equal_img (src, output_gray_level)

```
def my_equal_img(src, output_gray_level):
    ###################################################################
    # TODO                                                            #
    # my_equal_img완성                                                #
    # src : input image                                               #
    # output_gray_level : denormalized(정수값으로 변경된 gray_level)   #
    # dst : equalization된 결과 이미지                                #
    ###################################################################

    (h, w) = src.shape # h 는 몇줄인지 / w 는 몇열인지
    dst = np.zeros((h, w), dtype=np.uint8) # 이차원배열

    for row in range(h):
        for col in range(w):
            dst[row, col] = output_gray_level[src[row, col]]

    return dst
```

마지막 my_equal_img 함수는 지금까지 equalization 한 것을 원본이미지에 적용시켜 결괏값도 원본이미지처럼 2차원배열로 return 해 equalization 된 사진이 나올 수 있도록 해주는 함수이다. 그렇기 때문에 원본 이미지를 처음부터 끝까지 쭉 돌면서 결과가 저장되는 dst 배열에 equalization된 값을 저장한다. equalization 된 값을 구하려면 우리가 구해놓은 output_gray_level 배열을 받아오면 되는데, 결괏값이 저장되는 dst 와 원본 src는 같은 index로 매칭되어야하기 때문에 dst에는 output_gray_level 의 원본에서의 동일 index 값 위치에 있는 값을 저장해주면 된다.

- 각각의 이미지 저장 (in [IP]202102699_homework.py)

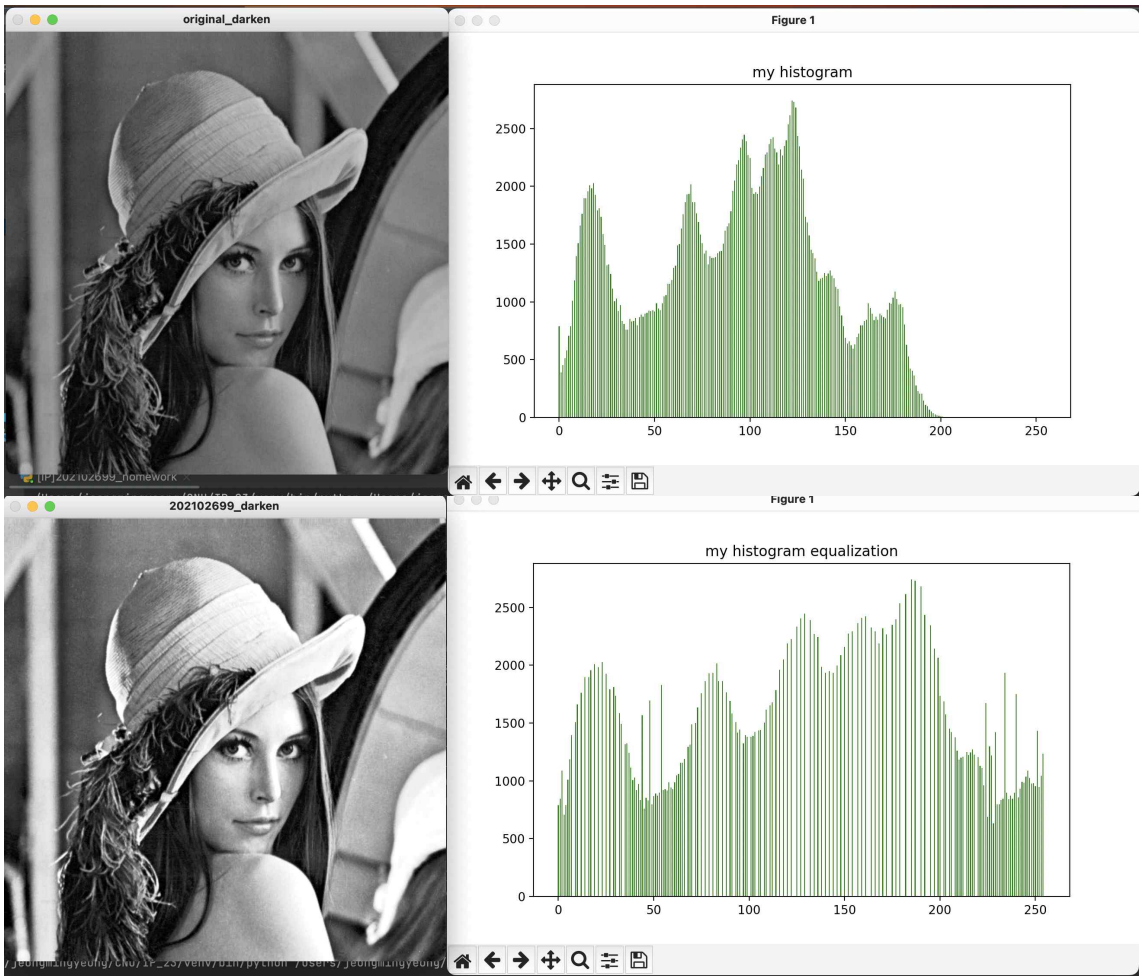
```
### darken
src = cv2.imread('save_image/Lena_darken.jpg', cv2.IMREAD_GRAYSCALE)
hist = my_calcHist(src)
dst, hist_equal = my_hist_equal(src)

plt.figure(figsize=(8, 5))
cv2.imshow('original_darken', src)
cv2.imwrite('result_image/original_darken.png', src)
binX = np.arange(len(hist))
plt.title('my_darken_histogram')
plt.bar(binX, hist, width=0.5, color='g')
plt.show()

plt.figure(figsize=(8, 5))
cv2.imshow('202102699_darken', dst)
cv2.imwrite('result_image/202102699_darken.png', dst)
binX = np.arange(len(hist_equal))
plt.title('my_darken_histogram equalization')
plt.bar(binX, hist_equal, width=0.5, color='g')
plt.show()
```

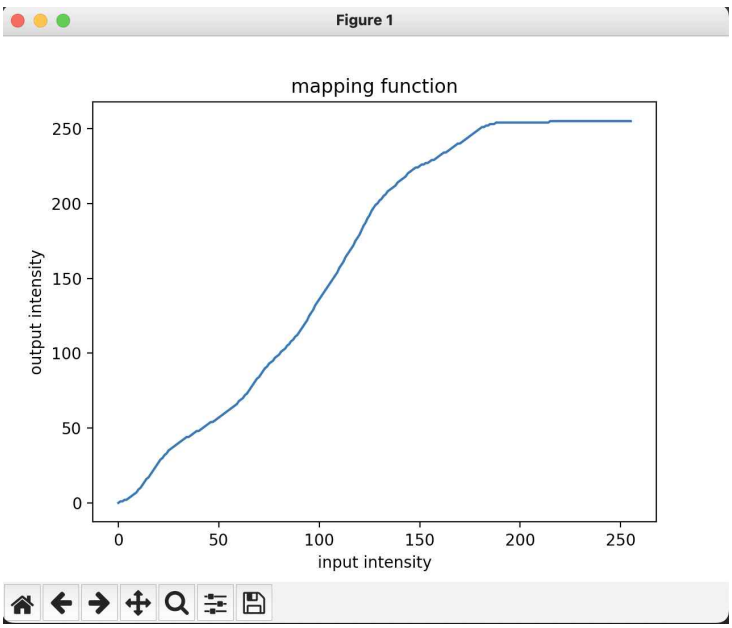
각각의 이미지 (darken, lighten, lower_contrast, raise_contrast) 들을 equalization 시켜 저장하는 것을 각각의 이미지에 따라 저장해주었다.

equalization 된 결과이미지는 'result_image' 폴더에 '202102699_이미지이름' 으로 저장했고, original 역시 'original_이미지이름' 으로 저장했다.

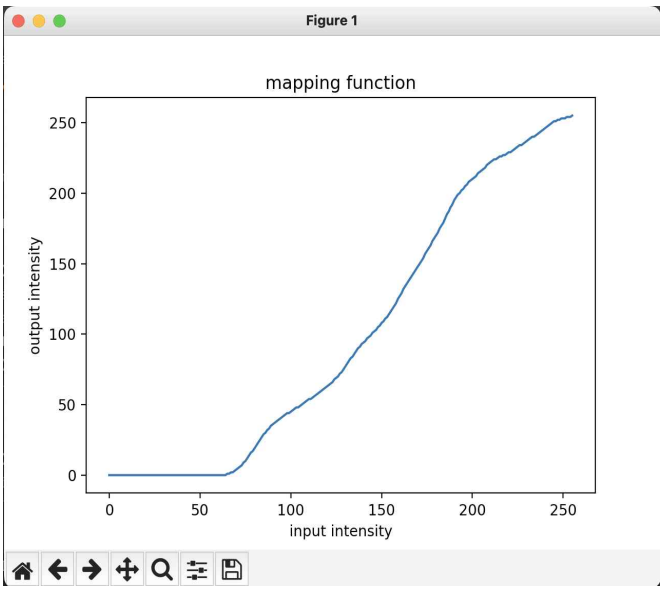
2. equalization 결과 이미지

[darken image equalization]

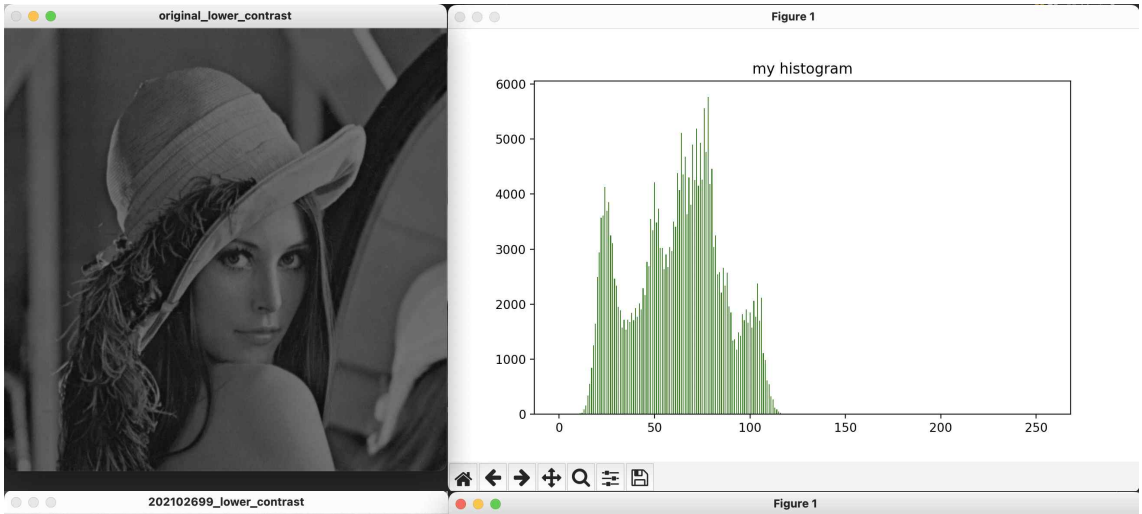


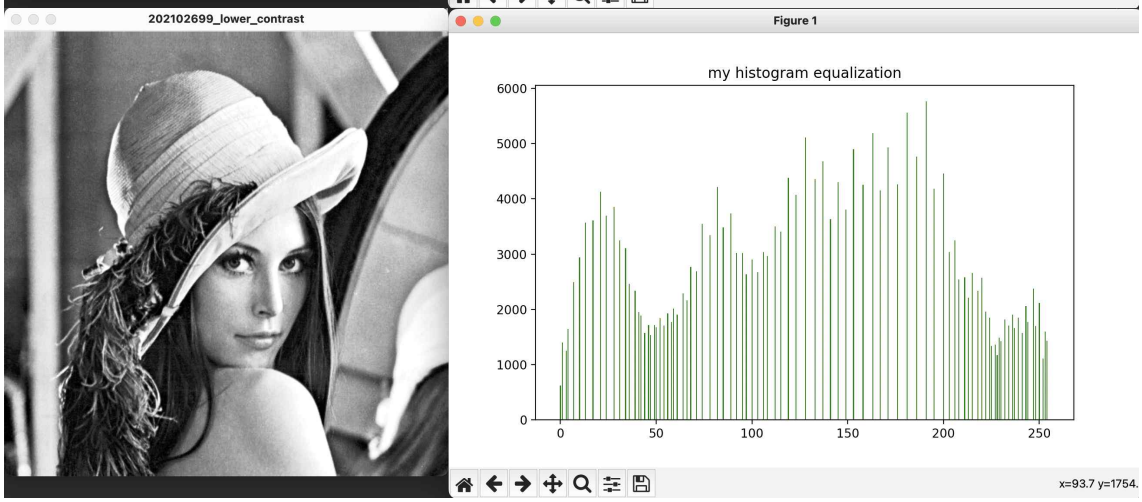


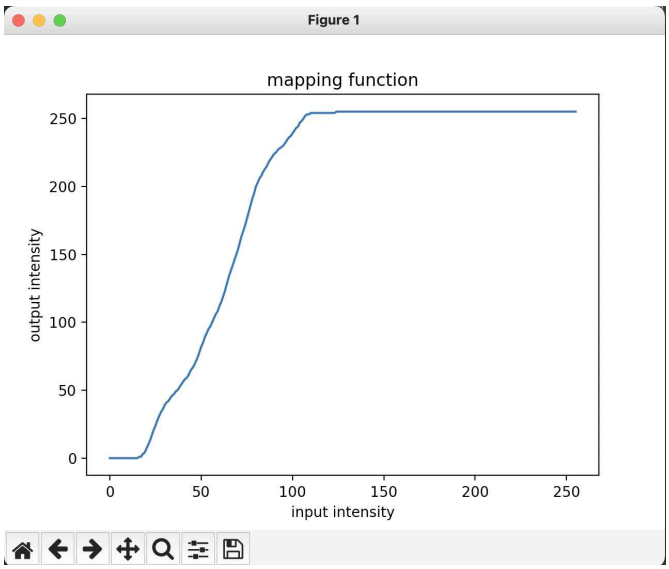
[lighten image equalization]



[lower_contrast image equalization]







[raise_contrast image equalization]