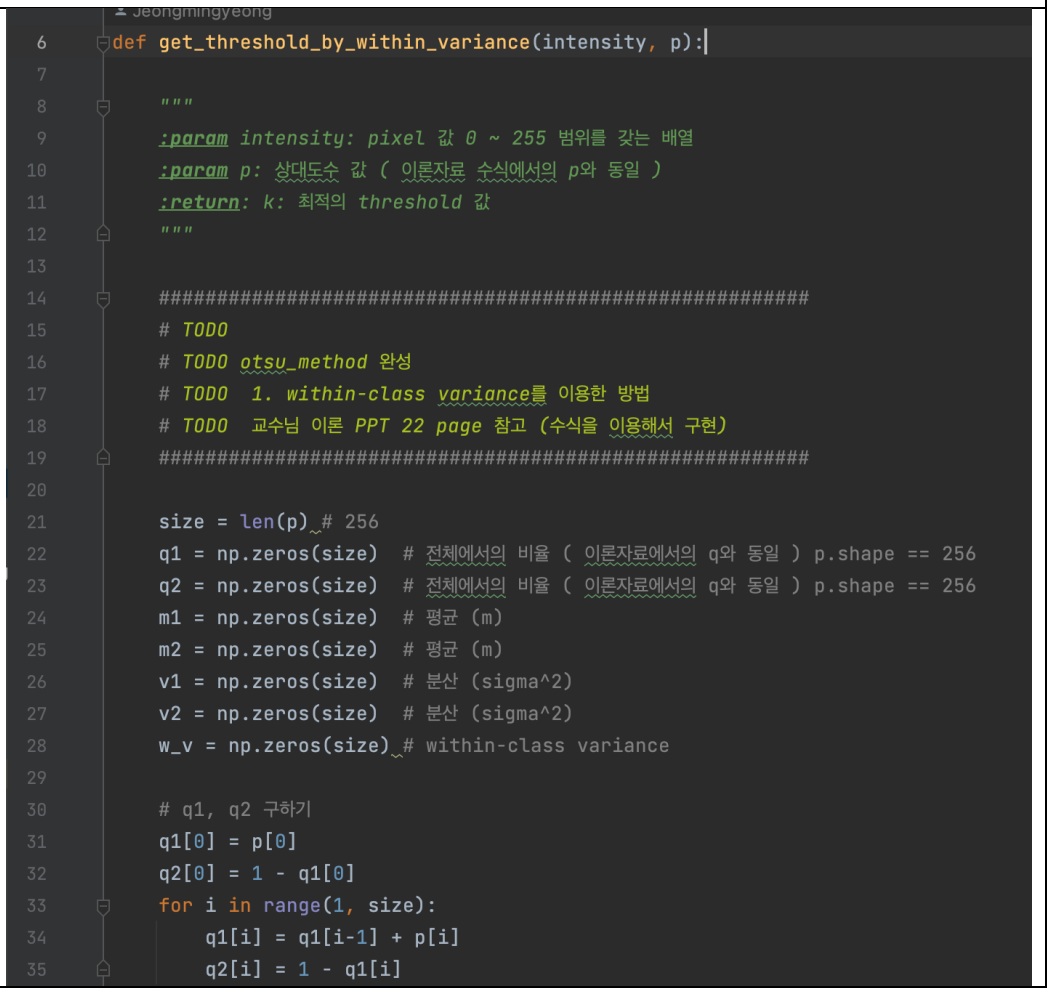
영상처리

9 주차 보고서

컴퓨터융합학부

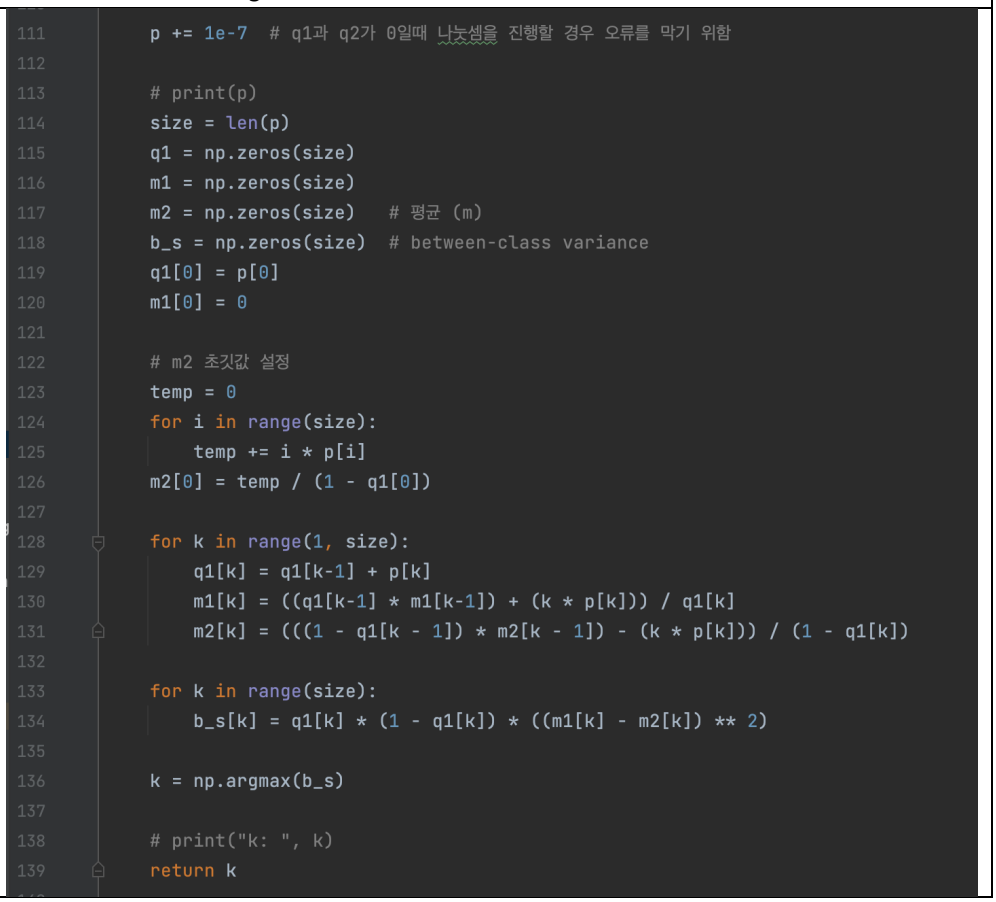
202102699 정민경

1. 과제 설명

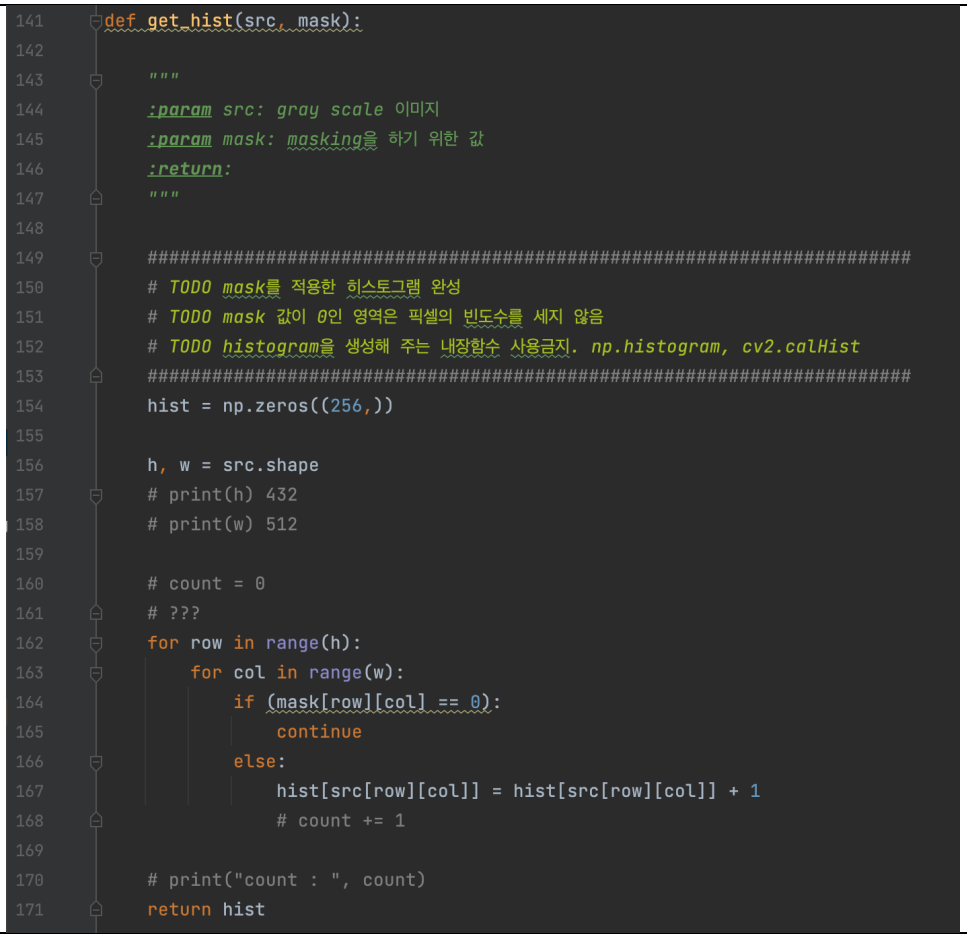
함수이름	get_threshold_by_within_variance ()
코드설명	이 함수는 “within-class variance” 방법으로 최적의 threshold 를 구하는 함수이다. 구현 방법은 이론자료에 있는 수식으로 구현했다. Within-class variance 방법은 2개의 class 로 나눠서 class 1, 2 각각의 q, m(평균), v(분산) 을 구한 후 최종적으로 w_v = q1 * v1 + q2 * v2 로 구한 within-class variance 의 최솟값을 찾는 argmin 함수를 사용해 최적의 threshold 를 구해주었다. 이 때 구현방법에서 신경써야 할 부분은 class 1 같은 경우 0부터 k 까지의 값이므로 for I in range (k+1 (또는 1, k+1)) 로 구현하면 되지만, class 2 같은 경우는 k+1 부터 size 만큼까지 구해야 하므로 for I in range (size - 2, -1, -1) 로 구현해서 size - 2 부터 0까지 돌며 그 안의 이중 for문으로 k+1 까지만 구할 수 있도록 구현해야한다.
이미지	

```python
Jeongmingyeong
def get_threshold_by_within_variance(intensity, p):

    """
    :param intensity: pixel 값 0 ~ 255 범위를 갖는 배열
    :param p: 상대도수 값 ( 이론자료 수식에서의 p와 동일 )
    :return: k: 최적의 threshold 값
    """

    ############################################################
    # TODO
    # TODO otsu_method 완성
    # TODO  1. within-class variance를 이용한 방법
    # TODO  교수님 이론 PPT 22 page 참고 (수식을 이용해서 구현)
    ############################################################

    size = len(p)# 256
    q1 = np.zeros(size)  # 전체에서의 비율 ( 이론자료에서의 q와 동일 ) p.shape == 256
    q2 = np.zeros(size)  # 전체에서의 비율 ( 이론자료에서의 q와 동일 ) p.shape == 256
    m1 = np.zeros(size)  # 평균 (m)
    m2 = np.zeros(size)  # 평균 (m)
    v1 = np.zeros(size)  # 분산 (sigma^2)
    v2 = np.zeros(size)  # 분산 (sigma^2)
    w_v = np.zeros(size)# within-class variance

    # q1, q2 구하기
    q1[0] = p[0]
    q2[0] = 1 - q1[0]
    for i in range(1, size):
        q1[i] = q1[i-1] + p[i]
        q2[i] = 1 - q1[i]
```

```
    # m1, m2 구하기
    m1[0] = 0
    for i in range(1, size):
        temp = 0
        for j in range(i+1):
            temp += (intensity[j] * p[j])
        if (q1[i] == 0):
            continue
        else:
            m1[i] = temp / q1[i]

    # m2 초기화
    if (q2[size-1] == 0):
        m2[size-1] = 0
    else:
        m2[size-1] = (intensity[size-1] * p[size-1]) / q2[size-1]

    for i in range(size-2, -1, -1): # size-2 부터 0까지 -1씩 줄여나가며 for 문 진행
        temp = 0
        for j in range(size-2, i, -1):
            temp += (intensity[j] * p[j])
        if (q2[i] == 0):
            continue
        else:
            m2[i] = temp / q2[i]

# v1, v2 구하기
v1[0] = 0
for i in range(1, size):
    temp = 0
    for j in range(i+1):
        temp += (((intensity[j] - m1[i]) ** 2) * p[j])
    if (q1[i] == 0):
        continue
    else:
        v1[i] = temp / q1[i]

# v2 초기화
if (q2[size - 1] == 0):
    v2[size - 1] = 0
else:
    v2[size - 1] = (((intensity[size-1] - m1[size-1]) ** 2) * p[size-1]) / q2[size - 1]

for i in range(size-2, -1, -1): # size-2 부터 0까지 -1씩 줄여나가며 for 문 진행
    temp = 0
    for j in range(size-2, i, -1):
        temp += (((intensity[j] - m2[i]) ** 2) * p[j])
    if (q2[i] == 0):
        continue
    else:
        v2[i] = temp / q2[i]

# within-class variance 계산
for k in range(size):
    w_v[k] = (q1[k] * v1[k]) + (q2[k] * v2[k])

k = np.argmin(w_v)
return k
```

함수이름	get_threshold_by_inter_variance ()
코드설명	이 함수는 “between-class variance” 방법으로 최적의 threshold 를 구하는 함수이다. 구현 수식은 이론자료의 moving average를 참고해서 구현했다. Between-class variance 를 구현할 때 구해야하는 값은 q1, m1, m2(평균) 이다. 그 이유는 q2 = 1 - q1 이고, between-class variance가 q1 * q2 * ((m1 - m2)^2) 로 정의되기 때문에 inter-class variance 방식에서는 분산이 필요가 없는 것이다. q1, m1, m2 을 구할때는 이론자료 수식을 참고해서 이전 값에서 변한만큼 추가 or 삭제를 해주는 방식으로 구해주었다. 그 후 between-class variance 를 위에서 언급한 수식으로 정의해주고, 이 between-class variance 의 최댓값이 inter-class variance 방식에서의 최적의 threshold 이므로 argmax 함수를 사용해서 threshold 를 구해주었다.
이미지	

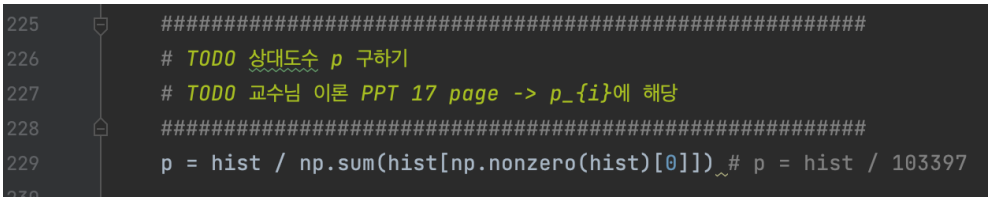
```
111     p += 1e-7  # q1과 q2가 0일때 나눗셈을 진행할 경우 오류를 막기 위함
112
113     # print(p)
114     size = len(p)
115     q1 = np.zeros(size)
116     m1 = np.zeros(size)
117     m2 = np.zeros(size)   # 평균 (m)
118     b_s = np.zeros(size)  # between-class variance
119     q1[0] = p[0]
120     m1[0] = 0
121
122     # m2 초깃값 설정
123     temp = 0
124     for i in range(size):
125         temp += i * p[i]
126     m2[0] = temp / (1 - q1[0])
127
128     for k in range(1, size):
129         q1[k] = q1[k-1] + p[k]
130         m1[k] = ((q1[k-1] * m1[k-1]) + (k * p[k])) / q1[k]
131         m2[k] = (((1 - q1[k - 1]) * m2[k - 1]) - (k * p[k])) / (1 - q1[k])
132
133     for k in range(size):
134         b_s[k] = q1[k] * (1 - q1[k]) * ((m1[k] - m2[k]) ** 2)
135
136     k = np.argmax(b_s)
137
138     # print("k: ", k)
139     return k
```

함수이름	get_hist ()
코드설명	이 함수는 mask 를 적용한 histogram 을 반환하는 함수이다. mask 의 값이 0인 경우, 우리는 그 값에 관심이 없으므로 픽셀의 빈도수를 세지 않는다. 즉 0으로 값을 넣어준다. (이 과정을 처음에 반환할 hist 배열을 만들 때 zero 로 만들어주었으므로 continue 로 구현해주었다.) mask 의 값이 0 이 아닌 경우, 우리는 이 값에 관심이 있으므로 src[row][col] 의 값을 index 로 넣어주어 반환할 hist 의 값을 1 추가시켜주도록 함수를 구현하였다. (src[row][col] 을 index 로 생각하는 이유는 hist 는 각각의 픽셀이 나오는 빈도를 저장하고있는 배열이기 때문이다)
이미지	

```python
def get_hist(src, mask):

    """
    :param src: gray scale 이미지
    :param mask: masking을 하기 위한 값
    :return:
    """

    ###########################################################################
    # TODO mask를 적용한 히스토그램 완성
    # TODO mask 값이 0인 영역은 픽셀의 빈도수를 세지 않음
    # TODO histogram을 생성해 주는 내장함수 사용금지. np.histogram, cv2.calHist
    ###########################################################################
    hist = np.zeros((256,))

    h, w = src.shape
    # print(h) 432
    # print(w) 512

    # count = 0
    # ???
    for row in range(h):
        for col in range(w):
            if (mask[row][col] == 0):
                continue
            else:
                hist[src[row][col]] = hist[src[row][col]] + 1
                # count += 1

    # print("count : ", count)
    return hist
```

함수이름	threshold ()
코드설명	이 함수는 mask 값이 0이 아니면서 src의 값이 0 ~ threshold 사이의 값이라면 반환할 dst라는 배열의 값을 255로 설정하고, 이 이외의 값이라면 0으로 설정해 반환하는 함수이다. (threshold 적용)
이미지	

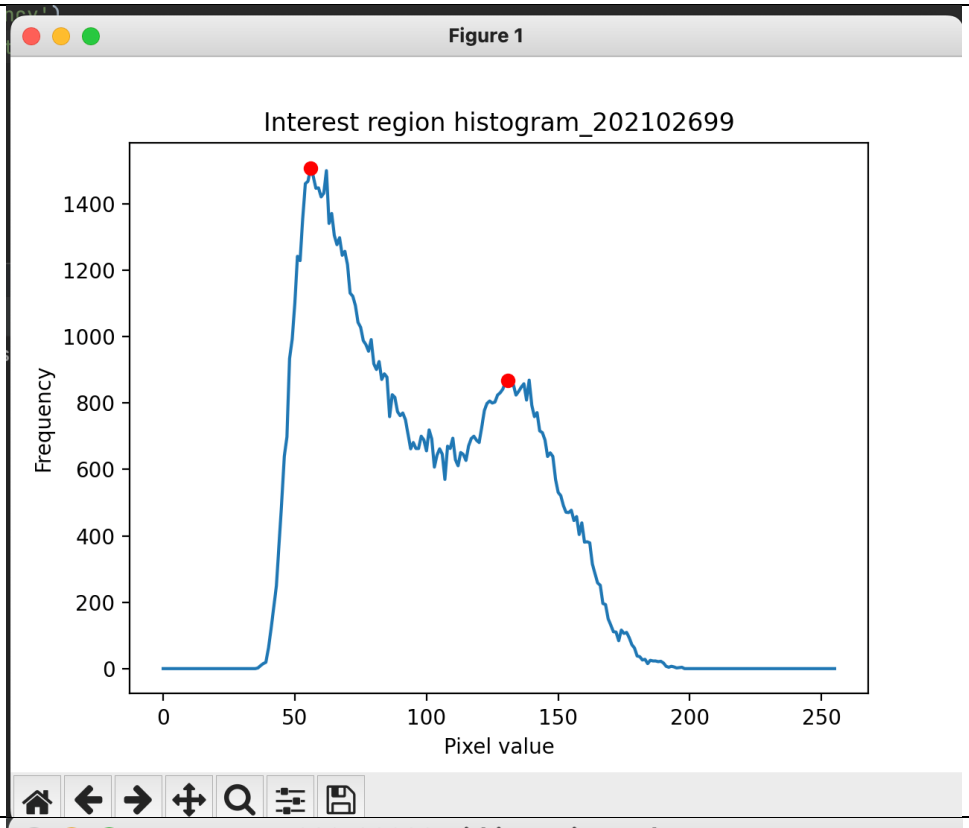
```python
def threshold(src, threshold, mask):
    """
    :param src: gray scale 이미지
    :param threshold: threshold 값
    :param mask: masking을 하기 위한 값
    :return:
    """

    ########################################################
    # TODO threshold 값을 이용한 이미지 값 채우기
    # TODO 0 < src <= threshold 이면 255로 채움
    # TODO mask의 값이 0인 좌표에 대해서는 값을 0으로 채움
    # TODO 이외의 영역은 모두 0으로 채움
    # TODO cv2.threshold 사용금지
    ########################################################

    h, w = src.shape
    dst = np.zeros((h, w), dtype=np.uint8)

    # ???
    for row in range(h):
        for col in range(w):
            if (mask[row][col] != 0) :
                # 0 < src <= threshold 인 경우
                if(0 < src[row][col] and src[row][col] <= threshold):
                    dst[row][col] = 255
            else: # mask 의 값이 0 인 경우
                dst[row][col] = 0

    return dst
```

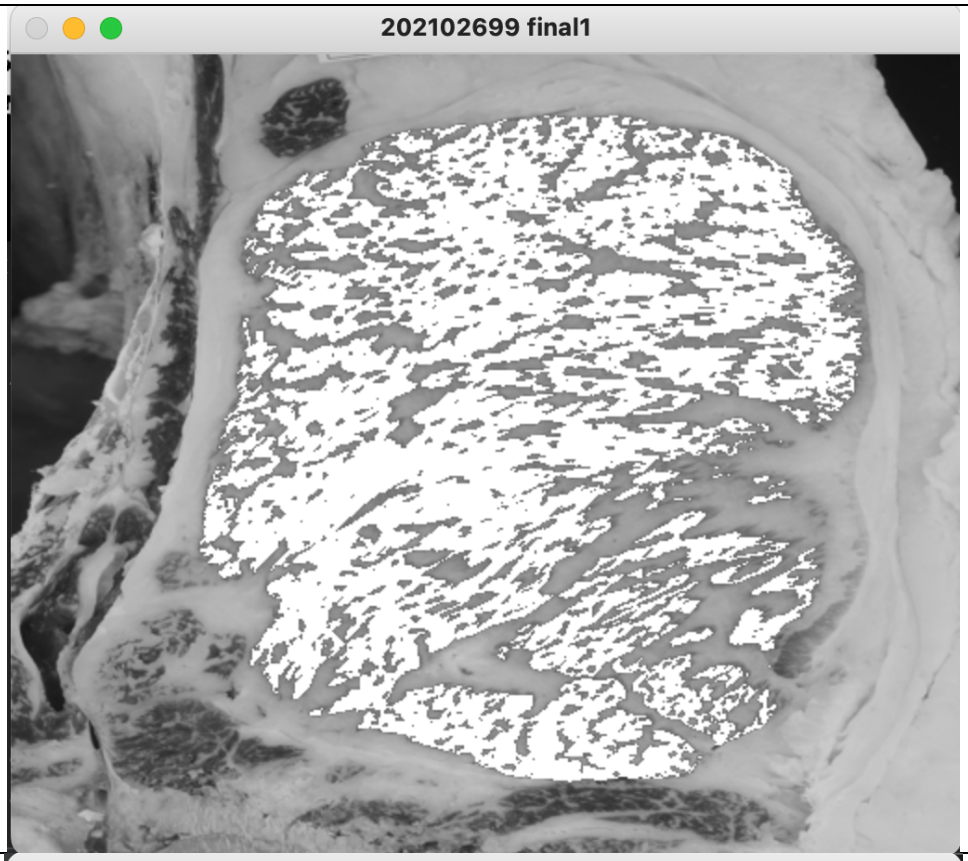
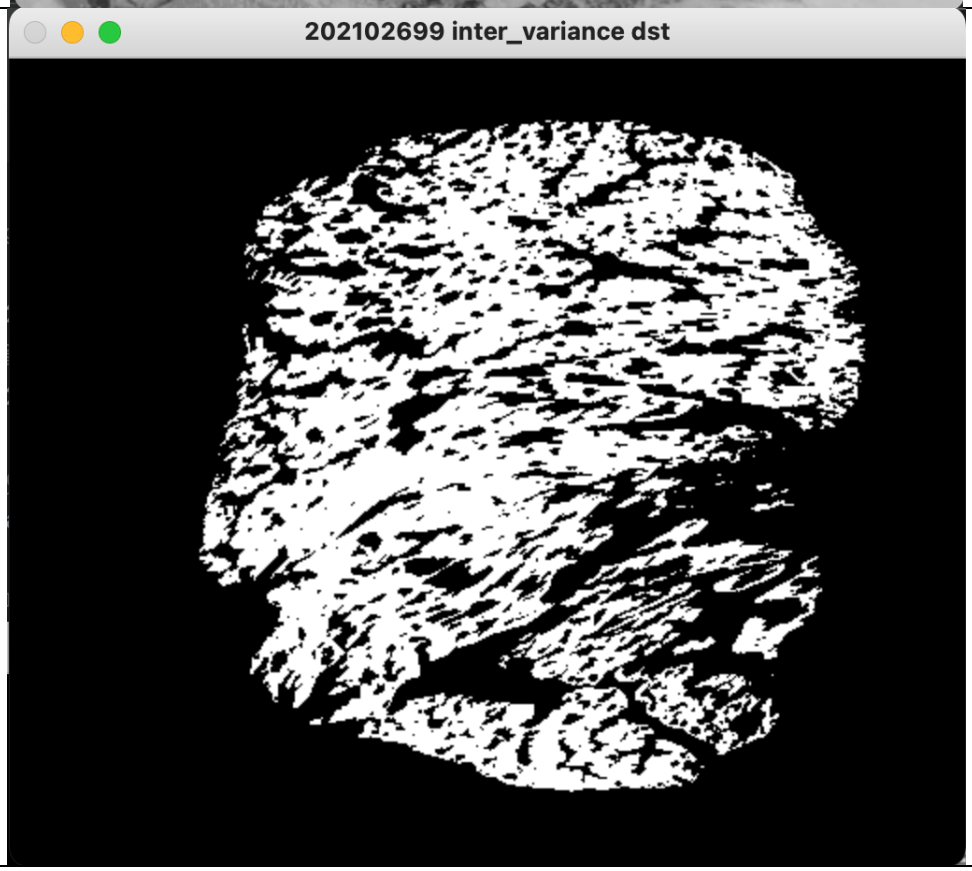
함수이름	otsu_method ()
코드설명	이 함수에서는 상대도수 p 를 구하는 수식과, 그래프에서 두개의 peak 좌표에 점을 찍는 함수를 완성시키는 함수이다. 먼저 상대도수 p 를 구하는 방법은 hist 를 masking 이 된 hist 에서 전체 픽셀로 나눈 값이다. np.sum(hist) 로 나눠준다면 전체 이미지 사이즈에 대한 상대도수이므로 안되고, 우리가 masking 을 진행했기 때문에 0이 아닌 값 즉, 우리가 관심있는 값의 pixel 수로 나눠줘야 masking 이 진행된 이미지 사이즈에 대한 상대도수를 구할 수 있다. 두번째로 그래프에서 두개의 peak 좌표에 점을 찍는 함수에서는 x, y 축의 위치를 순서대로 넣어주면 완성된다. 우리가 위에서 구한 threshold 를 기준으로 왼쪽에서 가장 큰 값, 오른쪽에서 가장 큰 값의 x, y 좌표를 구하면 된다. 내가 구현한 방식은, np.argmax 함수를 사용해서 hist[0 ~ threshold] 까지의 값 중 가장 큰 값의 Index 를 구해주었고, 이 값을 hist 의 index 로 넣어주게 되면 y 축의 좌표도 구할 수 있게 된다. 두번째 좌표 역시 이와 같은 방법으로 진행하게 되면 구할 수 있는데 이때 np.argmax 함수의 인자는 0~ threshold 범위가 아닌 threshold 를 기준으로 오른쪽에서 max 값을 찾아야 하기 때문에 threshold ~ 끝까지의 범위로 탐색해서 구현해주었다.
이미지	 

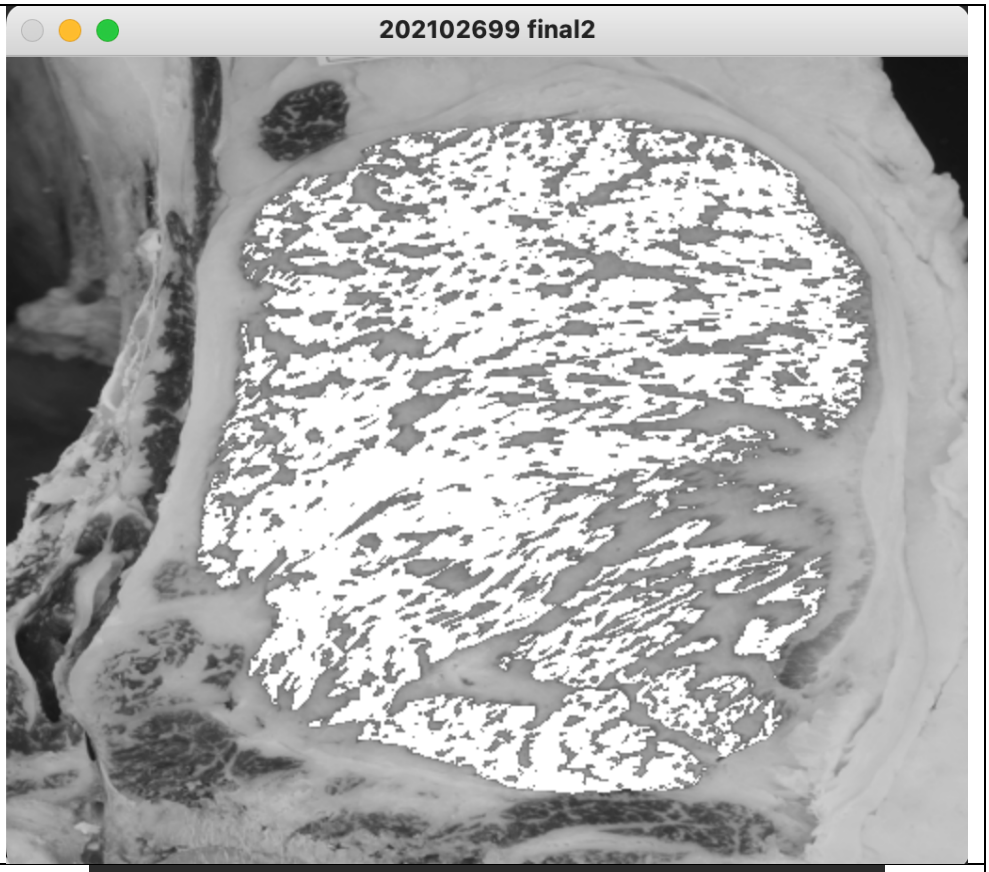
```
225     ############################################################
226     # TODO 상대도수 p 구하기
227     # TODO 교수님 이론 PPT 17 page -> p_{i}에 해당
228     ############################################################
229     p = hist / np.sum(hist[np.nonzero(hist)[0]])  # p = hist / 103397
```

```
247     ######################################################
248     # TODO Bimodal histogram 완성
249     # TODO 2개의 peak에 해당하는 픽셀 값에 점 찍기
250     # TODO 보고서에 결과 이미지 첨부
251     ######################################################
252     # Bi-modal Distribution
253     plt.plot(intensity, hist)
254     plt.plot(np.argmax(hist[:k1]), hist[np.argmax(hist[:k1])], color='red', marker='o', markersize=6)
255     plt.plot(np.argmax(hist[k1:]) + k1, hist[np.argmax(hist[k2:]) + k1], color='red', marker='o', markersize=6)
```

2. 결과 이미지

Histogram 에서 2개의 peak 좌표에 점 찍기	
Within-class-variance dst	

Final image 1	
Inter-class variance dst	

Final image 2	
구현 비교 결과	2가지 방식 결과 비교 : True Process finished with exit code 0