Lab1_2013755_강단이

1.1 Color 분리

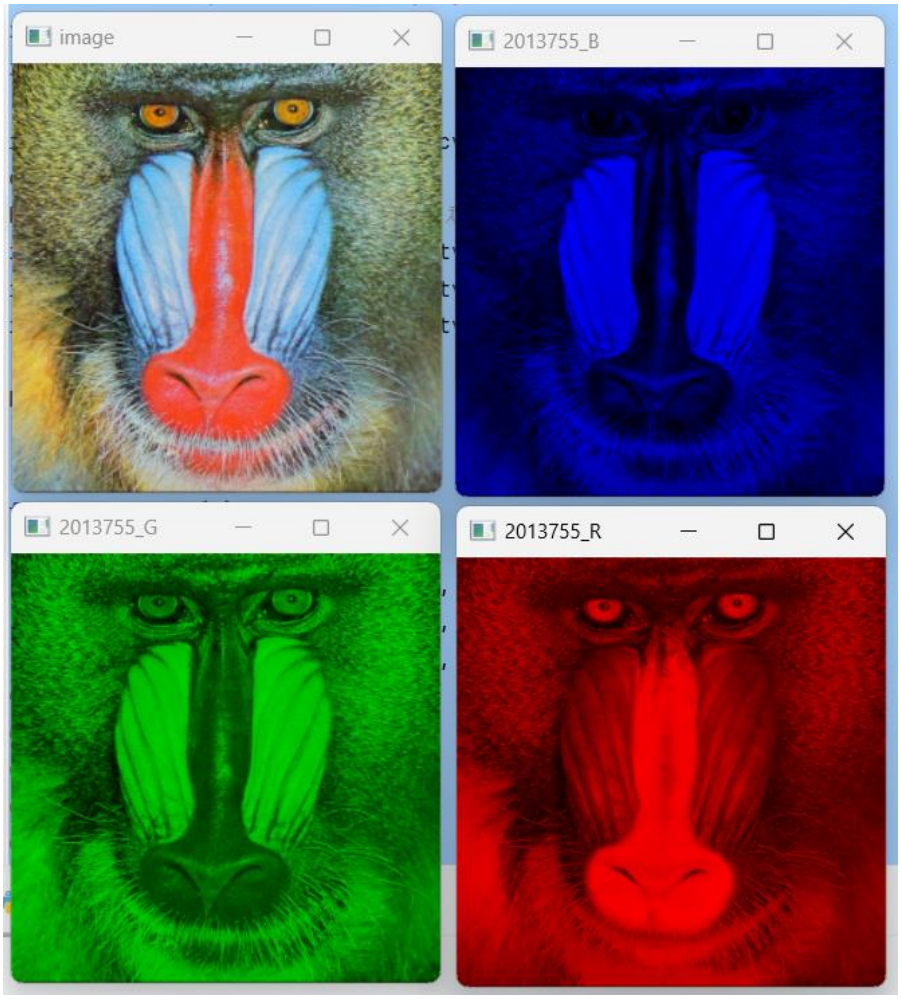
세부 기능 설명

Mandrill 영상을 img로 받았고, img2, img3, img4에는 본래와 동일한 가로 t로 사이즈에 0값으로 초기화를 해준다.

가로 세로를 for문으로 돌면서 원본인 img의 0번째 컬러 채널인 파랑의 정보만 img2의 파랑 컬러 채널에 저장하고, img3는 채널 1인 초록, img4는 채널 2인 빨강의 정보를 저장한다.

계산으로 나온 이미지들을 저장하며 화면에 띄워준다.

기능별 실행 화면 캡쳐



소스코드

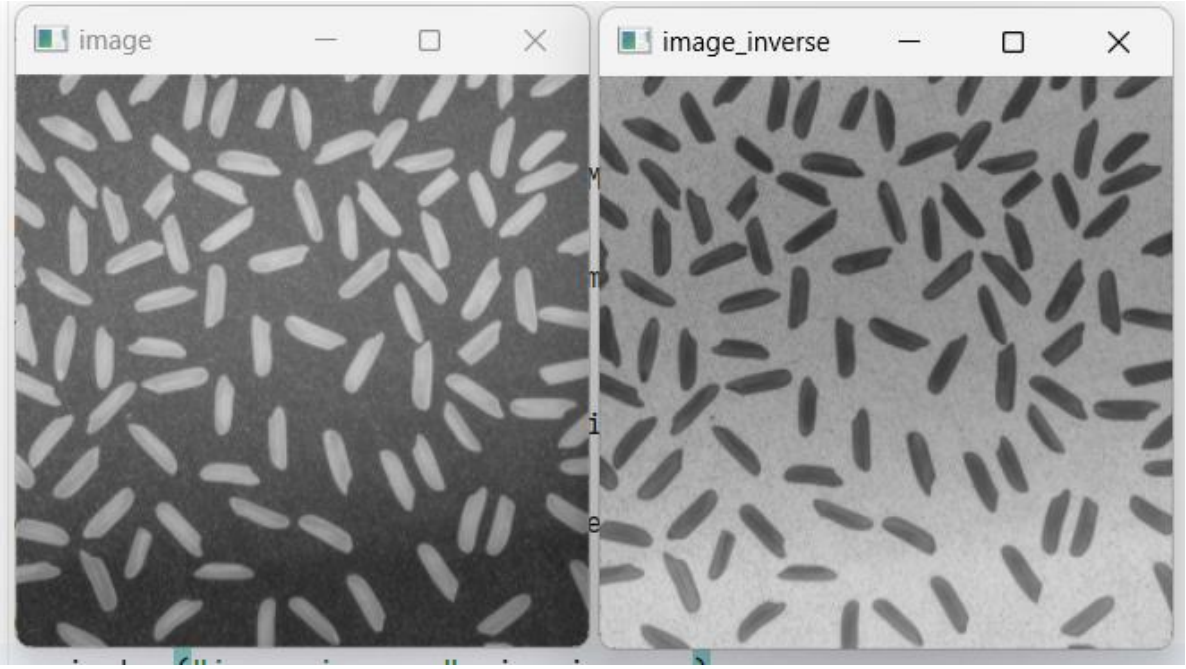
```
# lab02_colorSeparation into R, G, B
import cv2 as cv
import numpy as np

img = cv.imread("Mandrill.bmp", cv.IMREAD_COLOR)
cv.imshow("image", img)
H, W, C = img.shape[:] #높이 깊이 채널 개수
img2 = np.zeros((H, W, C), img.dtype) #0값을 넣음
img3 = np.zeros((H, W, C), img.dtype)
img4 = np.zeros((H, W, C), img.dtype)

h, w = img.shape[:2]

for y in range(h):
    for x in range(w):
        img2[y, x, 0] = img[y, x, 0]
        img3[y, x, 1] = img[y, x, 1]
        img4[y, x, 2] = img[y, x, 2]
cv.imwrite("lap02-1B.png", img2)
cv.imwrite("lap02-1G.png", img3)
cv.imwrite("lap02-1R.png", img4)
cv.imshow("2013755_B", img2)
cv.imshow("2013755_G", img3)
cv.imshow("2013755_R", img4)
cv.waitKey(0)
```

1.2 흑백영상 역상 이미지 만들기

세부 기능 설명

Rice.bmp를 읽어와 가로 세로 동일한 img_inverse를 만들고, 가로 세로 for문을 돌려 제일 밝은 값인 255에서 본래 img의 밝기를 빼주면 반전이 된다. 해당 값을 저장해 출력한다.

기능별 실행 화면 캡쳐



소스코드

```
# lab02_inverse
import cv2 as cv
import numpy as np

img = cv.imread("rice.bmp", cv.IMREAD_GRAYSCALE)
H, W= img.shape[:]
img_inverse = np.zeros((H, W), img.dtype)
for y in range(H):
    for x in range(W):
        img_inverse[y,x]= 255 - img[y, x]

cv.imwrite("inverse.png",img_inverse)

cv.imshow("image",img)
cv.imshow("image_inverse", img_inverse)
cv.waitKey(0)
```

1.3 Histogram Equalization

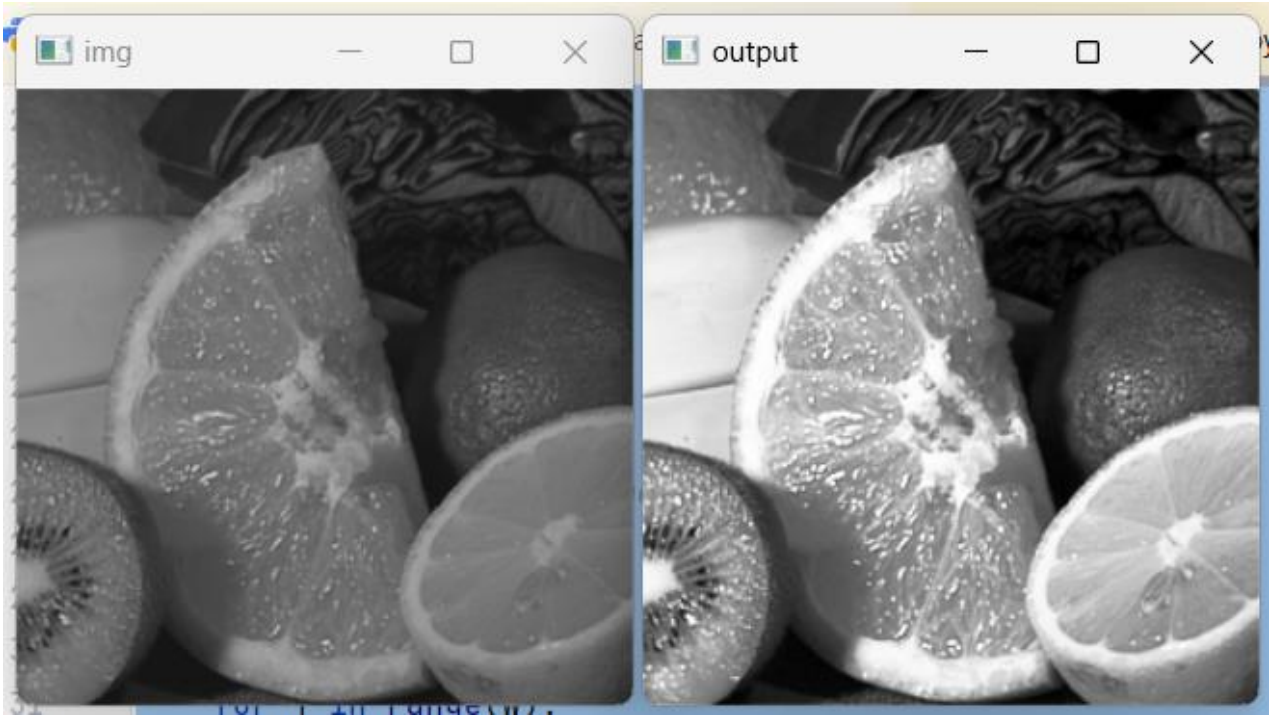
세부 기능 설명

Citrus 이미지를 읽어오고, 히스토그램과 누적합을 저장하기 위한 int 형식의 1차원 배열을 선언한다. 이미지의 크기에 맞게 for문으로 반복하여 원본 이미지 픽셀 하나의 밝기값에 해당되는 히스토그램 값을 늘려주며 hist[] 값을 얻는다.

누적합을 얻기 위해 hist에 저장된 값을 sum에 저장하며 sum값을 sum_of_hist에 저장해준다.

전체 화소 크기를 area에 저장하고, 최대 gray level인 255를 Dm 변수로 선언한 뒤, 이미지의 크기만큼 반복하며 원본의 화소값을 k에 저장하고, 누적합의 k번째에 해당되는 수에 Dm / area를 곱해준다.

기능별 실행 화면 캡쳐



소스코드

```
import cv2 as cv
import numpy as np

img = cv.imread("citrus.bmp", cv.IMREAD_GRAYSCALE)
cv.imshow("img", img)
H, W = img.shape[:]

output = np.zeros((H, W), img.dtype)
#valuable 선언 및 초기화
hist = np.zeros((256), int)
sum_of_hist = np.zeros((256), int)

#1. Calculation of histogram
for y in range(H):
    for x in range(W):
        k = img[y, x]
        hist[k] = hist[k]+1

sum=0
#2. Cumulative Histogram
for i in range(256):
    sum = sum + hist[i]
    sum_of_hist[i] = sum

#3. Transform the input image to output image
area = H * W
Dm = 255
for i in range(H):
    for j in range(W):
       k = img[i,j]
       output[i,j] = (Dm / area) * sum_of_hist[k]

cv.imshow("output", output)
cv.imwrite("lap02-3.png", output)
cv.waitKey(0)
```

1.4 Otsu Binarization

세부 기능 설명

픽셀의 빈도를 확률로 나타내는 P, 픽셀의 빈도를 나타내는 히스토그램 hist, 임계값 t를 기준으로 둘로 나눌 때 각자 구역속 픽셀들의 누적 확률 q1 / q2, 각 구역의 평균 명암값을 나타내는 mu1 / mu2, 각 구역의 분산을 계산한 sigma1 / sigma2, 임계값 t에 대한 분산의 가중합 sigmamu를 선언했다.

For문으로 hist[] 값인 히스토그램을 얻었다. 히스토그램 각각의 값이 이미지에서 얼마나의 비율을 차지하는지 확률을 계산해 P[]에 저장했다. T의 값을 찾기 위해 t가 0~255일 때까지 반복했고, 그 과정에서 0부터 t까지인 구역의 누적합은 q1, t+1부터 255까지는 q2에 누적 확률을 저장한다.

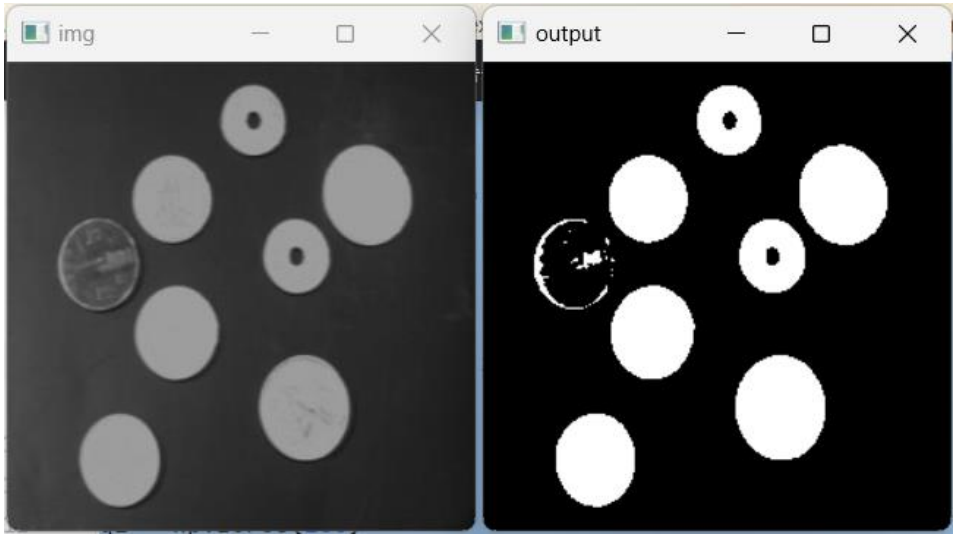
T 값을 기준으로 이보다 작고 큰 평균 명암값을 mu1에 저장한다. Q1[t]로 나눌 때 0으로 나누지 않기 위해 조건식을 추가했다.

수업시간에 배운 식을 이용해 분산을 구하는 sigma 계산식을 추가했다. 마찬가지로 0으로 나누지 않게 조건식을 추가했다.

0부터 255중 최적의 임계값 t를 구하기 위해 for문을 돌려 두 구역의 분산을 가중평균한 sigmamu를 구해 minSigma보다 작으면 이를 기억하는 식으로 minSigma가 최소가 되는 t값을 찾는다.

얻어진 임계값을 기준으로 픽셀 값을 두 가지 값(255, 0)으로 변환하여 이진화를 수행한다.

기능별 실행 화면 캡쳐



소스코드

```
import cv2 as cv
import numpy as np

img = cv.imread("coin.bmp", cv.IMREAD_GRAYSCALE)
cv.imshow("img", img)
H, W = img.shape[:]

output = np.zeros((H, W), img.dtype)
    #valuable 선언 및 초기화
P = np.zeros(256)
hist = np.zeros(256)
q1 = np.zeros(256)
q2 = np.zeros(256)
mu1 = np.zeros(256)
mu2 = np.zeros(256)
sigma1 = np.zeros(256)
sigma2 = np.zeros(256)
sigmamu = np.zeros(256)

    #1. Calculation of histogram
for y in range(H):
    for x in range(W):
        k = img[y, x]
        hist[k] = hist[k]+1

    #2. P[] 획득
for i in range(256):
    P[i] = hist[i] / (H * W) #명암값이 i일 확률 P[i]

    #3. Otsu Algorithm
for t in range(256):
    for i in range(t+1):
        q1[t] = q1[t] + P[i]
    for i in range(t+1, 256):
        q2[t] = q2[t] + P[i]
    for i in range(t+1):
        if q1[t] > 0:
            mu1[t] = mu1[t] + i * P[i] / q1[t]
    for i in range(t+1, 256):
        if q2[t]> 0:
```

```
                mu2[t] = mu2[t] + i * P[i] / q2[t]
        for i in range(t+1):
            if q1[t] > 0:
                sigma1[t] = sigma1[t] + (i - mu1[t])**2 * P[i] / q1[t]
        for i in range(t+1, 256):
            if q2[t] > 0:
                sigma2[t] = sigma2[t] + (i - mu2[t])**2 * P[i] / q2[t]

        #4. Sigma 획득
for t in range(256):
        sigmamu[t] = q1[t] * sigma1[t] + q2[t] * sigma2[t]

        #5. 최소로 만드는 t값 획득
minSigma = np.inf
minT = 0
for i in range(256):
        if minSigma > sigmamu[i]:
            minT = i
            minSigma = sigmamu[i]

        #6. 획득된 t값을 이용해 이진화
for i in range(H):
        for j in range(W):
            if img[i, j] > minT:
                output[i, j] = 255
            else:
                output[i, j] = 0

cv.imshow("output", output)
cv.imwrite("lap02-4.png", output)
cv.waitKey(0)
```