영상처리

PROJECT 2

과목	영상처리
이름	차원범
학번	2016124217
교수님	오병태 교수님
제출일	20.05.24

1. Image Clustering

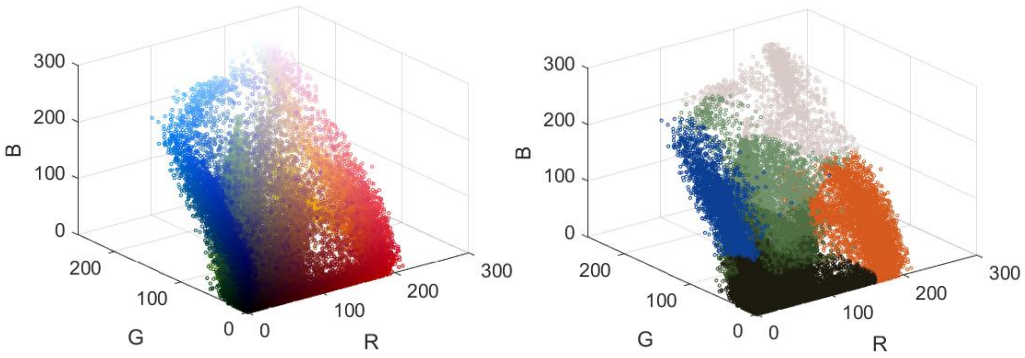
A. K-means

K-means는 주어진 공간에 임의의 k를 생성해 그 주위를 클러스터로 묶는다. 이후 자신의 평균을 이동시켜 가면서 클러스터의 중심으로 향하며, 이후 이 k를 대표값으로 이미지를 클러스터링 할 수 있다.

1) K-means in R, G, B space

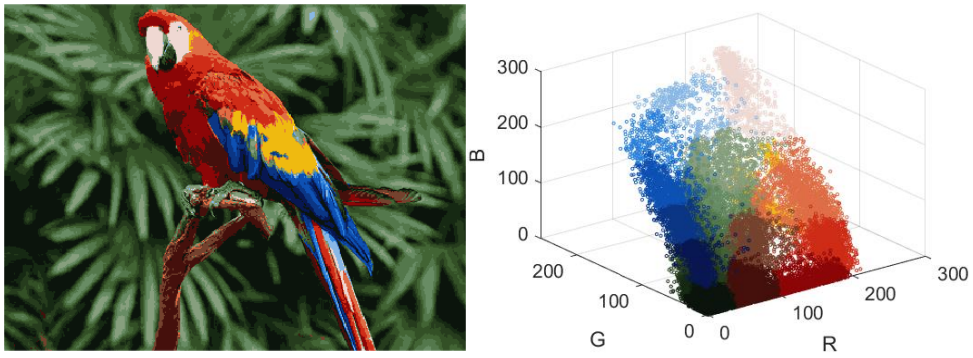
➢ 원본 이미지(좌) 와 K-means를 시행한 이미지 (우), **(*k = 7*)**

RGB 색영역에서 k-means를 시행한 결과이다. R, G, B영역에 임의의 k를 생성하고, 가장 가까운 k로 옮겨가는 과정을 거친다. 클러스터링을 통해 이미지에서 강조되고 있는 부분을 더욱 쉽게 알 수 있으며, 이미지의 경계 부분을 더욱 쉽게 볼 수 있다.



➢ 원본 이미지(좌) 와 K-means를 시행한 후(우)를 색공간으로 표현, *(k = 7)*

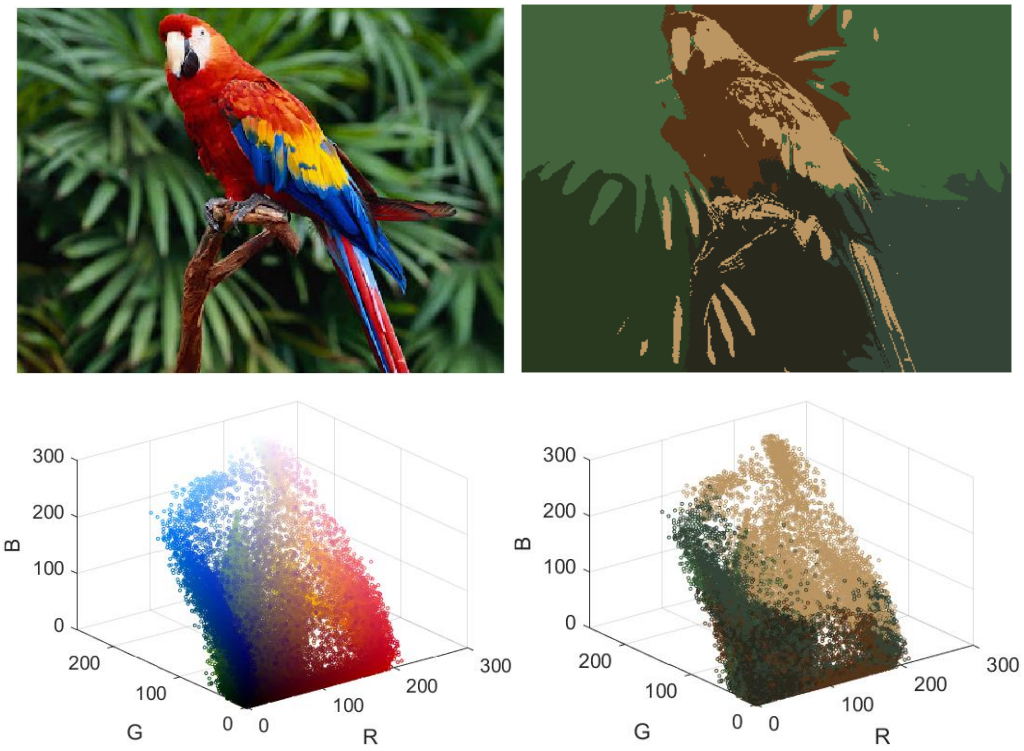
각각의 색공간을 3차원으로 표현하여 클러스터링이 정확하게 되었는지 확인하였다. 클러스터링 하기 전의 점들의 좌표에 최종적으로 도출된 k값을 대입하였다.



➢ 20의 값을 가지는 k로 k-means를 시행하고, 이를 표현한 이미지와 색공간

k개의 개수를 늘릴수록 원본 이미지와 비슷한 표현력을 가지는 것을 확인할 수 있었다. 또한 색공간으로 표현한 부분을 보면, 데이터의 밀도가 많은 cluster가 생성되고, 밀도가 낮은 부분이 더 적은 cluster가 생성이 되는 것을 볼 수 있었다. 따라서 이미지에서 많이 차지하고 있는 색이 더 섬세하게 표현되어 20개 정도의 cluster만으로도 원본 이미지와 비슷하게 보이도록 하는 것을 볼 수 있었다.

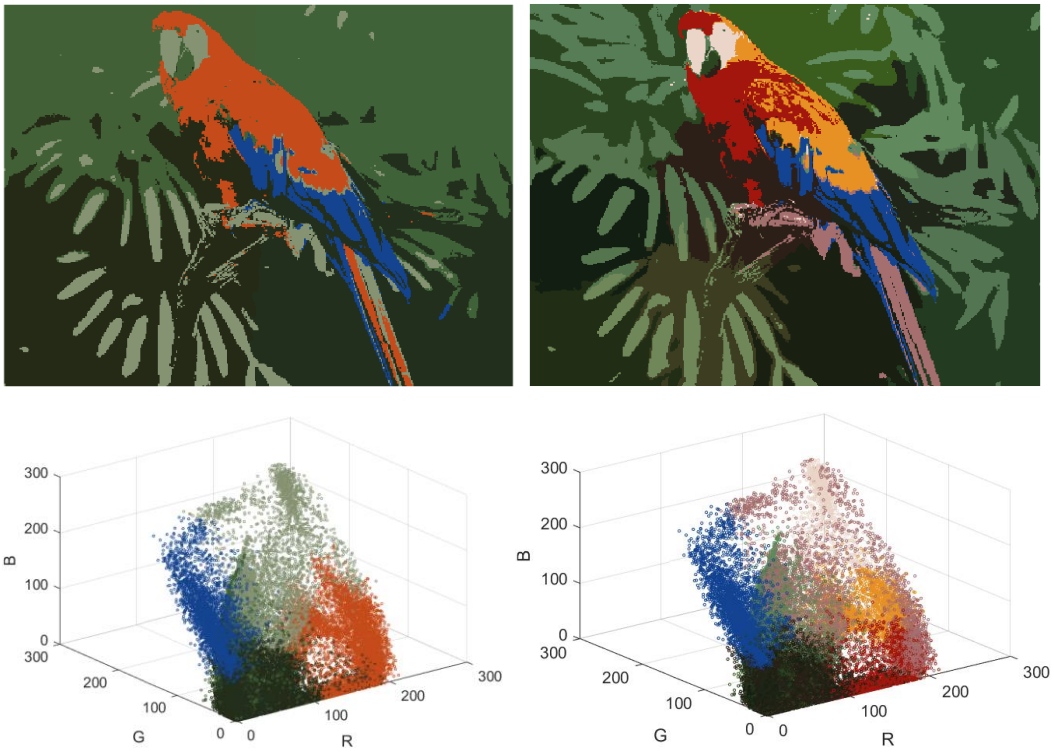
2) K-means in R, G, B, x, y space



➢ 원본 이미지(좌) 와 K-means를 시행한 이미지 (우), *(k = 7, 거리 가중치=1)*

x, y를 포함하여 clustering 한 결과는 포함하지 않은 결과와 판이하게 다른 것을 볼 수 있었다. 각 cluster는 색뿐만 아니라 거리까지 생각하기 때문에, 해당 cluster 주변의 색 위주로 고려한다. 따라서 위의 이미지의 경우, 중앙의 앵무새를 제외하고는 대부분 초록색이기 때문에 초록

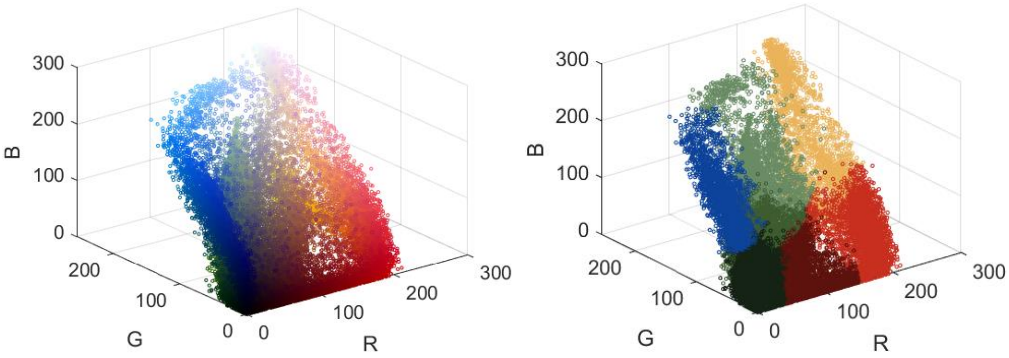
색이 대다수인 출력을 가지게 되었다.



➢ k = 7로 clustering한 이미지(좌), k = 20로 clustering한 이미지(우) *(거리 가중치 = 0.5)*

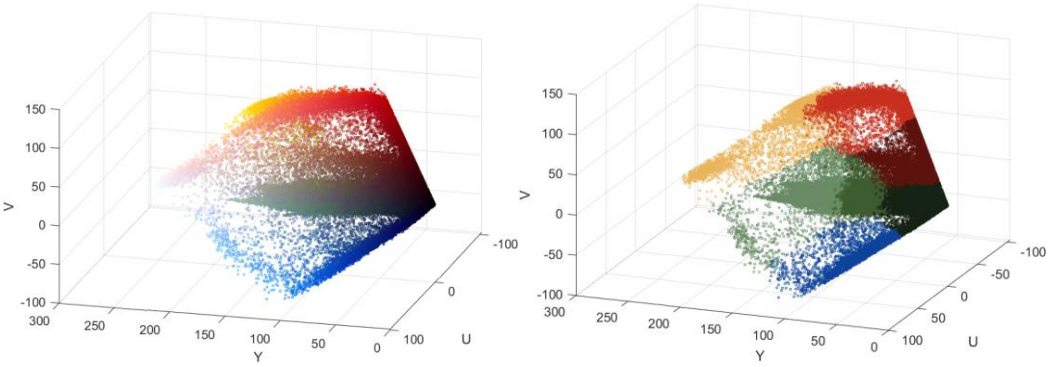
k-means를 시행하면서 거리 가중치를 수정할 수 있다. 이는 각 k가 가지는 R, G, B, x, y 값들과 클러스터 간의 거리를 계산할 때, x, y의 거리에 거리 가중치를 곱해 손쉽게 조정할 수 있다. 거리 가중치를 줄인 결과, 초록색이 과도하게 이미지를 침범하는 것을 줄이며, 적은 k로도 앵무새를 손쉽게 구별할 수 있었다. 색의 대비를 보다 쉽게 확인 가능하였으며 이미지에서의 중요 경계를 파악하는 데 더욱 손쉽게 확인할 수 있게 되었다.

3) K-means in Y, U, V space



➢ 원본 이미지(좌) 와 K-means를 시행한 이미지 (우), (*k = 7*)

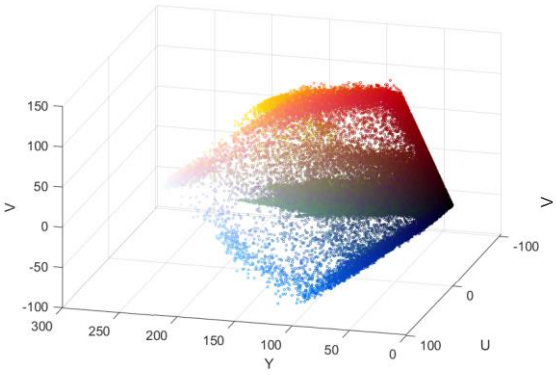
YUV영역에서 k-means를 시행한 결과, RGB영역에서 시행한 것과는 다른 결과를 확인할 수 있었다. R,G,B를 각 축으로 3차원으로 나타낸 그래프에서는 그 원인을 파악하기 어려웠지만 다음과 같이 Y,U,V를 각 축으로 나타낸 그래프에서는 확인하기 쉬웠다.

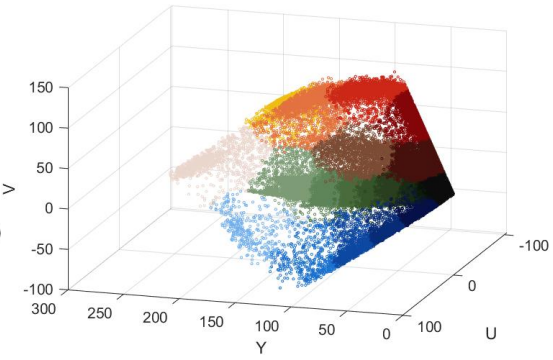


➢ 원본을 YUV 색공간으로 표현(좌) 와 K-means를 시행 한 후 표현(우), (*k = 7*)

YUV색공간으로 바꾸어 보면 RGB에서 보다 덜 규칙적으로 clustering이 표현되는것과는 별개로 보다 규칙적이고, 밀도가 높은 부분에서 적극적으로 clustering이 되는 것도 확인할 수 있었다. YUV색공간에서는 1개의 밝기 정보와, 2개의 색 정보를 가지고 있으므로, 밝기에 대해서 이전보다 섬세하게 clustering이 되는 것을 확인할 수 있었다.







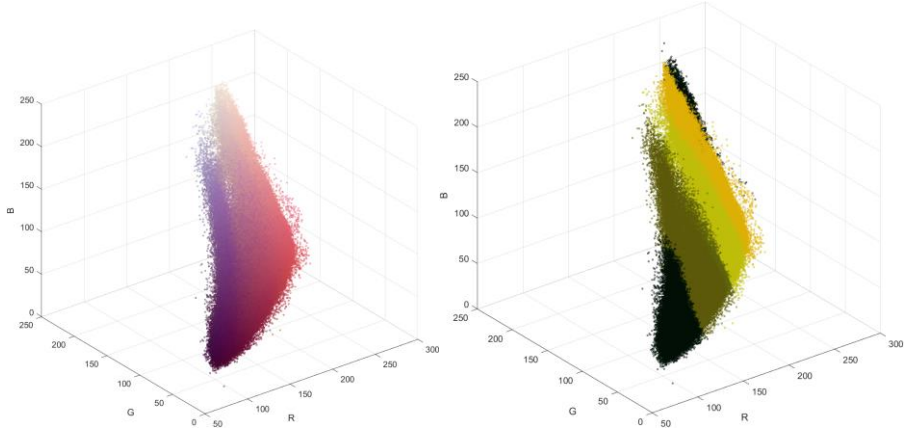
➢ 20의 값을 가지는 k로 k-means를 시행하고, 이를 표현한 이미지와 색공간

20 개의 k 로 이미지를 YUV 영역에서 clustering 한 결과는 RGB 영역에서 clustering 한 결과와 크게 다르지 않았다. 이를 통해 k 가 커질수록 색공간에 따른 차이가 적어진다는 걸 확인할 수 있었다.

B. Mean shift

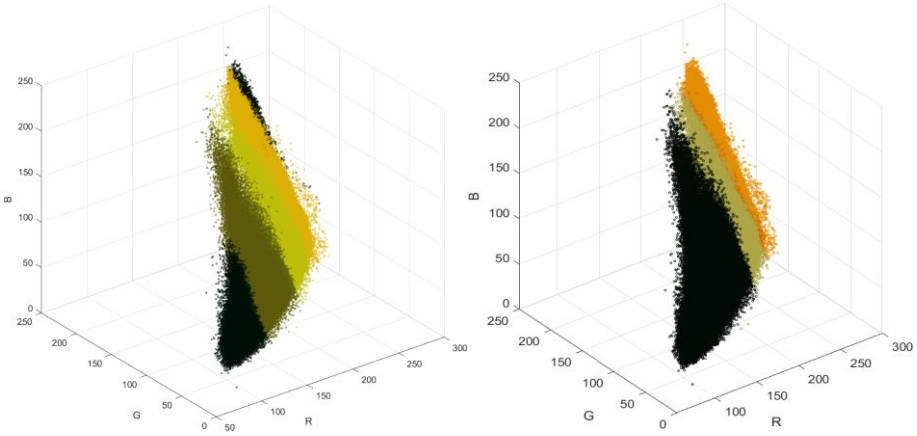
Mean shift는 영상 히스토그램의 모든 영역에서 시작한다. 처음 주어진 영역 크기 안에서 자신의 평균을 구하고, 다시 자신을 평균의 중간으로 이동시킨다. 이 과정을 반복하여 더 이상 평균이 움직이지 않을 때까지 반복한다. K-means에 비해 k의 개수를 정할 필요가 없기에 보다 발전된 방법이라고 볼 수 있으며, 영역의 평균을 구하는 과정을 바꿀 수 있기에 보다 유리한 점이 많다고 할 수 있다.

1) Mean shift in R, G, B space



➢ 입력 이미지(좌), meanshift 한 이미지(우), *(window size = 20)*

주어진 R, G, B 영역에서 히스토그램을 계산하여 3 개 히스토그램의 Gradient 가 최대가 되는 방향을 찾는다. 하지만 window 간의 비교가 제대로 이루어지지 않아 성공적인 결과를 가져오지는 못하였다.



➢ 평균을 사용한 이미지(좌), Gaussian kernel 을 사용한 이미지(우)

Mean 을 구하는 과정에 Gaussian kernel 을 더하여, 중간쪽의 값을 더 사용하도록 하였다. 비록 성공적인 결과는 아니지만, 사용하지 않았을 때와 비교해서 보다 많은 색을 표현하는 것을 볼 수 있다. 이로 미루어 보아 Cluster 의 숫자를 늘리는 데 유용하게 사용할 수 있을 것이라고 추정해볼 수 있다.

2. EigenFace

1) EigenFace to each person

Eigenface를 구하기 위하여 테스트 데이터셋의 모든 사람들의 얼굴을 평균 낸다. 이후 이 평균 낸 값을 이용해 모든 데이터셋에 빼서 정규화를 실시한다. 이 정규화한 데이터셋 집합으로 공분산 행렬을 구한 후, 이의 고유벡터와 고유값을 추출한다. 고유값이 가장 높게 나오는 벡터가 분산을 최대로 하는 값이며, 이들 벡터를 몇 개 이용해 고유얼굴을 추출할 수 있다.

➢ 평균 얼굴(좌) 와, 고유얼굴을 구하기 위한 두 개의 사진(우측 2개)

➢ 왼쪽부터 3개, 10개, 500개, 3000개, 6000개의 고유벡터를 사용하여 만든 고유 얼굴

위의 실험에서 고유벡터는 고유값이 가장 크게 나오는 순서대로 고유벡터를 이용하였다. 3개의 벡터에만 사영시켰을 때에는 두명의 얼굴의 분간이 거의 불가능하다. 하지만 10개의 데이터를 사용하였을 때에는 턱수염 등, 보다 큰 차이를 확인할 수 있었다. 이후 고유벡터의 수가 증가할수록 원본 이미지와 비슷하게 출력되는 것을 확인할 수 있다.

2) Identify each person

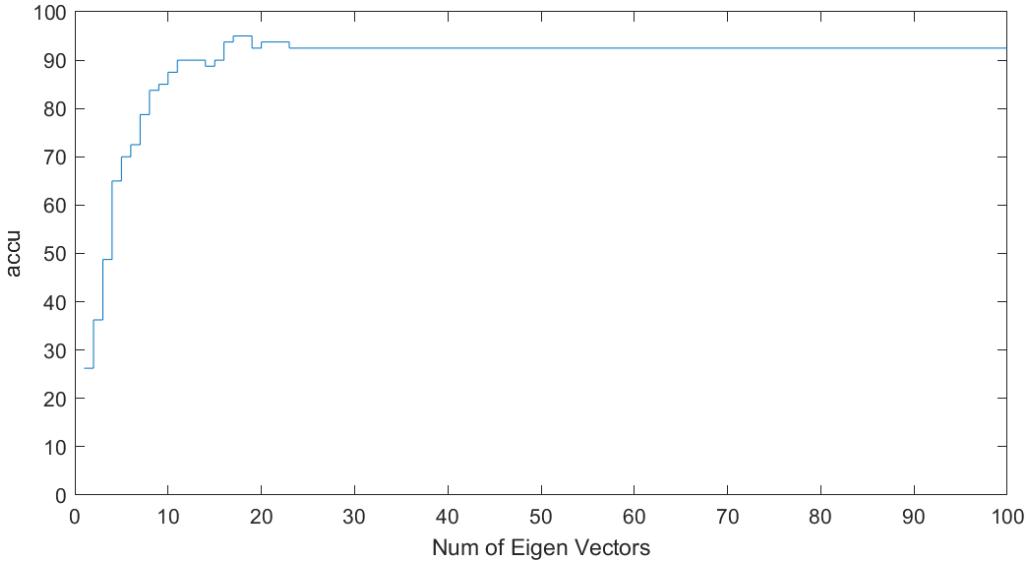
주어진 데이터셋에는 40명의 사람이 각각 10개의 표정, 구도 등을 가지고 있다. 여기서 8개는 훈련용 데이터셋으로 분리하여 고유벡터를 구하는 데 사용한다. 이후 남은 2개로 테스트를 진행하였다. 앞서 테스트에 쓰인 8개의 훈련용 데이터셋은 각 사람별로 8개의 값을 평균을 내어 총 40개의 훈련용 데이터셋으로 축소하여 진행하였으며 고유벡터는 100개를 이용하였다.

➢ 입력 이미지(데이터셋 안의 5번째 사람의 9번째 이미지)

Output :

Top 3 data: 5 10 35
Distance: 122.21 287.55 292.57

컴퓨터가 추론한 상위 3개의 사람 후보. 이 중 가장 가깝다고 추론한 사람은 5번째 사람이었다. 이는 실제 우리가 입력 이미지로 넣었던 사람이 5번째 사람이었으므로 맞는 추론이라고 볼 수 있다. 3개의 추론 값에서 거리를 비교해 보았을 때에도 5번 데이터셋과의 거리가 122.21로, 2,3번째로 가까운 데이터셋과의 거리(287.55, 292.57)보다 월등히 가까우므로 정확하게 추론했다는 것을 볼 수 있다.



생성된 고유 벡터를 기반으로 0개부터 100개 까지의 벡터를 이용하여 테스트해보았다. 10개의 고유벡터를 사용 시 약 80%의 정확도를 보여주었다. 이후 꾸준히 상승하여 20개 부근에서 95%에 가까운 정확도이며, 30개 이후부터는 조금 낮아져 약 92%의 정확도를 가지는 것을 확인하였다.

3) Best performance

앞서 나온 그래프에서 20개의 고유벡터에 사영시켰을 때에는 95%의 정확도가 나왔으나, 이후에는 조금 감소하여 92% 정도의 정확도가 나왔다. 이는 Overfitting문제이며, 훈련 데이터에 과적합되어 나타나는 현상이다. 20개의 고유벡터를 사용했을 때에는 정답이었지만, 100개의 고유벡터를 사용했을 때에는 오답인 얼굴과 그 고유얼굴을 아래에 표시하였다.

100개의 고유 벡터를 사용하였을 때, 컴퓨터가 오답으로 추론한 얼굴. 컴퓨터는 위 고유벡터를 기반으로 20번째 사람이라고 예상하였다. 하지만 정답은 8번째 사람이었다.

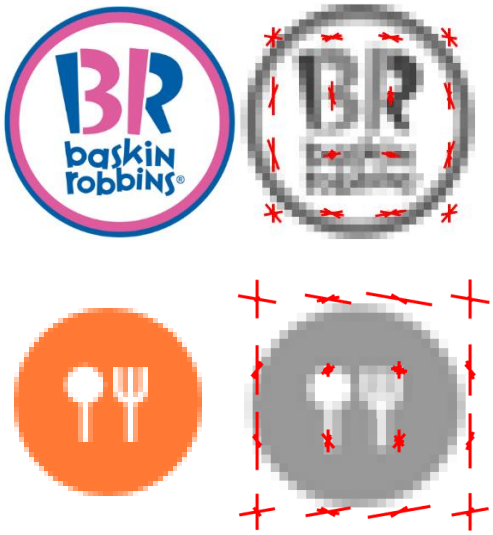
- 실제 정답인 고유얼굴(좌), 컴퓨터가 추론한 20번째 사람의 고유얼굴(우) *(d = 100)*

- 실제 정답인 고유얼굴(좌), 앞서 컴퓨터가 추론한 20번째 사람의 고유얼굴(우) *(d = 20)*

과적합으로 인해 조명이나 표정의 변화에도 제대로 인식을 하지 못하는 상황이 발생하였다. 따라서 과적합이 최대한 발생하지 않는 20개 전후의 차원으로 정사영 시켰을 때가 가장 높은 성능을 가지는 것으로 추론할 수 있다. 실제로 위의 이미지 상에서도 20차원에서의 구별력이 조금 더 높은 것을 볼 수 있다.

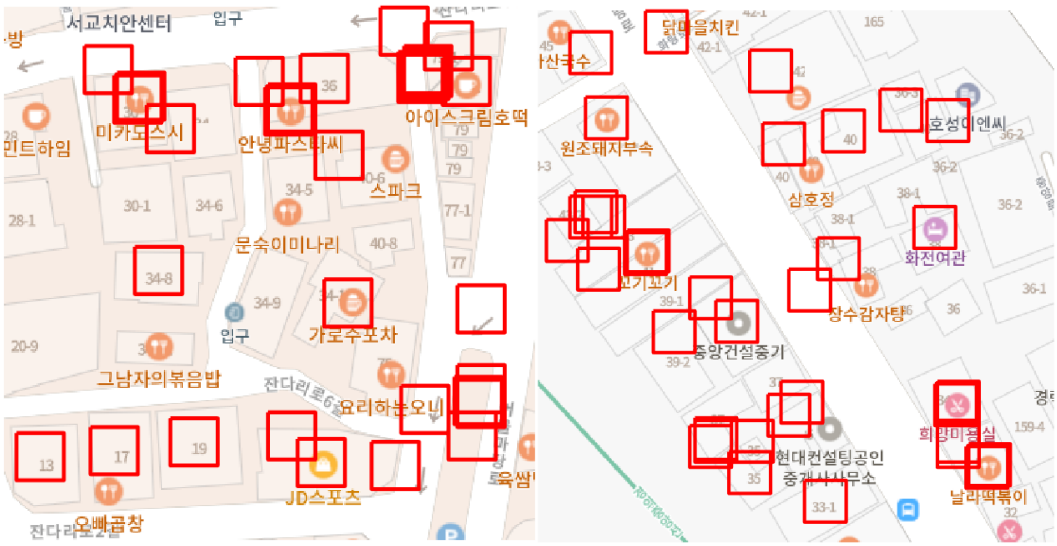
3. Local feature descriptor

1) Histogram of oriented gradient

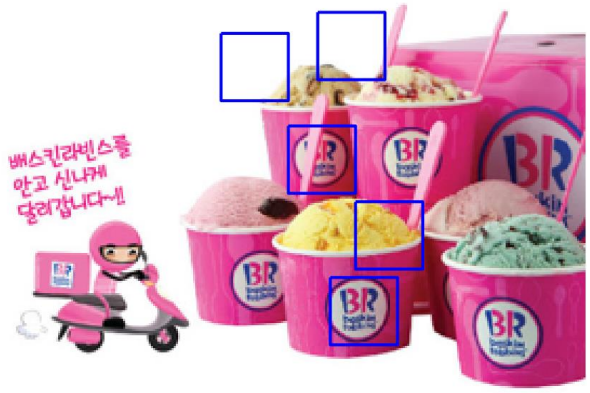


➢ 쿼리 이미지(좌)와, 이에 대해 32*32로 resize 후 HoG를 시행한 이미지(우)

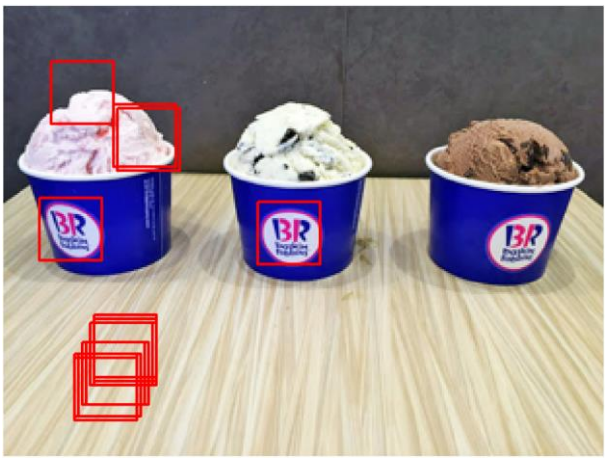
Local feature descriptor을 이용해서 주어진 로고의 histogram of oriented gradient를 구하였다. 이렇게 생성된 gradient 값을 토대로 Target 이미지의 gradient와 비교해 그 차이가 적다면 로고를 찾은 것으로 간주할 수 있다.



➢ 쿼리 이미지를 바탕으로 찾아낸 로고, (이미지1(좌), 이미지2(우)), *(threshold = 0.002)*



➢ 쿼리 이미지를 바탕으로 찾아낸 로고, (이미지3), *(threshold = 0.01, 0.55배 축소)*

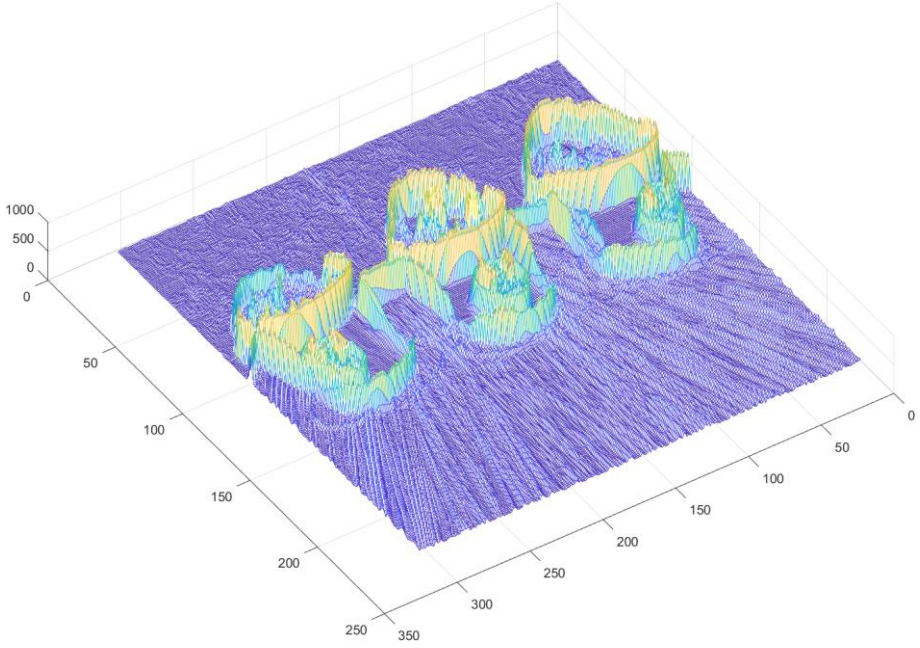


➢ 쿼리 이미지를 바탕으로 찾아낸 로고, (이미지4), *(threshold = 0.03, 0.6배 축소)*

지도에서 검출한 이미지인 첫번째 이미지를 보면, 찾고자 하는 식당 로고를 모두 찾지 못한 것을 볼 수 있다. 또한 비슷하게 생긴 로고가 많아, feature vector로는 제대로 된 성능을 내지 못하는 것을 볼 수 있었다.

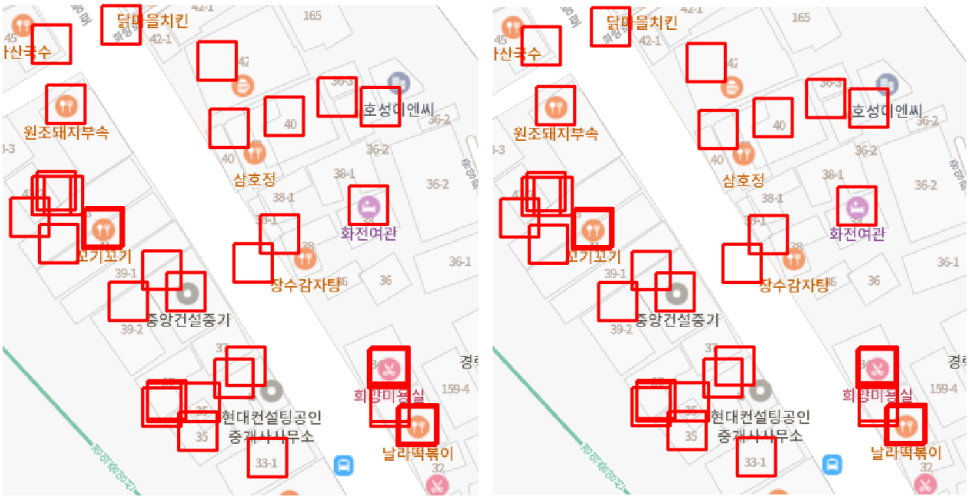
배스킨라빈스 로고를 이용해 찾은 경우에서는, 식당 로고의 경우와 다르게 False positive의 비율이 더 적었다. 그 이유는 지도와 다르게 비슷하게 생긴 로고가 적었으며, 로고의 복잡도 또한 올라간 덕분으로 보인다. 하지만 사진마다 로고의 크기가 달라 사진을 미리 resize해주는 과정이 필요하였다. 또한 로고가 가려져 있거나, 기울어져 있으면 제대로 찾지 못하는 경우가 많았다.

2) Improve Local feature descriptor with gradient



➢ 위 아이스크림 이미지의 Gradient를 나타낸 그래프

우리가 찾고자 하는 것은 로고이므로, 주변의 값들과 확연하게 다른 값을 가질 것이다. 이를 다르게 말하면 주변에 비해 Gradient가 높다고 볼 수 있을 것이다. 따라서 검출을 하기 전에 Gradient가 낮은 부분을 미리 제거하여 로고 검출에 이용할 수 있다면 보다 빠르고 정확한 검출이 가능할 것이다.



➢ 기존 방법(좌)와, 개선된 방법(우)로 로고를 찾은 결과 *(threshold = 10,000)*

지도 이미지의 경우는 개선한 알고리즘으로는 정확도를 개선하기 어려웠다. 하지만 개선 후 함수의 시행시간은 21.695초에서 12.094초로 44.3%의 개선을 달성하였다.

➢ 기존 방법(좌)와, 개선된 방법(우)로 로고를 찾은 결과 *(threshold = 14,000)*

지도 이미지가 아닌 실물 사진의 경우는 정확도 개선 또한 이루어졌다. 왼쪽 아래 부분의 패턴은 찾고자 하는 로고가 아니며 단순한 원목 패턴이다. 이곳의 Gradient는 낮다고 볼 수 있으며, Gradient가 낮은 부분을 생략하고 계산하는 알고리즘에서 효과적인 개선을 보였다. 시행시간은 9.57초에서 4.87초로 49.8%의 개선을 달성하였다.

	시간 개선	**정확률 (Presision)**	
		기존	개선 후
이미지1	43.3%	16%	16%
이미지2	44.3%	50%	50%
이미지3	11.2%	11%	11%
이미지4	49.8%	16%	**40%**
평균	37.15%	23.25%	29.25%

4개의 테스트 이미지로 테스트 한 결과, 시간 개선률은 약 37.15%를 달성할 수 있었고, 정확률은 23.25%에서 29.25%로 25%개선을 이룰 수 있었다.