제4장 영상의 화질 향상 기법

한밭대학교 전자공학과
곽수영

화질 향상 기법

- 영상 화질 향상을 위한 매핑 함수 M

$$s = M(r)$$

- 입력 영상의 한 픽셀 r을 매핑 함수 M에 적용하여 화질이 향상된 출력 픽셀 s를 얻어낼 수 있음
- 매핑 함수 M의 형태가 입력 픽셀에 대한 출력 결과를 결정짓는 연산자로 작용함
  - 임계값 적용(thresholding)
  - 히스토그램 스트레칭(histogram stretching)

이번 장에서 다룰 내용

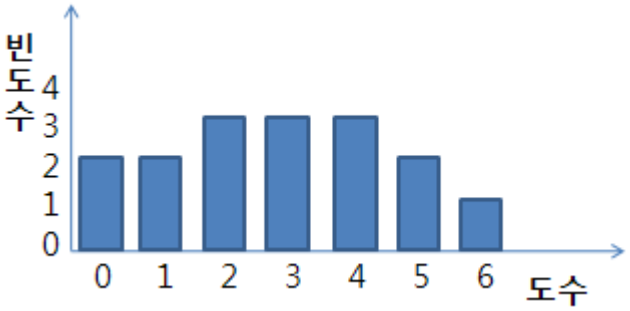
- 히스토그램(Histogram) 만들기
- 임계값 적용하기(Thresholding)
- 전역 임계값 적용하기(Global Thresholding)
- 적응적 임계값 적용하기(Adaptive Thresholding)
- 히스토그램 스트레칭(Histogram Stretching)
- 히스토그램 평활화(Histogram Equalization)

4.1 히스토그램 만들기

히스토그램 구하기

- 디지털 영상의 히스토그램
  - 관찰한 데이터의 특징을 한눈에 알아볼 수 있도록 데이터를 막대그래프 모양으로 나타낸 것
  - 디지털 영상에 대한 많은 정보를 제공함.

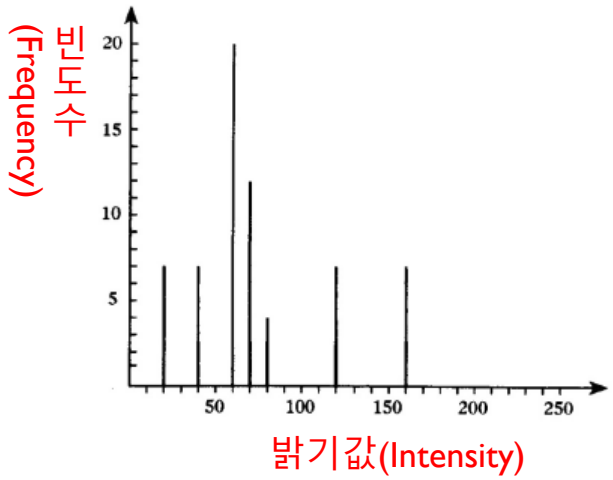
3	3	4	4
2	2	0	0
3	4	1	2
6	5	5	1



디지털 영상의 히스토그램

- 히스토그램(Histogram)?

20	20	20	20	20	20	20	40
160	60	60	60	60	60	60	40
160	60	70	70	70	70	60	40
160	60	70	80	80	70	60	40
160	60	70	80	80	70	60	40
160	60	70	70	70	70	60	40
160	60	60	60	60	60	60	40
160	120	120	120	120	120	120	120



- 수평축: 영상의 밝기값(Intensity): 0-255
- 수직축: 수평축의 밝기값에 대응되는 크기를 가진 픽셀수가 영상안에 몇 개나 있는지 나타내는 빈도수(frequency)

디지털 영상의 히스토그램

- 히스토그램(Histogram)
  - 영상 내에서 각 그레이스케일 값에 해당하는 픽셀의 개수를 함수의 형태로 나타낸 것

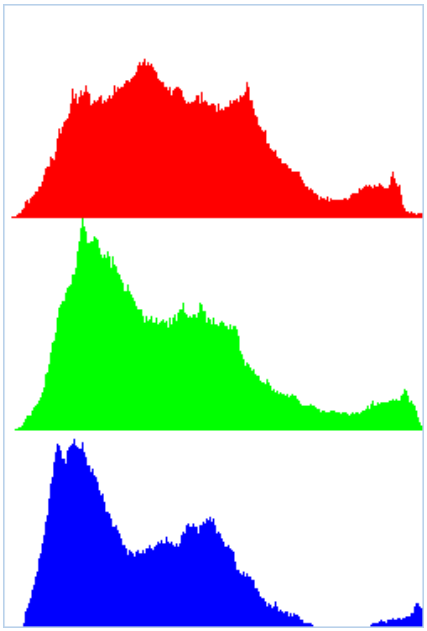
$$h(g) = n_g$$

- 정규화된 히스토그램(Normalized histogram)
  - 각 픽셀의 개수를 영상 전체 픽셀 개수로 나누어준 것
  - 해당 그레이스케일 값을 갖는 픽셀이 나타날 확률

$$p(g) = \frac{n_g}{N} \Rightarrow \sum_{g=0}^{255} p(g) = 1$$

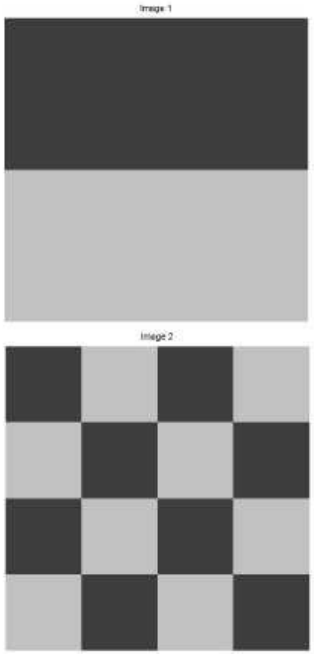
칼라영상에서의 히스토그램

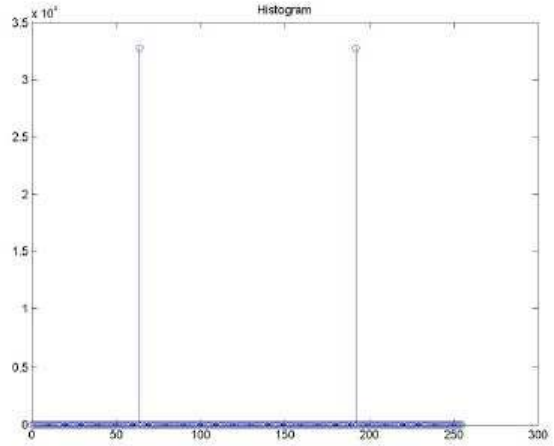
- 컬러 이미지같은 경우는 R,G,B 채널별로 분리하여 히스토그램 그리기



디지털 영상의 히스토그램

- 히스토그램의 단점

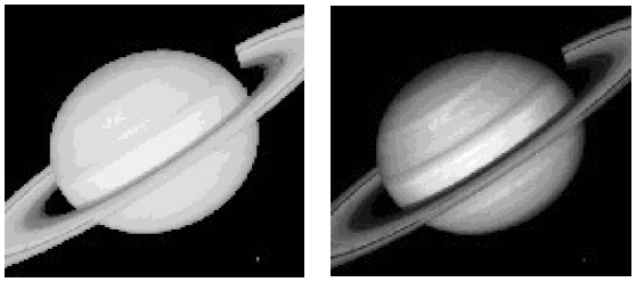




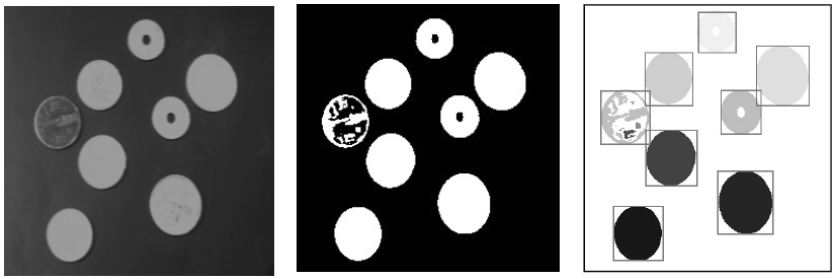
Histogram reflects the pixel intensity distribution, not the spatial distribution!

디지털 영상의 히스토그램

- 히스토그램의 용도
  - 화질향상



  - 물체인식
    - 이진화 후, 255값을 가지는 영역 및 위치를 자동 추출 할 경우



히스토그램 구하기

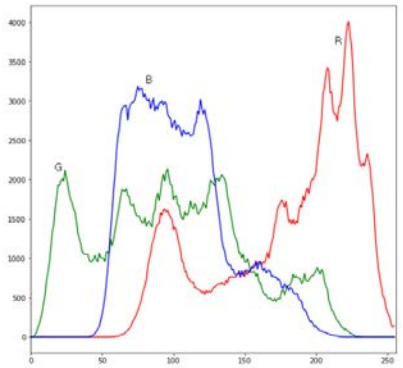
- 히스토그램 구하는 함수 calcHist()

```
hist=cv.calcHist(images, channels, mask, histSize, ranges[, hist[, accumulate]])
```

  - images : uint8 또는 float32 유형의 입력 이미지
  - channel : 히스토그램을 계산하는 채널의 인덱스.
    - 예를 들어 입력이 그레이 영상인 경우 값은 [0]이다.
    - 컬러 영상의 경우 [0], [1] 또는 [2]로 인덱스를 표현
  - mask : 마스크 영상.
    - 전체 영상의 히스토그램을 찾으려면 "None"으로 지정
    - 영상의 특정 영역에 대한 히스토그램을 찾으려면 해당 영역에 대한 마스크 영상을 만들어 마스크로 제공해야 한다
  - histsize : 히스토그램 bin 개수.
    - 영상 전체 색상을 나타내고 싶을 때 [256]로 표기
  - ranges : 색상 범위
    - 일반적으로 [0,255]로 쓴다

히스토그램 구하기

```
from google.colab.patches import cv2_imshow
import numpy as np
import cv2
import matplotlib.pyplot as plt
from google.colab import drive
drive.mount('/content/gdrive')
#영상 불러오기
img = cv2.imread('/content/gdrive/My Drive/Image_Processing/lena.jpg')
# BGR채널순서를 RGB채널로 변경
RGB_img1 = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
# RGB 채널 나누기
R_img1,G_img1,B_img1=cv2.split(RGB_img1)
plt.figure(figsize=(20,20))# 영상의 크기를 키워주자
plt.subplot(111), plt.imshow(RGB_img1)  #출력영상의 위치 지정 및 표시
plt.axis("off")
# 히스토그램 출력, 채널은 0으로 표시
hist = cv2.calcHist([R_img1],[0],None,[256],[0,255])
plt.subplot(112),plt.plot(hist,color = 'r')  #히스토그램을 red로 그림
plt.xlim([0,256])   #도형에서 x축의 범위를 나타냄
hist = cv2.calcHist([G_img1],[0],None,[256],[0,255])
plt.plot(hist,color = 'g')
plt.xlim([0,255])   #도형에서 x축의 범위를 나타냄
hist = cv2.calcHist([B_img1],[0],None,[256],[0,255])
plt.plot(hist,color = 'b')
plt.xlim([0,255])     #도형이 표현될 위치, 기본형은 subplot(nrows,ncols,index,**kwargs)로 (221)은
plt.show()            2행 2열에서 1번째 위치 (2,2,1)과 동일하다
```



hist = cv2.calcHist([R_img1],[0],None,[256],[0,256]):

[R_imag]영상 배열, 채널은 0 (R, G, B로 각각 나누어져 있으므로), 마스크는 사용하지 않고, 표현하는 히스토그램 빈은 256, X축의 범위는 0~255이다.

4.2 임계값 적용하기(Thresholding)

임계값 적용하기

- 전경 영역과 배경 영역의 밝기 차이나 색상 차이를 이용하여 영상의 배경으로부터 전경 영역을 분리할 수 있는 가장 기본적인 방법
  - 관심 영역 vs. 비관심 영역
  - 배경(background) vs. 객체(object)

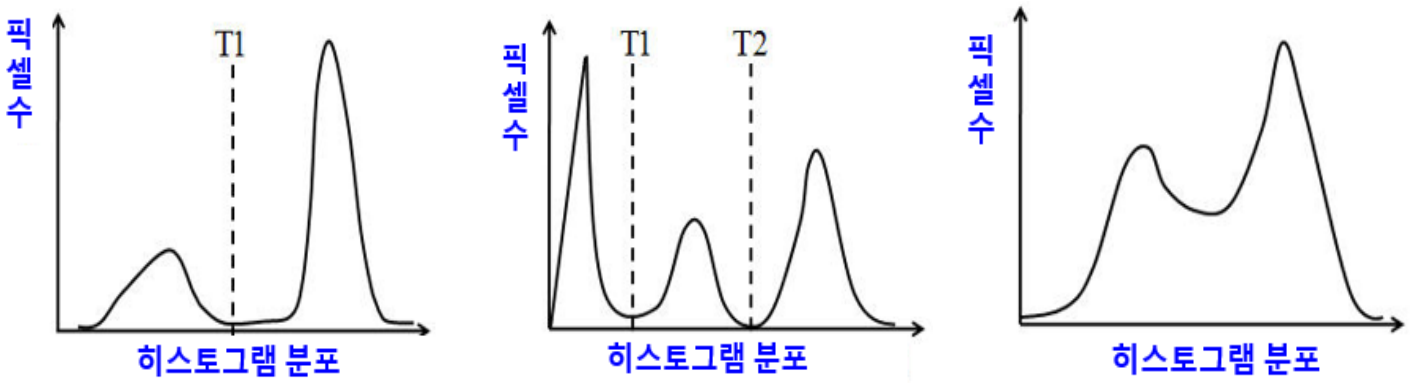
- 그레이스케일 영상의 이진화

$$g(x,y) = \begin{cases} 0 & \text{if } f(x,y) \le T \\ 255 & \text{if } f(x,y) > T \end{cases}$$

T : 임계값, 문턱치(threshold)

임계값 적용하기

- 임계값은 인위적으로 프로그래머에 의해 결정될 수도 있지만, 영상으로부터 히스토그램을 생성하여 결정할 수도 있음



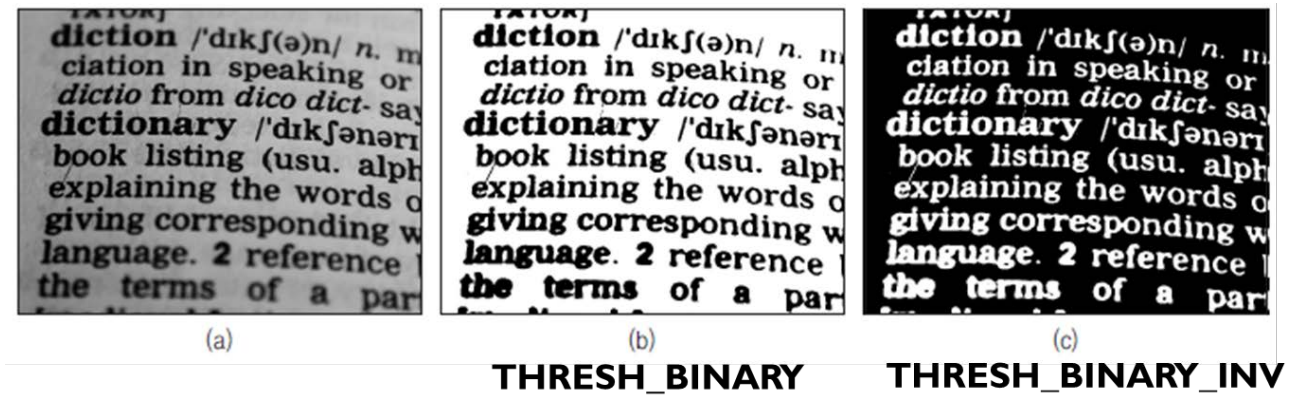
히스토그램을 이용한 임계값 결정

임계값 적용하기 실습

cv2.threshold(src, thresh, maxval, type)

- src : input image로 single-channel image (grayscale image)
- thresh: 임계값
- maxval: 임계값을 넘었을때 적용할 value
- type: thresholding type

ThresholdTypes 열거형 상수	설명
THRESH_BINARY	$\mathrm{dst}(x,y) = \begin{cases} \mathrm{maxval} & \mathrm{src}(x,y) > \mathrm{thresh}\ 일\ 때 \\ 0 & 그\ 외 \end{cases}$
THRESH_BINARY_INV	$\mathrm{dst}(x,y) = \begin{cases} 0 & \mathrm{src}(x,y) > \mathrm{thresh}\ 일\ 때 \\ \mathrm{maxval} & 그\ 외 \end{cases}$



4.3 전역 임계값 적용하기(Global Thresholding)

전역 임계값 적용하기

- 영상을 두 개의 그룹으로 분할하고 각 그룹마다 평균값을 계산하여 이 값들을 임계값 계산에 적용하는 방법을 사용

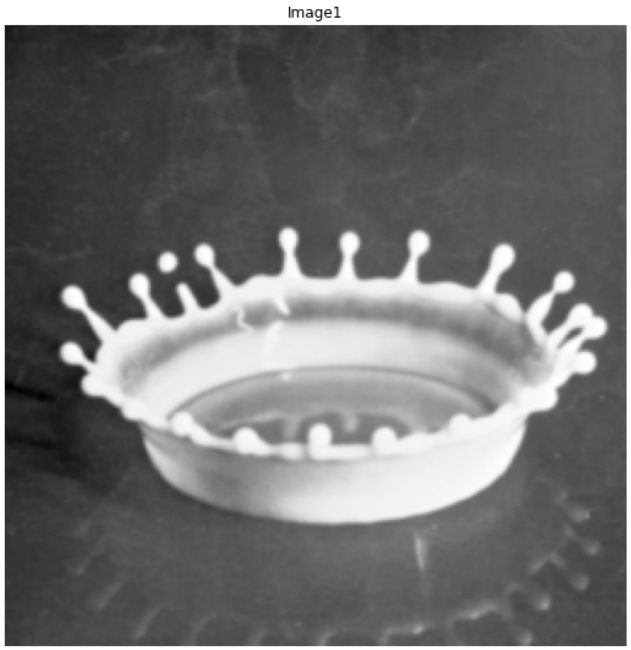
1) 초기 임계값 T를 결정한다.
2) 임계값 T를 이용하여 영상을 분할한다. 분할 결과로 2개의 그룹이 생성되는데, 임계값 T보다 큰 픽셀로 이루어진 그룹을 G1, 반대의 그룹을 G2로 표시한다.
3) 그룹 G1과 G2의 색상 평균(mean1, mean2)을 계산한다.
4) 두 그룹의 평균값으로 임계값을 갱신한다.

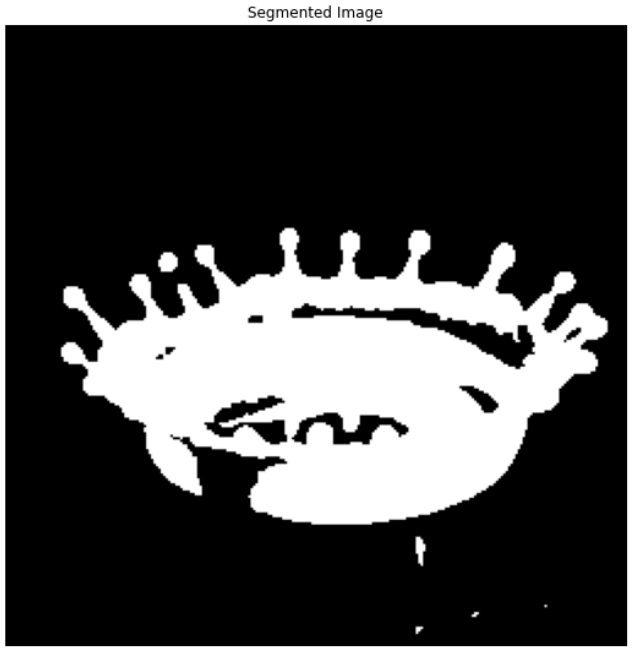
$$T=(1/2)(mean1 + mean2)$$

5) 이전 임계값과 새로운 임계값의 차이가 미리 정의된 파라미터 T0 보다 작을 동안 단계 2부터 단계 4를 반복한다.

- 종료 조건에 해당하는 T0에 따라 영상 분할 속도가 달라질 수 있음
- T0값이 작을 경우 최적의 임계값을 추정할 수 있는 장점이 있지만 많은 연산이 필요하므로 실행 속도가 느려질 수 있음

연습 4-3 : 전역임계값으로 영상 분할





4.4 적응적 임계값 적용하기(Adaptive Thresholding)

적응적 임계값 적용하기

- 하나의 임계값만으로 전경과 배경을 구분하는 문제점을 해결하기 위해 영상을 일정 블록으로 분할하고 각 분할된 블록마다 제각기 다른 임계값을 적용하는 방법

적응적 임계값 적용하기

- 임계값을 결정하는 방법
  - 블록 안의 픽셀 밝기 분포에 대한 평균

$$T = \frac{\sum I(x, y)}{\text{블록안의픽셀개수}}$$

  - 블록 안의 픽셀 밝기의 중간값

$$T = Median(I(x, y) \in block)$$

  - 블록 안의 픽셀 밝기의 최대값과 최소값의 평균

$$T = \frac{\max(i(x, y)) + \min(i(x, y))}{2}$$

적응적 임계값 적용하기

- 블록의 크기가 너무 작을 경우, 영상이 세분화되어 분리되므로 잡영상이 많아짐
- 블록의 크기가 너무 큰 경우, 블록 내의 픽셀들은 일정한 조명을 갖는다는 적응적 임계값 적용 방법의 기본 가정을 위반하므로 전역적 임계값 방법과 유사한 결과를 나타내는 문제점이 발생할 수 있음
- 영상의 특성과 크기에 따라 블록의 크기를 가변적으로 결정

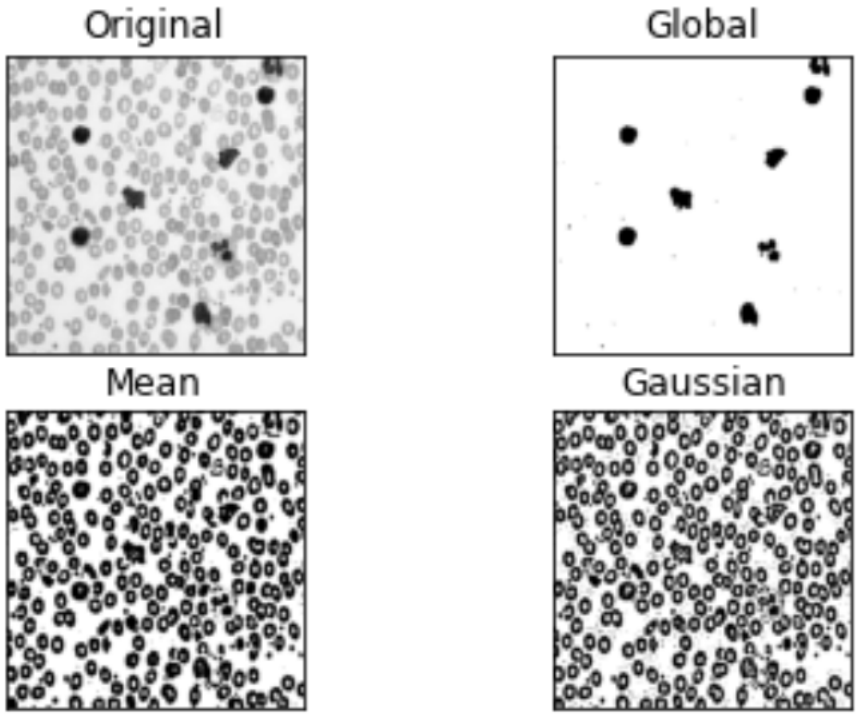
적응적 임계값 적용하기 실습

cv2.adaptiveThreshold(src, maxValue, adaptiveMethod, thresholdType, blockSize, C[, dst])

- src – grayscale image
- maxValue – 임계값
- adaptiveMethod – thresholding value를 결정하는 계산 방법
- thresholdType – threshold type
  - cv2.ADAPTIVE_THRESH_MEAN_C
    - blockSize 영역의 모든 픽셀에 평균 가중치를 적용
  - cv2.ADAPTIVE_THRESH_GAUSSIAN_C
    - blockSize 영역의 모든 픽셀에 중심점으로부터의 거리에 대한 가우시안 가중치 적용
- blockSize – thresholding을 적용할 영역 사이즈
- C – 평균이나 가중평균에서 차감할 값

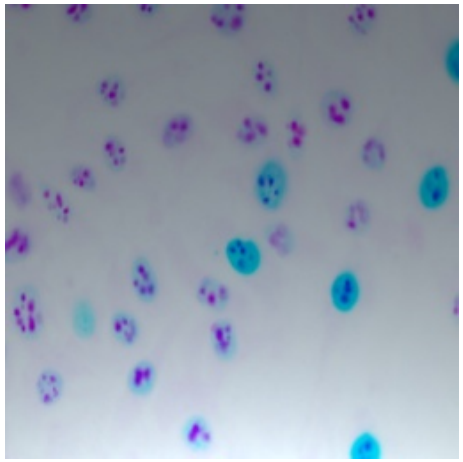
적응적 임계값 적용하기 실습

```
from google.colab.patches import cv2_imshow
import cv2
import numpy as np
import matplotlib.pyplot as plt
from google.colab import drive

drive.mount('/content/gdrive')
#영상 불러오기
img = cv2.imread('/content/gdrive/My Drive/Image_Processing/cell.bmp',0)

# img = cv2.medianBlur(img,5)

ret, th1 = cv2.threshold(img,127,255,cv2.THRESH_BINARY)

th2 = cv2.adaptiveThreshold(img,255,cv2.ADAPTIVE_THRESH_MEAN_C,cv2.THRESH_BINARY,15,2)
th3 = cv2.adaptiveThreshold(img,255,cv2.ADAPTIVE_THRESH_GAUSSIAN_C,cv2.THRESH_BINARY,
15,2)

titles = ['Original','Global','Mean','Gaussian']

images = [img,th1,th2,th3]

for i in range(4):
  plt.subplot(2,2,i+1),plt.imshow(images[i],'gray')
  plt.title(titles[i])
  plt.xticks([]),plt.yticks([])

plt.show()
```


Original
Global
Mean
Gaussian

적응적 임계값 적용하기 실습

- 조명 왜곡이 발생한 영상에 대한 적응적 임계값 적용 결과



원본 영상

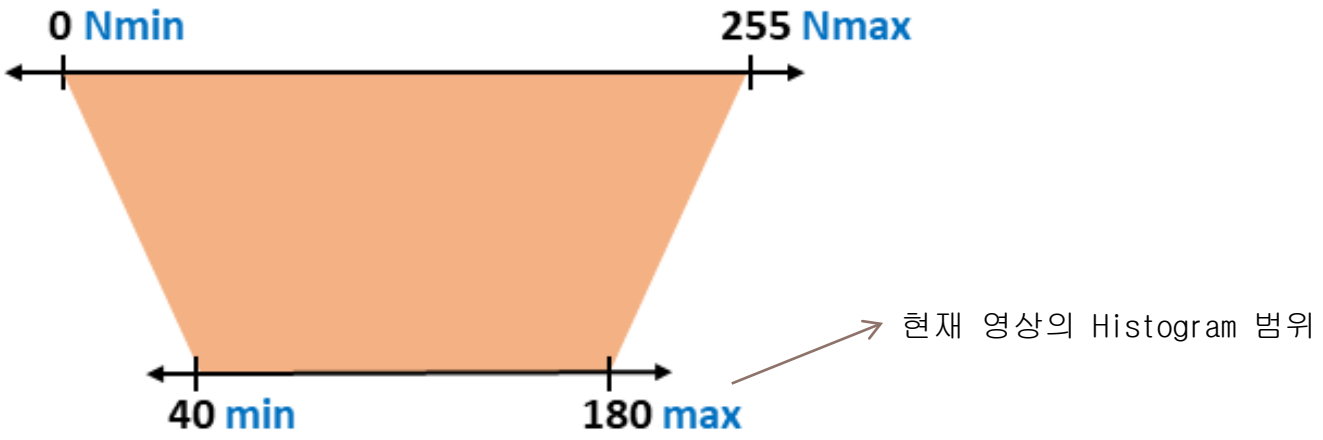
전역 임계값 적용 결과

적응적 임계값 적용 결과

4.5 히스토그램 스트레칭 (Histogram Stretching)

히스토그램 스트레칭

- 영상의 밝기값 범위를 확장(또는 축소) 시킴으로써 영상의 대비를 향상시키는 방법
- 영상 픽셀들에 대해 선형 확장 함수(linear scaling function)를 적용하여 구현



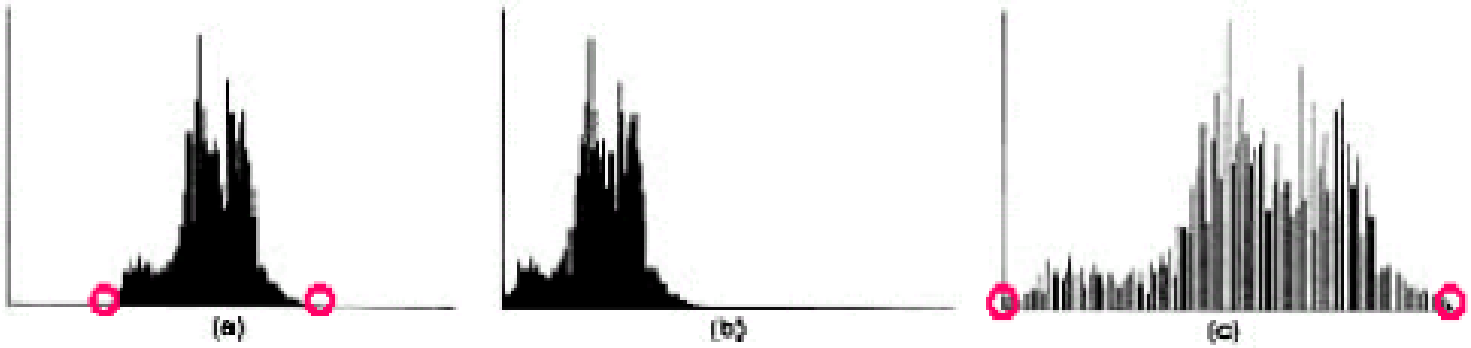
히스토그램 스트레칭

- 히스토그램 스트레칭 알고리즘
  - 예제) low=40, high=180 -> 0~255로 변환

$$new\ pixel = \frac{old\ pixel - low}{high - low} \times 255$$

old pixel은 원 영상 화소의 명도 값
new pixel은 결과 영상 화소의 명도 값
low는 히스토그램의 최저 명도 값
high는 히스토그램의 최고 명도 값

3. 히스토그램 스트레칭



(a) 그레이 레벨 영상

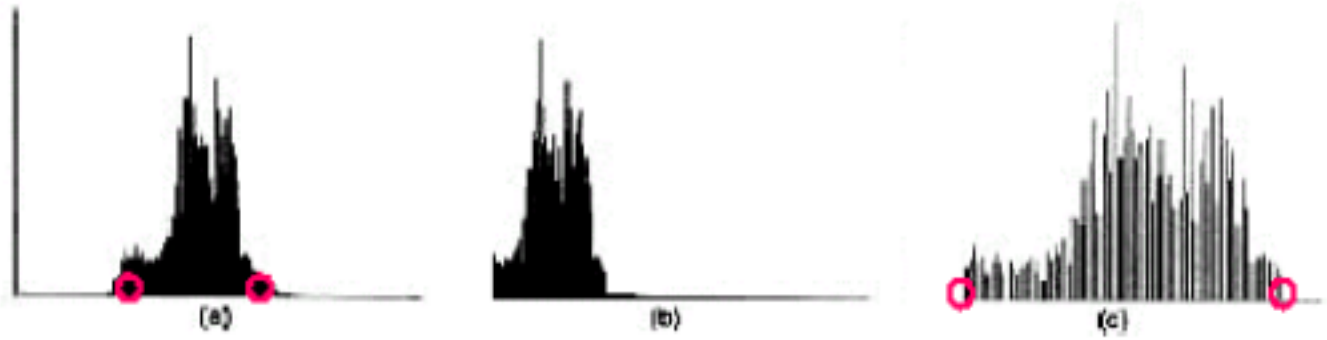
(b) 기본 명암 대비 스트레칭 영상

히스토그램 스트레칭

- 앤드-인 탐색(상하위 절단법)
  - 일정한 양의 화소를 흰색이나 검정색으로 지정하여 히스토그램의 분포를 좀더 균일하게 만듦
  - 앤드-인 탐색 수행 공식
    - 두 개의 임계 값(low, high) 사용

$$new\ pixel = \begin{cases} 0 & old\ pixel \le low \\ \dfrac{old\ pixel - low}{high - low} \times 255 & low \le old\ pixel \le high \\ 255 & high \le old\ pixel \end{cases}$$

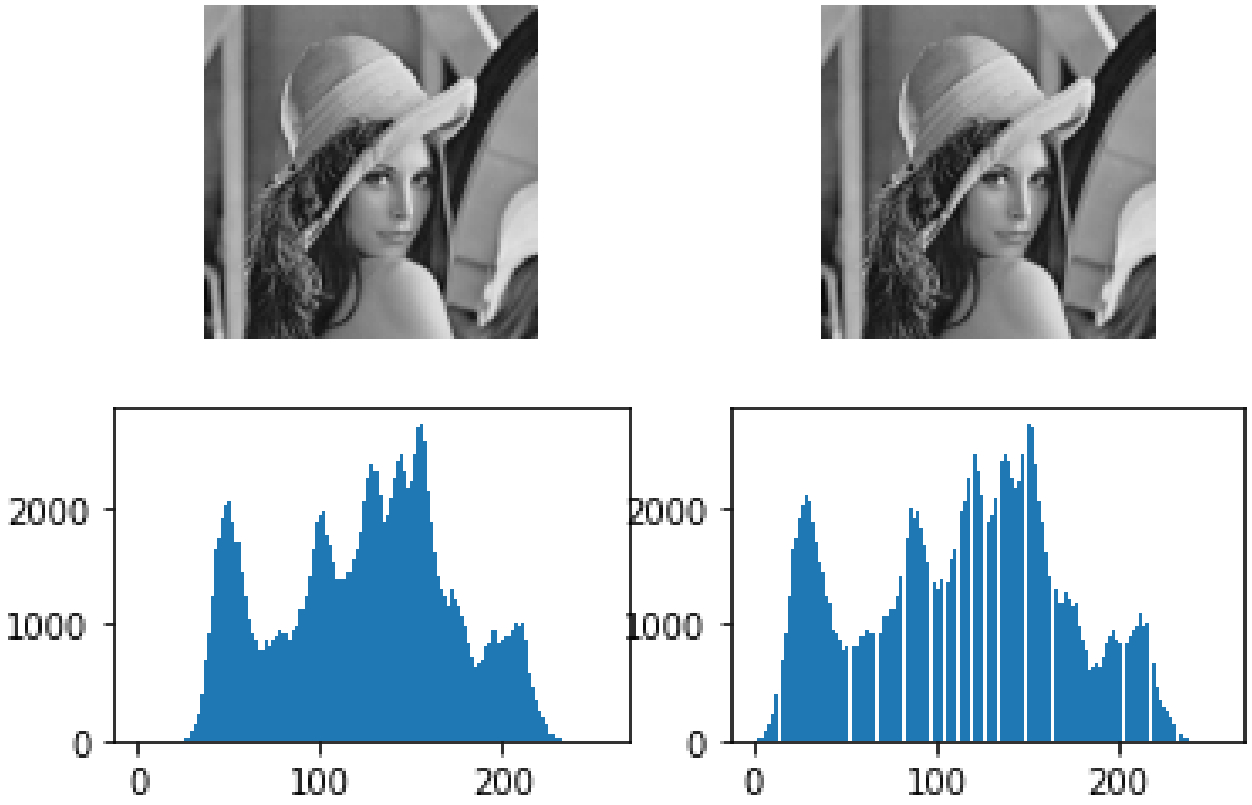
히스토그램 스트레칭



(a) 그레이 레벨 영상

(b) 엔드-인 탐색 영상

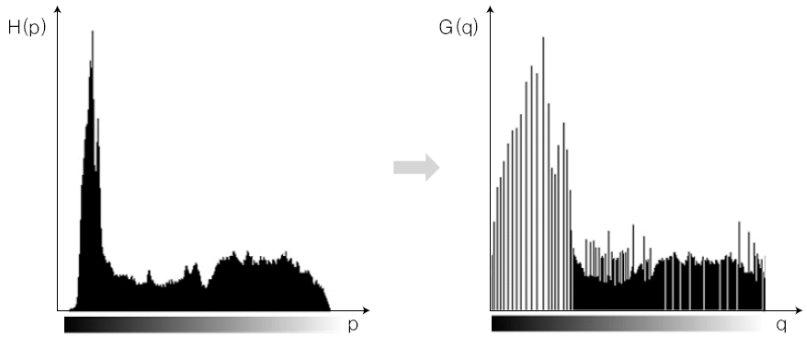
```
from google.colab.patches import cv2_imshow
import cv2
import numpy as np
import matplotlib.pyplot as plt
from google.colab import drive

drive.mount('/content/gdrive')
#영상 불러오기
img = cv2.imread('/content/gdrive/My Drive/Image_Processing/lena.jpg', cv2
.IMREAD_GRAYSCALE)
out=img.copy()
height,width=img.shape
high=img.max()
low=img.min()

for i in range(width):
  for j in range(height):
    out[i][j]=((img[i][j]-low)*255/(high-low))

plt.figure()
plt.subplot(2,2,1),plt.axis("off"),plt.imshow(img,cmap='gray')
plt.subplot(2,2,2),plt.axis("off"),plt.imshow(out,cmap='gray')
plt.subplot(2,2,3),plt.hist(img.ravel(), 256, [0,256])
plt.subplot(2,2,4),plt.hist(out.ravel(), 256, [0,256])
plt.show()
```


2000
1000
0
0
100
200
2000
1000
0
0
100
200

4.6 히스토그램 평활화(Histogram Equalization)

히스토그램 평활화

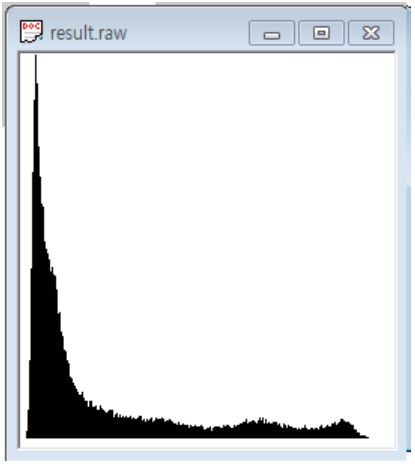
- 히스토그램 평활화 기법(Histogram Equalized)
  - 어둡게 촬영된 영상의 히스토그램을 조절하여 명암 분포가 빈약한 영상을 균일하게 만들어 줌.
  - 영상의 밝기 분포를 재분배하여 명암 대비를 최대화
  - 명암 대비 조정을 자동으로 수행
  - 각 명암의 빈도는 변경하지 않음.
  - 검출 특성이 좋은 영상만 출력하지는 않지만 영상의 검출 특성을 증가시킴



[그림 5-10] 히스토그램 평활화를 수행한 뒤 변화된 히스토그램 모습

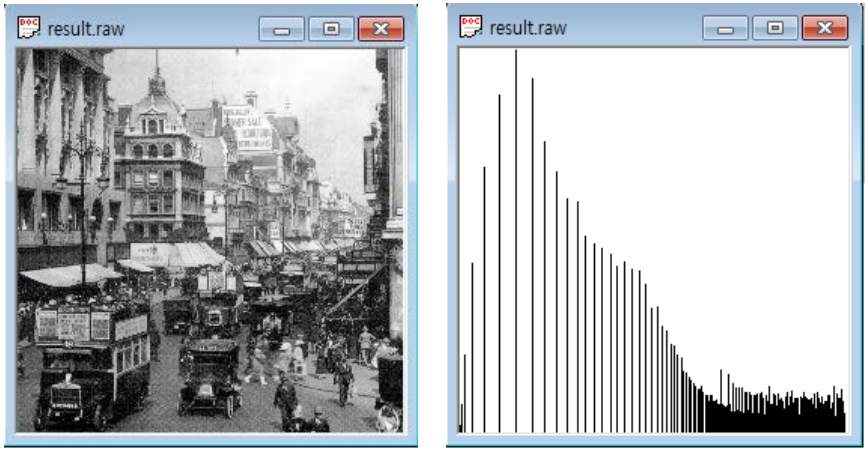
히스토그램 평활화

- why 평활화?
  - 인간의 눈은 영상의 절대적 밝기의 크기보다 **대비**가 증가할 때 인지도가 증가한다.

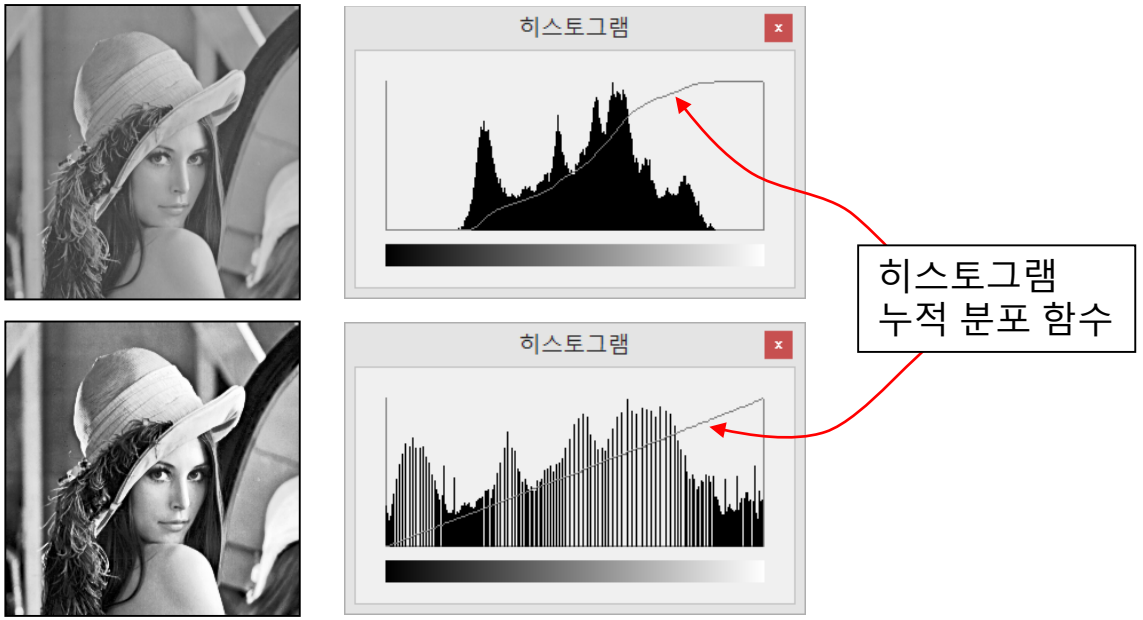


히스토그램 평활화

- 히스토그램 평활화의 단계
  - 1) 원시 입력영상의 밝기값에 대한 히스토그램을 생성한다.
    2) 생성된 히스토그램을 정규화합 히스토그램으로 변형한다.
    3) 정규화합 히스토그램을 이용하여 입력영상을 다시 매핑한다.



히스토그램 평활화



히스토그램 평활화

- 1단계 : 입력 영상의 히스토그램 생성
- 2단계 : 히스토그램의 누적합 계산

$$sum[i] = \sum_{j=0}^{i} hist[j]$$

- 3단계 : 히스토그램 누접합을 전체 픽셀개수로 나누어 정규화함(정규화 누적합), 정규화된 값에 최대 gray level값을 곱한다.

$$n[i] = sum[i] \times \frac{1}{N} \times I_{\max}$$

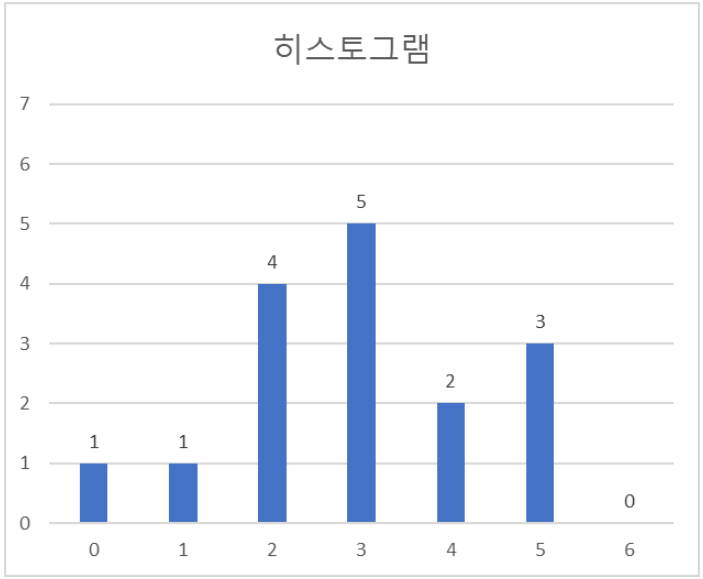
  - N은 화소의 총 수, Imax는 최대 명도 값
- 4단계
  - 3단계 값을 반올림하여 매핑되는 값을 구하고, 입력 영상의 픽셀값을 대응되는 값으로 변경

히스토그램 평활화

- 1단계: 히스토그램 계산
  - 빈도수 hist[j]에서의 히스토그램 생성

2	2	3	5
4	2	3	3
4	3	5	1
0	3	2	5

명암값	빈도수
0	1
1	1
2	4
3	5
4	2
5	3
6	0



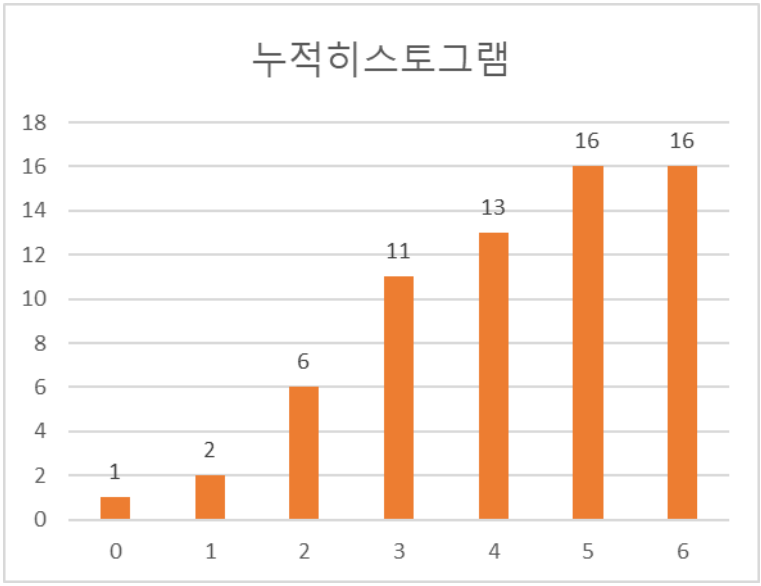
히스토그램 평활화

- 2단계: 누적히스토그램 계산
  - 누적합 sum[i]생성

$$sum[i] = \sum_{j=0}^{i} hist[j]$$

2	2	3	5
4	2	3	3
4	3	5	1
0	3	2	5

명암값	빈도수	누적빈도수
0	1	1
1	1	2
2	4	6
3	5	11
4	2	13
5	3	16
6	0	16



히스토그램 평활화

- 3단계: 누적히스토그램 정규화 x 최대 화소값
  - 누적빈도수/전체 픽셀개수 X 최대 화소값

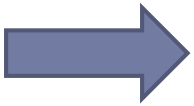
2	2	3	5
4	2	3	3
4	3	5	1
0	3	2	5

명암값	빈도수	누적빈도수	누적히스토그램 정규화 x 최대 화소값	
0	1	1	0.375	= 1/16*6
1	1	2	0.75	= 2/16*6
2	4	6	2.25	= 6/16*6
3	5	11	4.125	=11/16*6
4	2	13	4.875	=13/16*6
5	3	16	6	=16/16*6
6	0	16	6	=16/16*6

히스토그램 평활화

- 4단계:
  - 정규화된 누적합을 반올림하여 정수로 매핑

명암값	빈도수	누적빈도수	누적히스토그램 정규화 x 최대 화소값	반올림
0	1	1	0.375	0
1	1	2	0.75	1
2	4	6	2.25	2
3	5	11	4.125	4
4	2	13	4.875	5
5	3	16	6	6
6	0	16	6	6

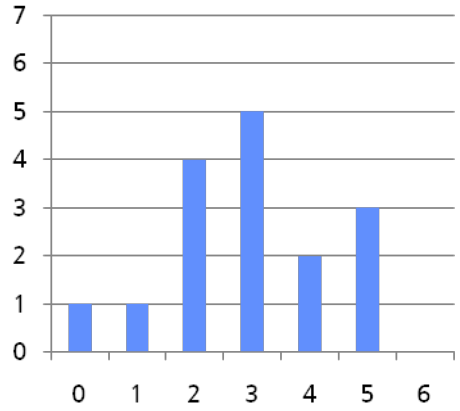


명암값	매핑값
0	0
1	1
2	2
3	4
4	5
5	6
6	6

히스토그램 평활화

2	2	3	5
4	2	3	3
4	3	5	1
0	3	2	5

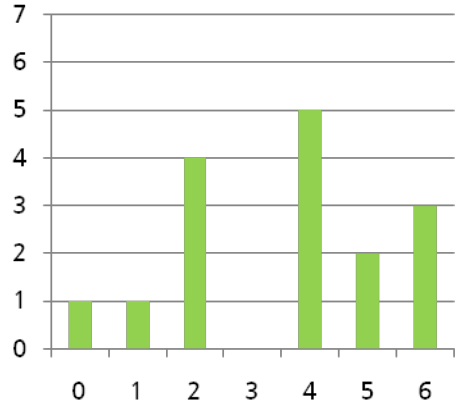
원 영상



명암값	매핑값
0	0
1	1
2	2
3	4
4	5
5	6
6	6

2	2	4	6
5	2	4	4
5	4	6	1
0	4	2	6

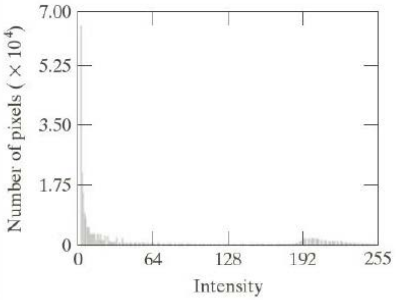
평활화 영상



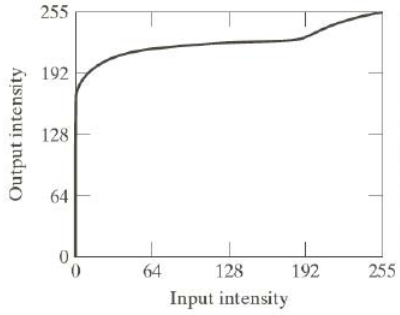
히스토그램 평활화

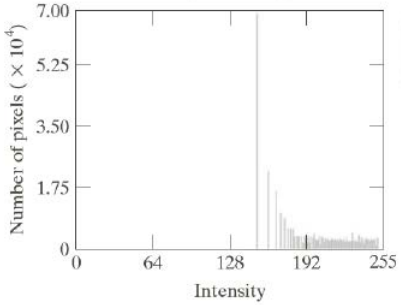
- 항상 좋은게 아님

히스토그램



0근처의 값이 많음





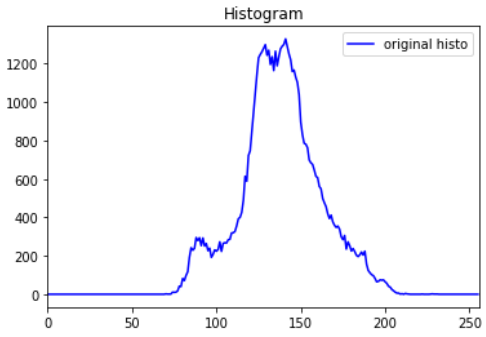
히스토그램 평활화

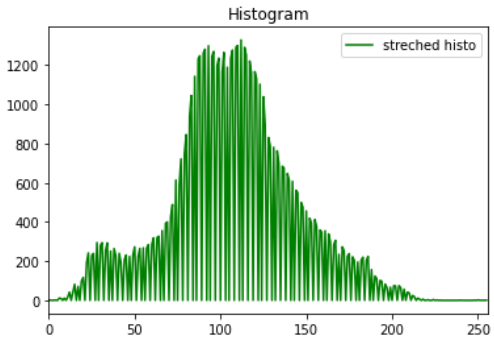
히스토그램 평활화

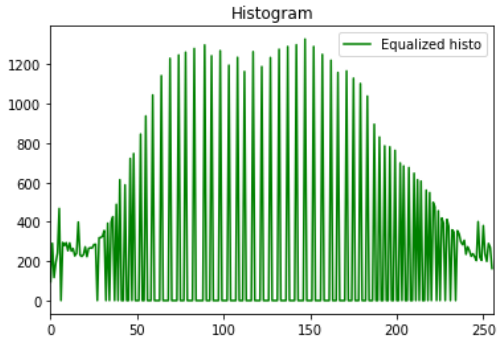
원 영상

히스토그램 스트레칭

히스토그램 평활화







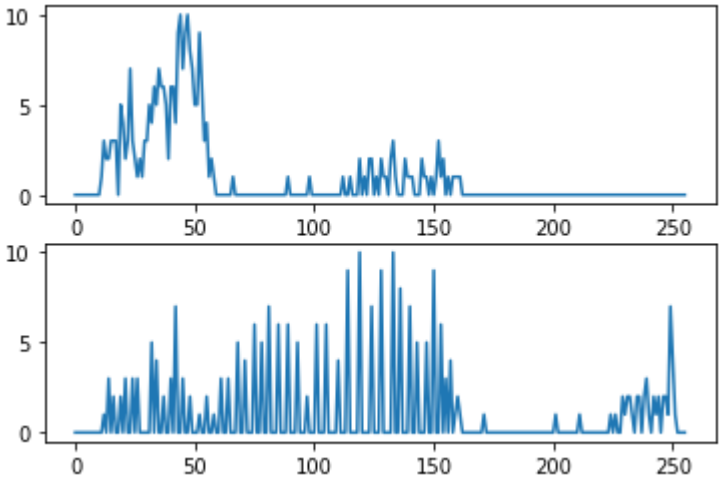
히스토그램 평활화

cv2.equalizeHist(src)

- src: 그레이 스케일 영상

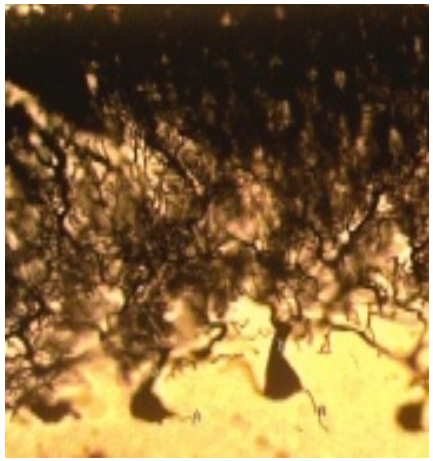
```
from google.colab.patches import cv2_imshow
import cv2
import numpy as np
import matplotlib.pyplot as plt
from google.colab import drive

drive.mount('/content/gdrive')
img = cv2.imread('/content/gdrive/My Drive/Image_Processing/girl.jpg', 0)

img2 = cv2.equalizeHist(img)

dst = np.hstack((img,img2)) # 영상 옆으로 쌓기
cv2_imshow(dst)

plt.figure()
img_hist=cv2.calcHist(img,[0],None,[256],[0,255])
plt.subplot(2,1,1), plt.plot(img_hist)
img2_hist=cv2.calcHist(img2,[0],None,[256],[0,255])
plt.subplot(2,1,2), plt.plot(img2_hist)
plt.show()
```

히스토그램 평활화



히스토그램 평활화

- 히스토그램 평활화와 스트레칭의 결과 비교



원 영상

히스토그램 스트레칭 결과
(최하 5%, 최고 95%)

히스토그램 평활화 영상