|  |
| --- |
| 02. Histogram  K-means  Histogram equalization and K-means clustering |

|  |
| --- |
| **과제 요약** |
| **1. 그림 3-6에 대한 히스토그램 평활화의 수행 결과**  **2. 그림 3-9에 대한 k-means 알고리즘 수행 결과(k= 10, 20, 30)** |

|  |  |  |  |
| --- | --- | --- | --- |
| **제출 일자** | 2018. 10. 26 | **전 공** | 컴퓨터공학 |
| **과 목** | 영상처리 특론 | **학 번** | 1004660 |
| **담당 교수** | 이 준 재 | **이 름** | 박 준 수 |

# ***요 약***

*이미지에 포함된 각 영역의 밝기 정도는 그 차이가 심해질수록 대비가 높다라고 표현한다. 이미지의 대비가 낮은 경우, 이미지 내의 객체들을 식별하기 어려우며 이는 이미지의 퀄리티에 영향을 미친다. 히스토그램 평활화는 이미지의 대비를 높여줌으로써, 이미지를 보다 나은 이미지로 개선시키는 방법이다. 본 문서에서는 히스토그램 평활화를 구현하고 이에 몇가지 이미지를 적용해 봄으로써 이미지의 개선 여부를 확인한다.*

*K-means 클러스터링 알고리즘은 이미지 내의 다양하게 분포된 색상을 K개로 군집화(그룹화)하여, 이미지의 정보를 양자화하는 기법이다. 이미지의 크기가 필요이상으로 크다거나, 이미지 색상의 구성이 단순한 경우 K-means 클러스터링 알고리즘을 이용하여 이미지의 크기를 줄일 수 있다. 뿐만 아니라 K-means 클러스터링 알고리즘은 이미지의 객체들을 판별하는 알고리즘에 응용될 수 있다. 본 문에서에서는 K-means 클러스터링 알고리즘을 구현하고 이에 몇가지 이미지를 적용해 봄으로써 이미지가 단순화 됐을 시, 이미지의 퀄리티 차이를 비교한다.*

# **1. 서 론**

**1.1. 히스토그램 평활화**

히스토그램 평활화(equalization; 평준화, 균일화)는 이미지의 히스토그램을 사용해 특정 영역에 집중된 값을 고루 퍼뜨리는 이미지 처리 방법이다[1].

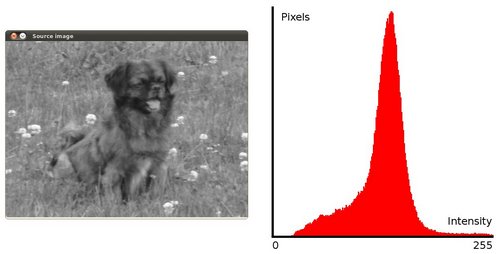


Figure . 평활화를 적용하기 전 이미지와 히스토그램[2]

Figure 1은 이미지(좌측)에 대한 히스토그램(우측)을 보인다. 히스토그램은 이미지의 색상 정보의 크기를 그래픽화하여 표현한 정보다. 이를 통해 각 명도 값의 픽셀 수를 정량화 한다.

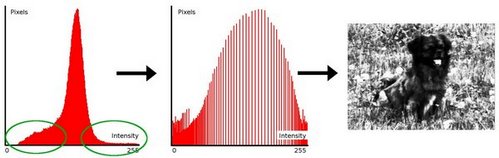


Figure . Figure 1을 평활화 한 히스토그램과 이미지[2]

회색조 이미지에 대해 이를 적용하면, 이미지의 대비가 향상된다. Figure 2를 Figure 1의 이미지의 대비가 증가 했음을 확인할 수 있다.

일반적으로 회색조의 이미지에 사용하여, 이미지의 대비를 개선하는 용도로 사용되지만, 컬러 이미지에 사용할 수도 있다. 하지만, 이 경우 각각의 색상 채널에 상대 분포가 변경되어 이미지의 색상 균형이 크게 달라질 수 있다. 이 경우 RGB 좌표계를 다른 좌표계로 변경하여 사용함으로써, 색조 및 채도의 변화 없이 적용 가능하다.

균등화는 주어진 이미지의 히스토그램을 더욱 넓은 형태의 히스토그램으로 변환하여 값들을 분산시킨다. 이를 수행하기 위해서는 누적 분포 함수(CDF; Cumulative Distribution Function)가 필요하다. 이는 히스토그램의 처음 값부터 끝 값에 이를 때까지 누적한 값이다. 즉, cdf(x)는 히스토그램의 0번째 값부터 x 값까지의 누계이다. 결과적으로 cdf는 아래의 Figure 3과 같이 표현되는 데이터 집합이 된다.

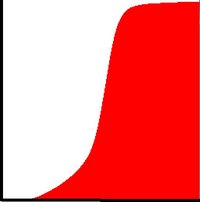


Figure . CDF 예시

이렇게 만들어진 cdf를 통해, 원본 이미지의 특정 픽셀의 값들을 매핑함으로써, 결과를 얻을 수 있다.

**1.2. K-Means 클러스터링**

K-means 알고리즘은 주어진 데이터를 k개의 클러스터로 묶는 알고리즘으로, 각 클러스터와 거리 차이의 분산을 최소화하는 방식으로 동작한다. 신호처리에서 유래하여, 데이터 마이닝, 클러스터 분석 등에 사용되는 인기있는 알고리즘이다[3].

K-means 알고리즘은 전체 데이터를 K개의 클러스터로 나누는 것을 목표로 한다. 각 데이터에 대해 K개의 데이터 중 가장 가까운 데이터로 데이터를 대치한다. 이미지 처리에서는 전체 픽셀의 색 정보들을 K 개의 색 정보로 압축, 양자화 하여, 이미지의 크기를 줄이는 기법으로 사용된다.

이 알고리즘은 데이터를 양자화하기 위해 모든 데이터에 대해 각각의 거리를 계산해야 한다는 근본적인 문제를 안고 있으며, 이 문제는 NP문제이다.

아래의 Figure 4와 Figure 5는 K-means 알고리즘의 동작 전과 동작 후의 샘플 데이터를 보인다.

|  |  |
| --- | --- |
| Test Data  Figure . 양자화 전의 샘플 데이터[4] | Result of KMeans Clustering  Figure . 양자화 후의 샘플 데이터[4] |

K-means 알고리즘은 Figure 4의 데이터들을 Figure 5의 2개의 지점을 지정한 것과 같이, 그룹화한다 (위의 예는 k=2). 그리고 이 데이터를 이미지에 적용함으로써, 이미지의 데이터를 줄일 수 있다.

**2. 구현 및 실험**

본 실험은 다음과 같은 환경의 Jupyter Notebook으로 실험되었다.

|  |
| --- |
| matplotlib==3.0.0  numpy==1.15.1 |

**1.1. 히스토그램 평활화**

히스토그램 평활화를 구현하기 위해, 행렬 계산(이미지 계산)에 numpy 모듈을 사용하며, 그래프와 이미지 입출력을 위해 matplotlib 모듈을 사용한다.



Figure . 원본 이미지(좌)와 히스토그램 평활화 수행 이미지(우)

Figure 6에서 볼 수 있듯 히스토그램 평활화를 수행하고 난 뒤의 이미지가 원본이미지에 비해 대비가 더욱 뚜렷해졌음을 알 수 있다.

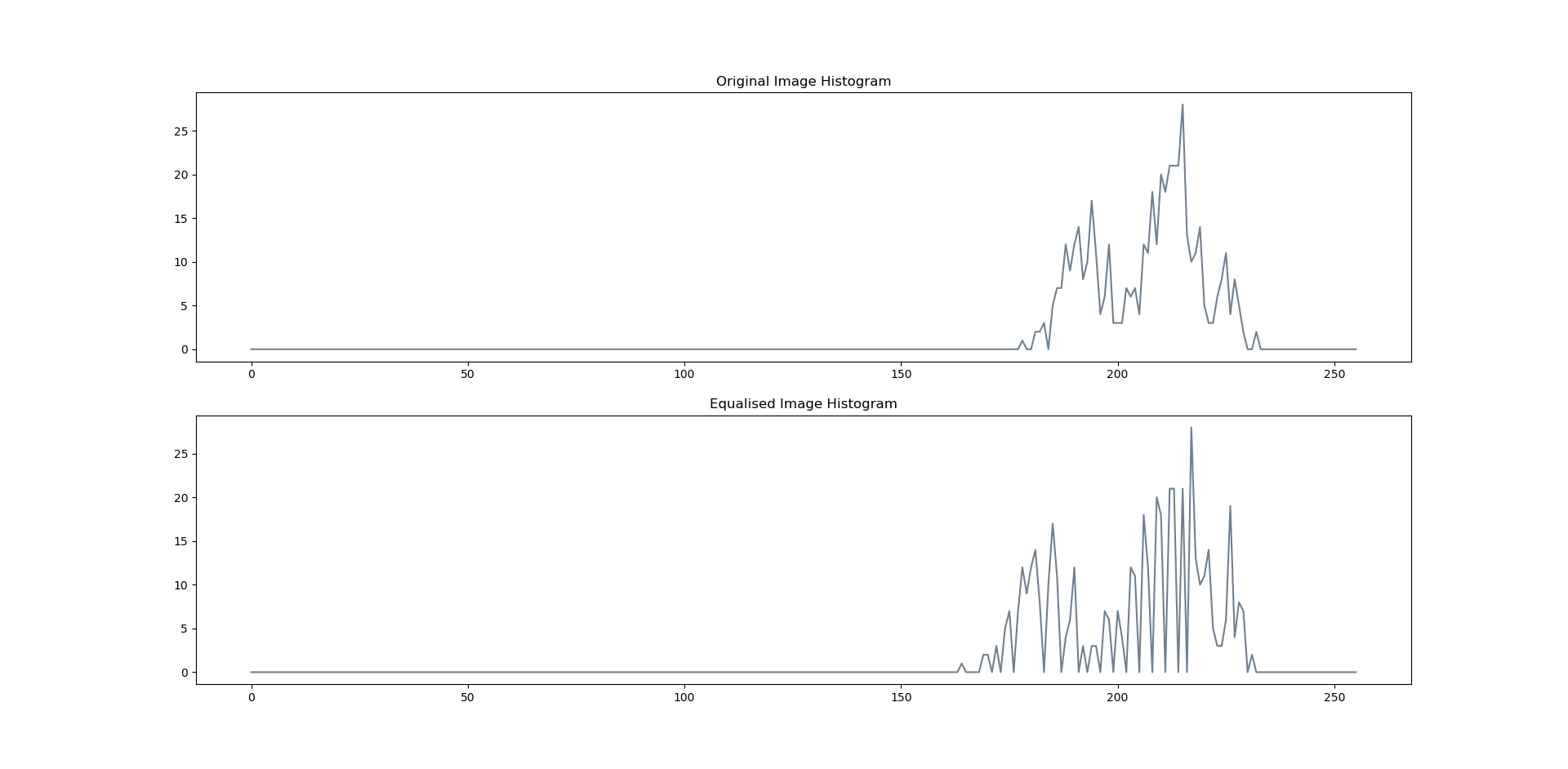


Figure . 원본이미지(상)와 평활화 수행 이미지(하)의 히스토그램

서론에서 언급한 바와 같이, 평활화를 수행한 이미지의 히스토그램은 원본 이미지의 히스토그램을 넓힌듯한(좌우로 당긴 듯한) 모양을 보인다.

아래 Figure 8은 흑백이미지를 사용해, 평활화를 이용한 대비의 차이를 더욱 극명하게 보인다.

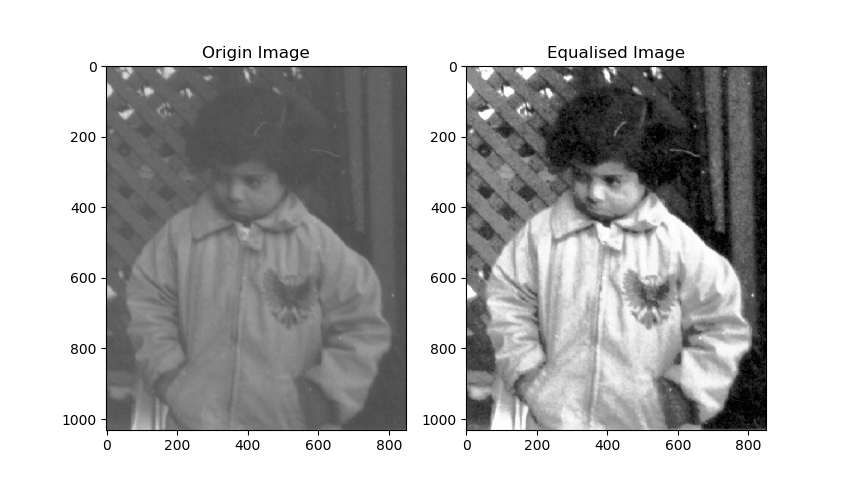


Figure . 흑백 이미지를 이용한 히스토그램 평활화 수행 전(좌)과 후(우)

히스토그램 평활화 구현 소스코드 전문은 [5]에 있으며, IPython Jupyter Notebook을 이용해 시각화한 소스코드는 [6]에 있다.

**1.2. K-Means 클러스터링**

K-Means 클러스터링을 구현하기 위해, 행렬 계산(이미지 계산)에 numpy 모듈을 사용하며, 그래프와 이미지 입출력을 위해 matplotlib 모듈을 사용한다.

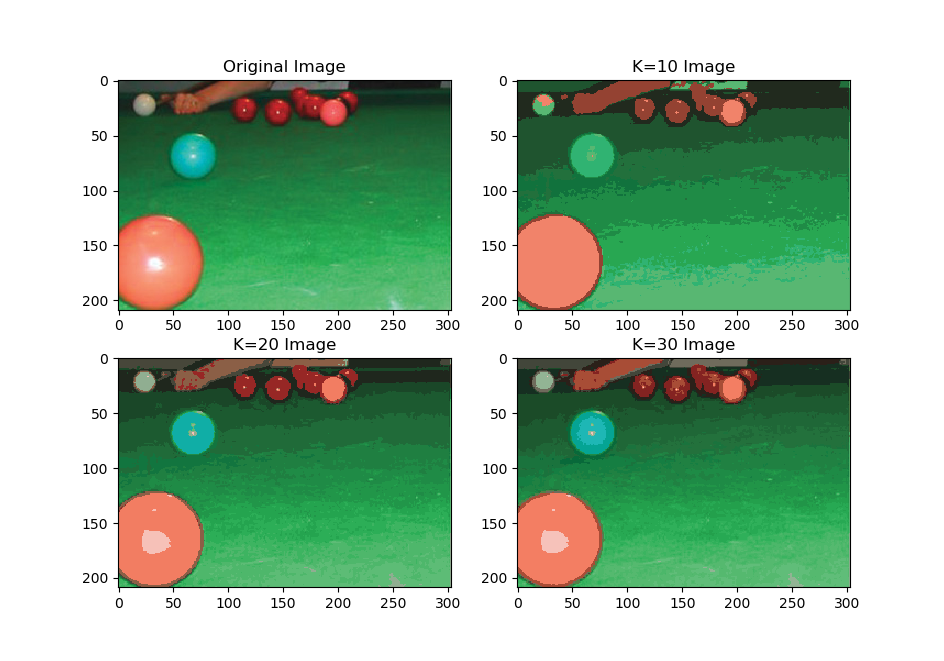


Figure . k를 10, 20, 30으로 설정한 이미지

Figure 9에는 원본 이미지(좌상)에 k를 10(우상), 20(좌하), 30(우하)으로 각각 설정한 이미지들을 보인다. 위에서 알 수 있듯, k가 10인 경우 좌측하단에 붉은 색 공에 빛 반사색이 드러나지 않지만, k가 20을 넘어가면 빛 반사의 흔적들이 드라며, 다른 색조들도 단순하게 양자화 되었음을 알 수 있다.

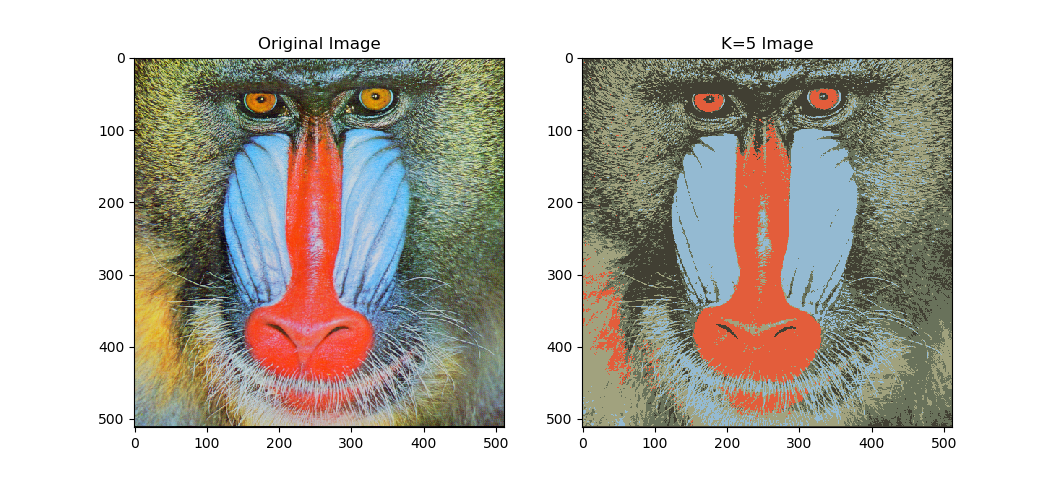


Figure . 다양한 색상을 가진 이미지에 적용한 K-means 알고리즘

Figure 10에는 원본(좌)과 이를 5개의 색상으로 양자화 한 이미지(우)를 보인다. 원숭이의 볼과 코 부분의 색상들이 극명하게 달라졌음을 알 수 있다.

K-Means 알고리즘의 구현 소스코드 전문은 [7]에 있으며, IPython Jupyter Notebook을 이용해 시각화한 소스코드는 [8]에 있다.

# **3. 결론**

히스토그램 평활화는 이미지의 밝기 정보를 조절함으로써, 이미지가 가지고 있는 대비를 개선한다. 이는 다양한 이유로 광원, 대비 등이 부족한 이미지에 적용하여, 이미지의 질을 눈에 띄게 향상시킬 수 있으며, 이를 몇가지 이미지에 대한 실험으로 보였다.

K-means 클러스터링 알고리즘은 이미지 전체에 흩어진 색상정보들을 몇가지(k 개)로 그룹화하여, 이미지의 색상 데이터들을 양자화한다. 이를 응용함으로써, 이미지 내의 객체들의 라인을 선별하거나, 이미지의 크기를 줄이는 등의 효과를 볼 수 있다. 몇가지 실험을 통해 K-means 클러스터링 알고리즘의 동작과 이미지의 변화를 실험으로 보였다.

# **4. 참고문헌**

[1] *Histogram equalization - Wikipedia*. Available: <https://en.wikipedia.org/wiki/Histogram_equalization>

[2] *Histogram Equalization — OpenCV 2.4.13.7 documentation*. Available: <https://docs.opencv.org/2.4/doc/tutorials/imgproc/histograms/histogram_equalization/histogram_equalization.html>

[3] *k-means clustering - Wikipedia*. Available: <https://en.wikipedia.org/wiki/K-means_clustering>

[4] *K-Means Clustering in OpenCV — OpenCV 3.0.0-dev documentation*. Available: <https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_ml/py_kmeans/py_kmeans_opencv/py_kmeans_opencv.html>

[5] *Github 02\_quantization/histogram/withoutCV.py at master · parkjoonsuu/Advanced-Image-Processing*. Available: <https://github.com/parkjoonsuu/Advanced-Image-Processing/blob/master/02_quantization/histogram/withoutCv.py>

[6] *Github 02\_quantization/histogram/withoutCV.ipynb at master · parkjoonsuu/Advanced-Image-Processing*. Available: <https://github.com/parkjoonsuu/Advanced-Image-Processing/blob/master/02_quantization/histogram/withoutCV.ipynb>

[7] *Github 02\_hist\_and\_kmeans/k-mean/withoutCv.py at master · parkjoonsuu/Advanced-Image-Processing*. Available: <https://github.com/parkjoonsuu/Advanced-Image-Processing/blob/master/02_hist_and_kmeans/k-mean/withoutCv.py>

[8] "Github 02\_hist\_and\_kmeans/k-mean/withoutCv.ipynb at master · parkjoonsuu/Advanced-Image-Processing."