수치해석 HW14

20180086 13 안상욱

1. Overview

Mean shift, k-mean 두 가지 방법 모두 먼저 그림을 읽어온 뒤 rgb 정보를 Lab값으로 변환했습니다.

변환한 Lab 값을 가지고 clustering 한 뒤 그 결과를 다시 RGB값으로 변환한 뒤 출력해주었습니다.

먼저, mean shift방법은 cluster 의 수가 적당히 나올 수 있는 h 값을 찾기 위해 h 값을 조정해 가면서 여러 번 코드를 실행해 보았습니다.

K-mean 방법은 k의 값을 5, 15, 25로 설정해서 실행해 본 뒤 결과를 비교해 보았습니다.

2. 그림

다음과 같은 세 그림으로 clustering을 진행해 보았습니다. 그림은 순서대로 picture1, picture2, picture3이라는 이름으로 저장했습니다.

3. K-mean 코드

먼저, K-mean의 코드입니다.

그림을 읽어와서 rgb라는 변수에 rgb값을 저장해 주었습니다. 그리고 이 값을 Lab로 변경해서 lab라는 변수에 저장해 주었습니다.

그 후 clustering할 수 있도록 배열의 모양을 바꾸어서 Z라는 변수에 저장해주었습니다.

K값을 지정한 뒤 clustering 을 한 값을 res에 저장시켜주고 원래의 lab 배열과 같은 모양으로 reshape해준 뒤 다시 lab 를 rgb 로 바꾸어서 이미지를 출력해주었습니다.

K값을 5, 15, 25 일때 모두 출력해서 비교해보았습니다.

```
rgb = io.imread("picture1.jpg")
lab = color.rgb2lab(rgb)

originShape = lab.shape
Z = lab.reshape((-1,3))

#k-means
Z = np.float32(Z)
criteria = (cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER, 10, 1.0)
K = 15
ret,label,center=cv2.kmeans(Z,K,None,criteria,10,cv2.KMEANS_RANDOM_CENTERS)

res = center[label.flatten()]
res2 = res.reshape((lab.shape))

result = color.lab2rgb(res2)

plt.imshow(result)
plt.show()
```

4. Mean shift 코드

앞의 방법과 같이 rgb 값을 lab 값으로 바꾸어 준 뒤 reshape해서 Z라는 변수에 넣어주었습니다.

H값을 임의로 정한 뒤 mean shift를 이용해 clustering해준 후의 배열을 result에 담았습니다. 그 후 result의 lab값을 rgb로 바꾸어 준 뒤 그림을 출력해 주었습니다. 그리고 만들어진 cluster의 개수도 출력해 주었습니다.

h 값 (bandwidth)를 적절한 수의 cluster가 나오도록 조정해가면서 이미지를 출력해 보았습니다.

```
from skimage import io, color
import matplotlib.pyplot as plt
import numpy as np
import cv2
from sklearn.cluster import MeanShift
from sklearn.datasets import make_blobs

rgb = io.imread("picture1.jpg")
lab = color.rgb2lab(rgb)

originShape = lab.shape
Z = lab.reshape((-1,3))

#mean-shift
ms = MeanShift(bandwidth = 7)

ms.fit(Z)

labels=ms.labels_
print(labels)

cluster_centers = ms.cluster_centers_


labels_unique = np.unique(labels)
n_clusters_ = len(labels_unique)
print("number of estimated clusters : %d" % n_clusters_)

result = []
for i in labels:
    result.append(cluster_centers[i])
result = np.reshape(result, originShape)
result = color.lab2rgb(result)

plt.imshow(result)
plt.show()
```

5. Picture1 실행 결과

맨 처음의 사진은 원본 사진이고, 순서대로 K-mean의 K값이 5, 15, 25일때, mean shift를 사용했을 때의 사진입니다. Mean shift를 할 때 h 의 값을 7로 설정했을 때 cluster의 수가 13개가 나옴을 확인해 볼 수 있었습니다.

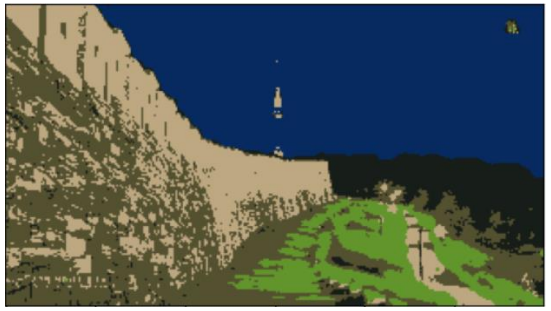
```
ms = MeanShift(bandwidth = 7)
number of estimated clusters : 13
```

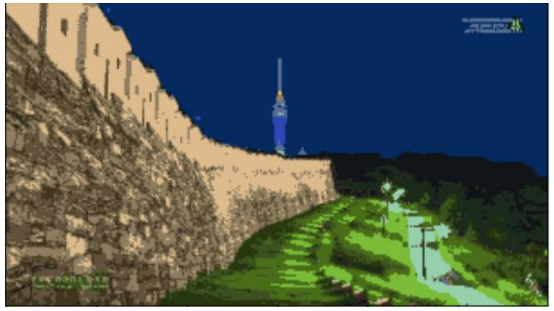
6. Picture2 실행 결과

맨 처음의 사진은 원본 사진이고, 순서대로 K-mean의 K값이 5, 15, 25일때, mean shift를 사용했을 때의 사진입니다. Mean shift를 할 때 h 의 값을 7로 설정했을 때 cluster의 수가 80개가 나옴을 확인해 볼 수 있었습니다.

```
ms = MeanShift(bandwidth = 7)

number of estimated clusters : 80
```





7. Picture3 실행 결과

맨 처음의 사진은 원본 사진이고, 순서대로 K-mean의 K값이 5, 15, 25일때, mean shift를 사용했을 때의 사진입니다. Mean shift를 할 때 h 의 값을 7로 설정했을 때 cluster의 수가 24개가 나옴을 확인해 볼 수 있었습니다.

```
ms = MeanShift(bandwidth = 7)
```

```
number of estimated clusters : 24
```

8. 결과 분석

K-mean 방법에서는 K 값이 클수록, 즉 cluster의 수가 많을수록 원본 사진에 가까운 선명한 사진이 나오는 것을 확인해볼 수 있었습니다. K값이 5일때보다 15일 때가, 15일 때보다 25일 때 더 선명하고 원본에 가까운 이미지가 나오는 것을 확인해 볼 수 있었습니다.

Mean shift 방식에서는 h 값 즉, bandwidth 가 작을수록 더 많은 cluster 가 나와 원본과 유사한 이미지를 얻을 수 있고, h값이 클수록 cluster 의 수가 적어져서 덜 선명한 이미지가 나오는 것을 확인해 볼 수 있었습니다. 코드를 실행해 볼 때 bandwidth의 값을 1.5 정도로 실행해 보았는데 cluster의 개수가 4000개 가량으로 너무 많이 나오는 것을 확인해 볼 수 있었고, 너무 크게 하면 cluster의 수가 0으로 나와 올바른 결과가 나오지 않았습니다. 그래서 그 사이의 값으로 적절히 조율해서 Mean shift를 실행했습니다.

K-mean과 mean shift 방식을 비교해 보았을 때 cluster 의 개수가 비슷할 때 K-mean의 그림이 조금 더 선명하게 나오는 것을 확인해 볼 수 있었습니다.