Chapter

4

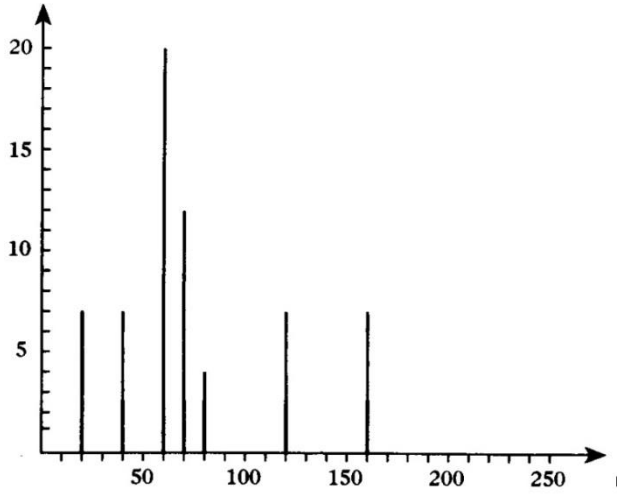
화소처리(2)

3

히스토그램

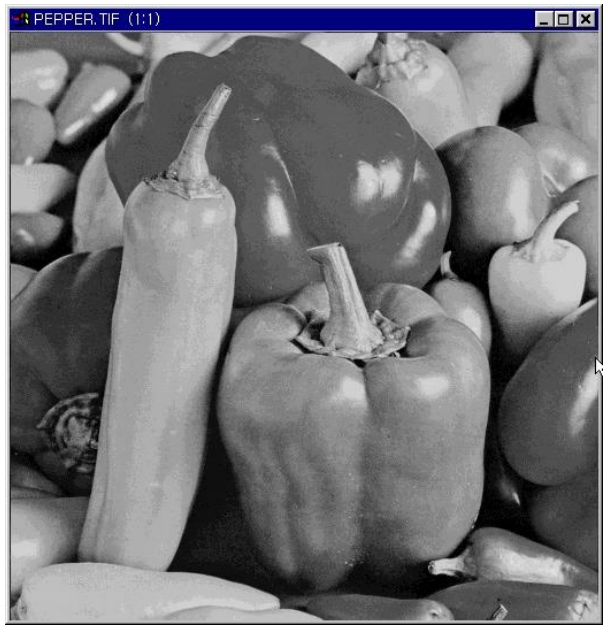
히스토그램(Histogram)이란 ?

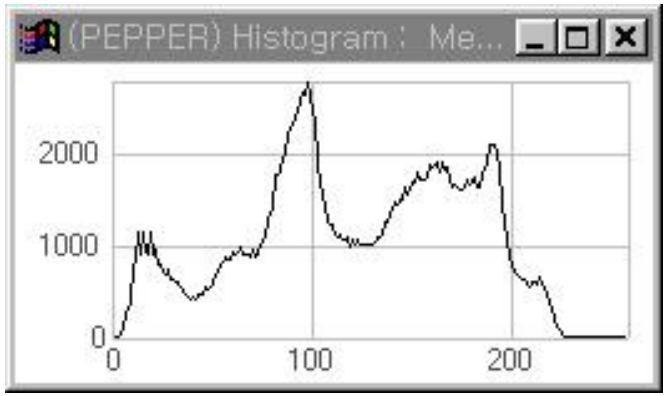
20	20	20	20	20	20	20	40
160	60	60	60	60	60	60	40
160	60	70	70	70	70	60	40
160	60	70	80	80	70	60	40
160	60	70	80	80	70	60	40
160	60	70	70	70	70	60	40
160	60	60	60	60	60	60	40
160	120	120	120	120	120	120	120



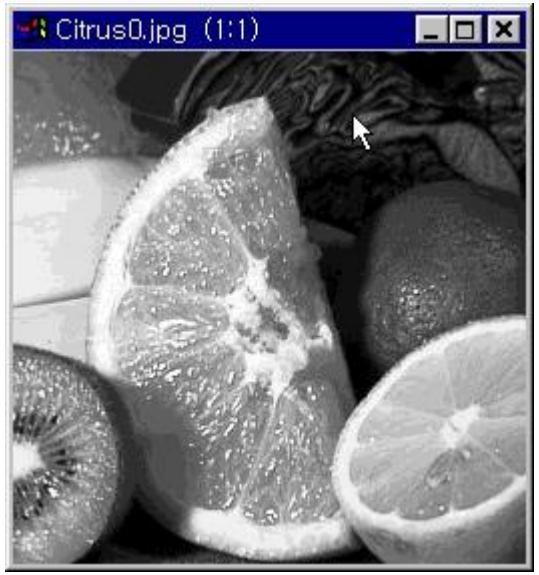
- 수평축: 영상의 밝기(intensity) 값
- 수직축: 수평축의 밝기값에 대응되는 크기를 가진 픽셀 수가 영상 안에서 몇 개나 있는지 나타내는 빈도수(frequency)

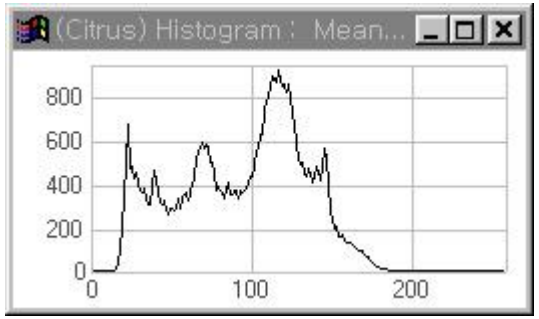
- 흑백영상의 경우 수평축은 0~255의 범위, 수직축의 값은 영상의 크기와 밝기의 분포에 따라 달라짐

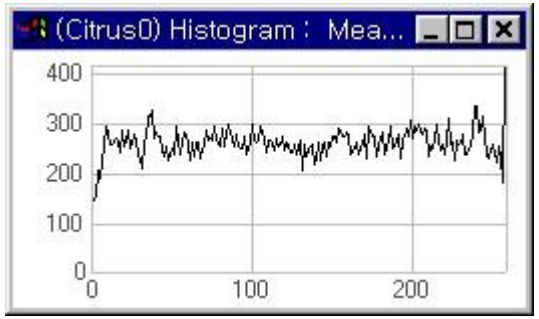




밝기분포가 다른 영상의 예

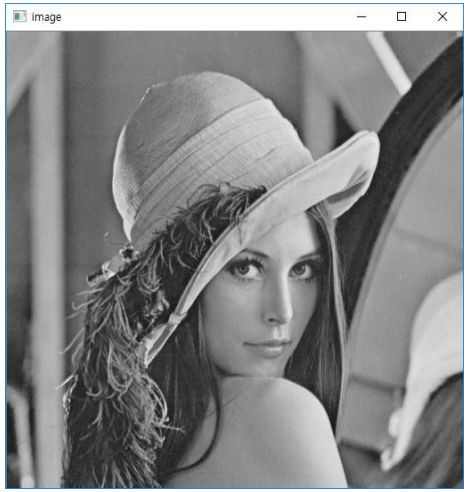


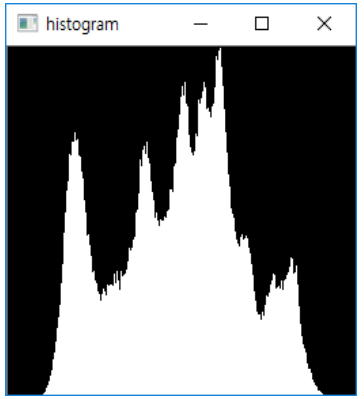




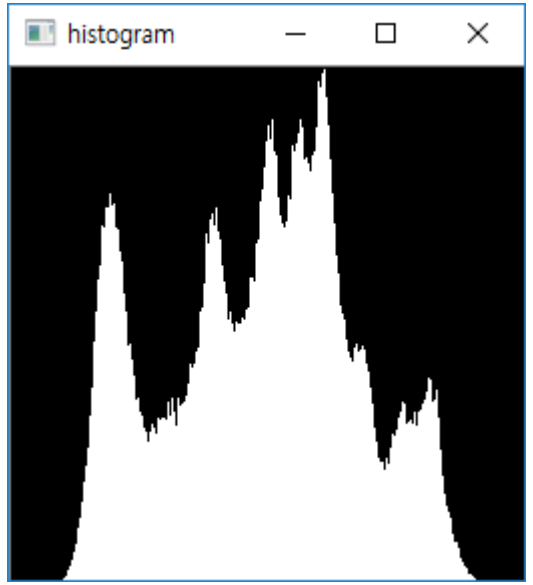
예제: 히스토그램 그리기


```
import cv2
import numpy as np
import math

def main():
    filename = "lena.jpg"
    img = cv2.imread(filename, cv2.IMREAD_GRAYSCALE)
    cv2.imshow('image', img)

    # Create Histogram
    Hist = np.zeros((256)) # 0으로 초기화
   maxValue = 0
    ysize = img.shape[0]
    xsize = img.shape[1]
    for y in range(ysize):
        for x in range(xsize):
            Hist[img.item(y,x)] = Hist[img.item(y,x)] + 1
            # Calculate maxValue for normalization
            if(Hist[img.item(y,x)] > maxValue):
                maxValue = Hist[img.item(y,x)]
```





```
    # Create Null Image
    imgHist = np.zeros((256, 256), dtype="uint8")
    # Draw the histogram on imgHist
    for i in range(256):
        value = Hist[i]
        normalized = math.floor(value * 255 / maxValue)
        for j in range(255, 255-normalized, -1):
            imgHist.itemset((j, i), 255)

    cv2.imshow('histogram', imgHist)
    cv2.waitKey(0)
    cv2.destroyAllWindows()

main()
```

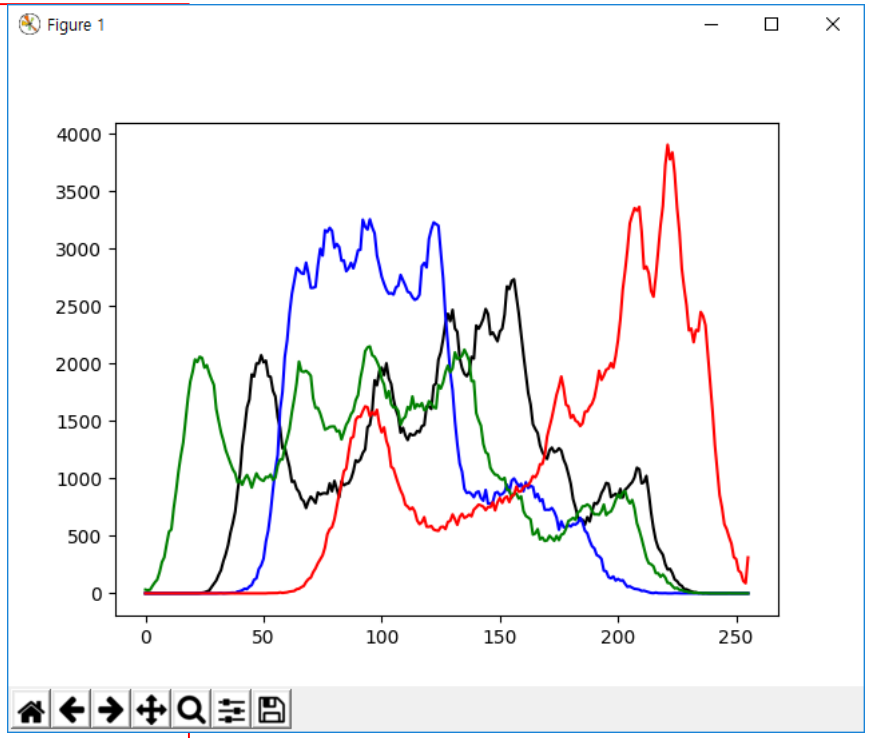


cv2.calcHist()

- **cv.calcHist(images, channels, mask, histSize, ranges[, hist[, accumulate]]) -> hist**
  - images : 입력 영상 list
  - channels : 히스토그램을 구할 채널을 나타내는 list
    - list이므로 반드시 [ ]로 지정해야 함
  - mask : 마스크 영상
    - 입력 영상 전체에서 히스토그램을 구하려면 None 지정
  - histSize : 히스토그램 각 차원의 크기(빈(bin)의 개수)를 나타내는 list
    - list이므로 반드시 [ ]로 지정해야 함
  - ranges : 히스토그램 각 차원의 최솟값과 최댓값으로 구성된 list
  - hist : 계산된 히스토그램(ndarray)
  - accumulate : 기존의 hist 히스토그램에 누적하려면 True, 새로 만들려면 False

continue…

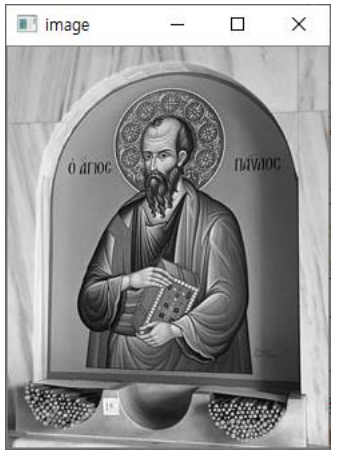
```python
import cv2
import numpy as np
import matplotlib.pyplot as plt

def showHistogram():
    filename = "lena.jpg"

    gray_img = cv2.imread(filename, cv2.IMREAD_GRAYSCALE)
    # 히스토그램 계산
    hist = cv2.calcHist([gray_img], [0], None, [256], [0,256])
    # matplotlib로 히스토그램 그리기
    plt.plot(hist, color='black')
    #plt.show()
    color_img = cv2.imread(filename, cv2.IMREAD_COLOR)
    for i, c in enumerate(('blue', 'green', 'red')):
        # 히스토그램 계산
        hist = cv2.calcHist([color_img], [i], None, [256], [0,256])
        # matplotlib로 히스토그램 그리기
        plt.plot(hist, color=c)
    plt.show()
    cv2.waitKey(0)
    cv2.destroyAllWindows()

showHistogram()
```




```
cv2.calcHist(images, channels, mask, histSize, ranges)
```

예제 6.3.1 영상 히스토그램 계산 – 08.calc_histogram_opencv.py

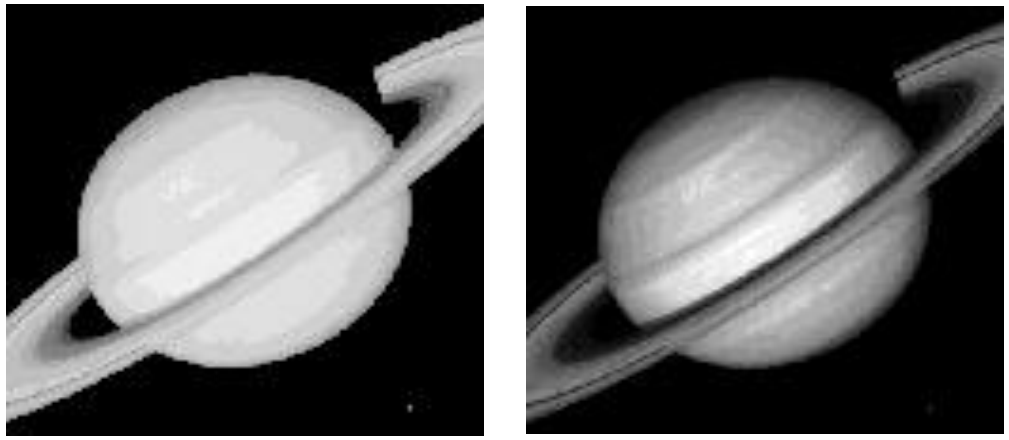
```
01  import numpy as np, import cv2
02
03  def calc_histo(image, histSize, ranges=[0, 256]): …                # 소스 내용 생략
12
13  image = cv2.imread("images/pixel_test.jpg", cv2.IMREAD_GRAYSCALE)   # 영상 읽기
14  if image is None: raise Exception("영상파일 읽기 오류")
15
16  histSize, ranges = [32], [0, 256]                                   # 히스토그램 간격수, 값 범위
17  gap = ranges[1]/histSize[0]                                         # 계급 간격
18  ranges_gap = np.arange(0, ranges[1]+1, gap)                         # 넘파이 계급범위&간격
19  hist1 = calc_histo(image, histSize, ranges)                         # User 함수
20  hist2 = cv2.calcHist([image], [0], None, histSize, ranges)          # OpenCV 함수
21  hist3, bins = np.histogram(image, ranges_gap )                      # numpy 모듈 함수
22
23  print("User 함수: \n", hist1.flatten())                             # 1차원 행렬 1행 표시
24  print("OpenCV 함수: \n", hist2.flatten())
25  print("numpy 함수: \n", hist3)
```



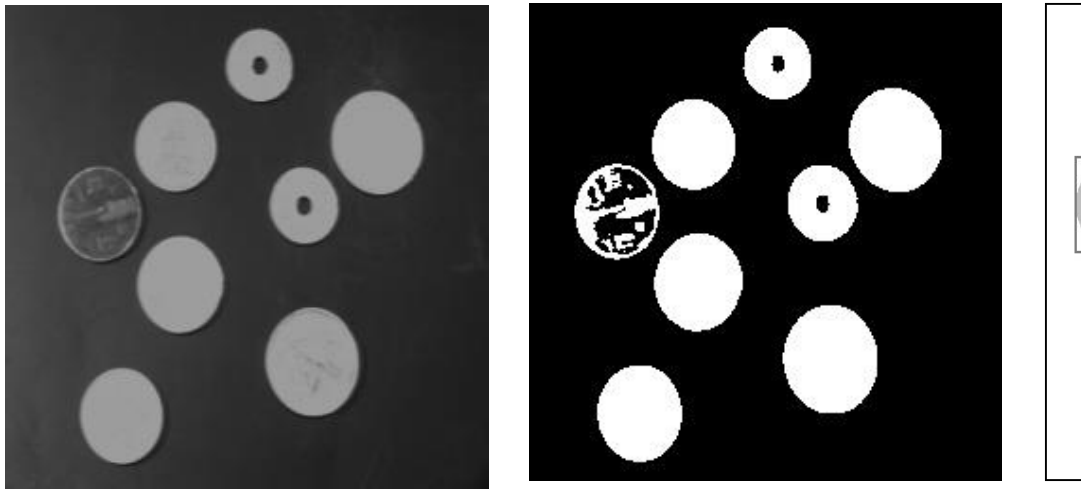

```
Run: 08.calc_histogram_opencv
C:\Python\python.exe D:/source/chap06/08.calc_histogram_opencv.py
User 함수:
[  97.  247.  563. 1001. 1401. 1575. 1724. 1951. 2853. 3939. 3250. 2549.
 2467. 2507. 2402. 2418. 2727. 3203. 3410. 3161. 2985. 2590. 3384. 4312.
 4764. 3489. 2802. 2238. 1127.  628.  199.   37.]
OpenCV 함수:
[  97.  247.  563. 1001. 1401. 1575. 1724. 1951. 2853. 3939. 3250. 2549.
 2467. 2507. 2402. 2418. 2727. 3203. 3410. 3161. 2985. 2590. 3384. 4312.
 4764. 3489. 2802. 2238. 1127.  628.  199.   37.]
numpy 함수:
[  97  247  563 1001 1401 1575 1724 1951 2853 3939 3250 2549 2467 2507
 2402 2418 2727 3203 3410 3161 2985 2590 3384 4312 4764 3489 2802 2238
 1127  628  199   37]
```

히스토그램의 용도

- ❖ 화질향상
- ❖ 물체인식



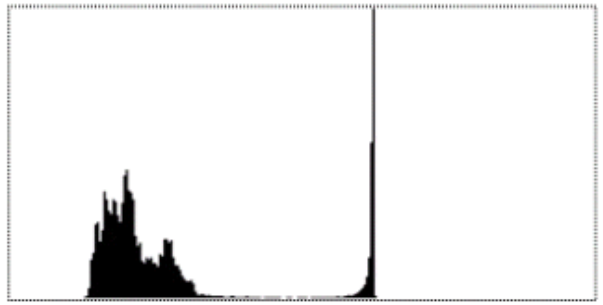
화질 향상



물체 인식 (이치화 후, 255값을 가지는 영역 및 위치를 자동추출)

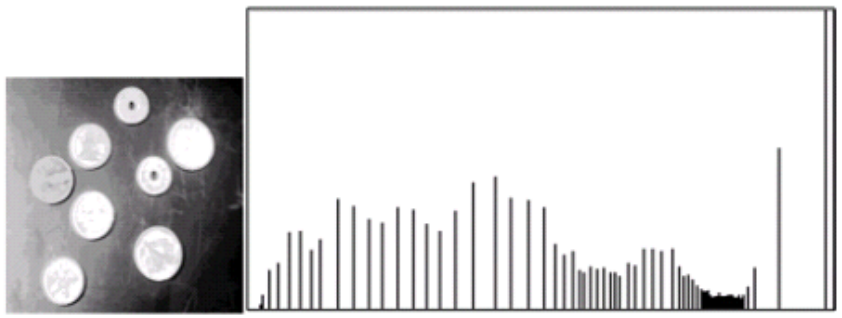
화질향상

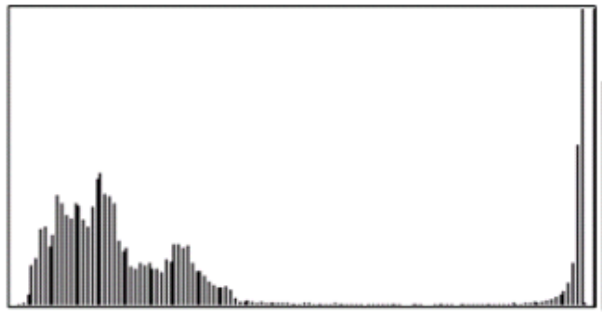
픽셀들이 특정 밝기
영역에 몰려있음

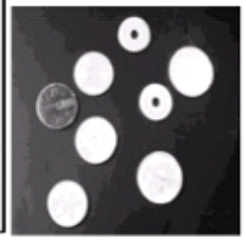


Histogram
Equalization

Histogram
stretching





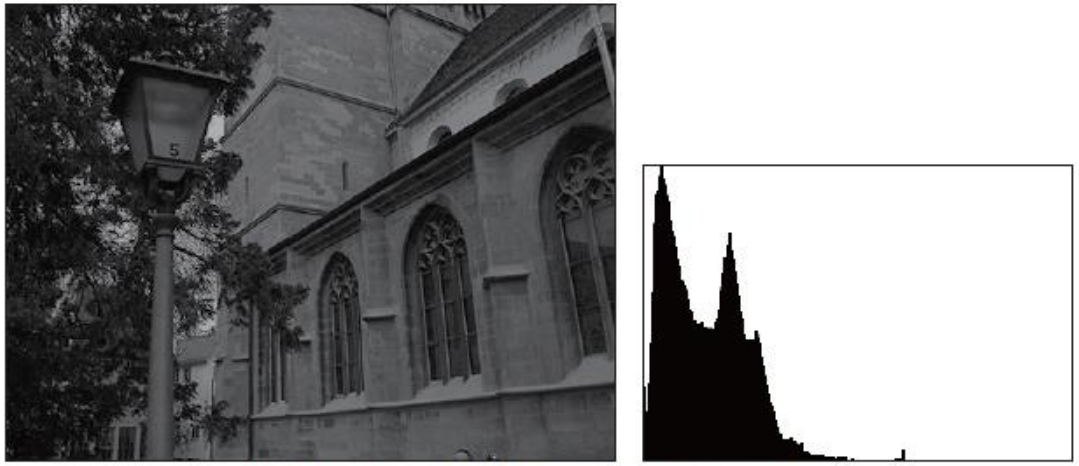


픽셀들이 여러 밝기 영역으로
넓게 퍼지며 분포하게 됨

양쪽으로 일정하게 당긴 것에 불과

❖ 히스토그램의 분포가 좁아서 영상의 대비가 좋지 않은 영상

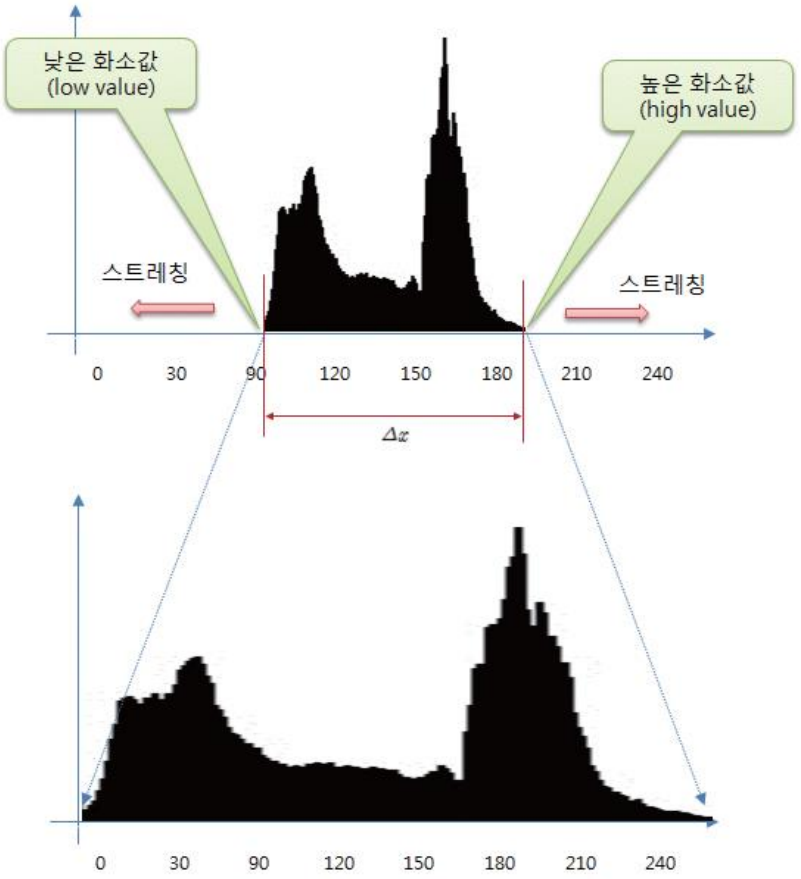
(a) 밝은 부분을 많이 분포하는 영상과 히스토그램



(b) 어두운 부분을 많이 분포하는 영상과 히스토그램

히스토그램 스트레칭

❖ 명암 분포가 좁은 히스토그램을 좌우로 잡아당겨(스트레칭해서) 고른 명암 분포를 가진 히스토그램이 되게 하는 것

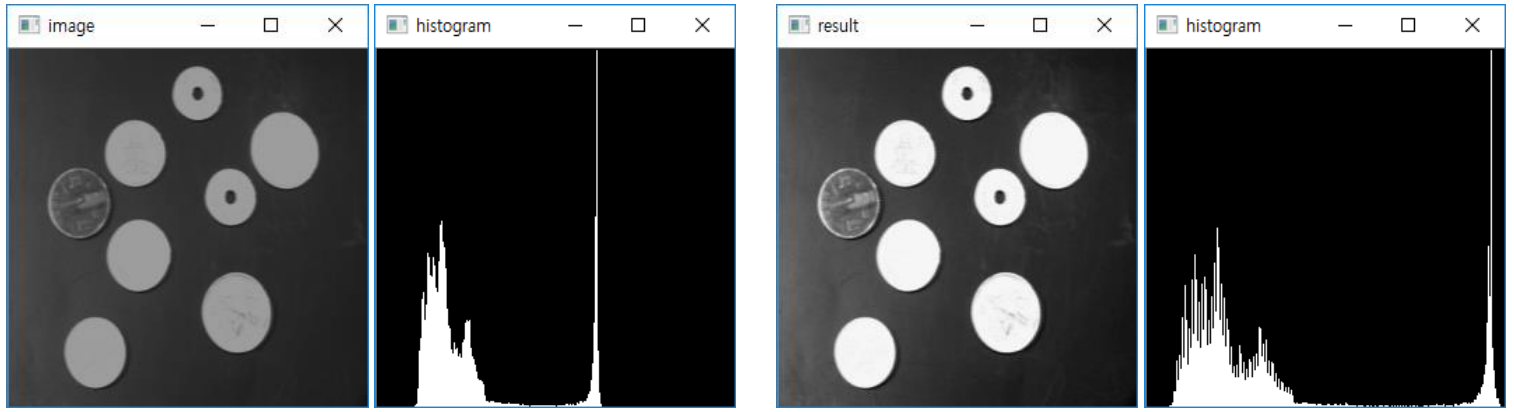


$$\text{새 화소값} = \frac{(\text{화소값} - low)}{high - low} * 255$$

예제: 히스토그램 스트레칭

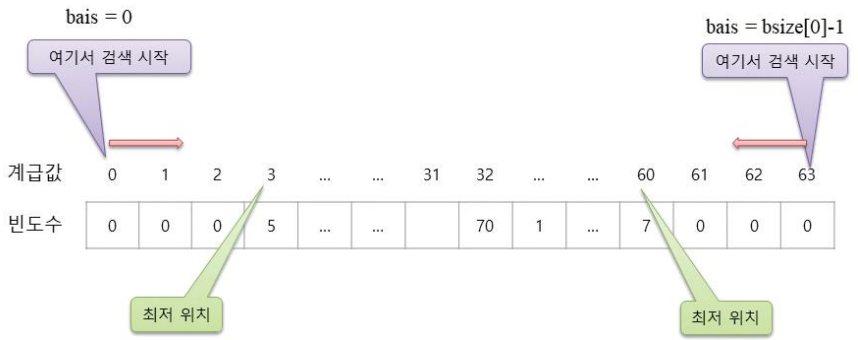
```
import cv2
import numpy as np

def main():
    filename = "coin.jpg"
    img = cv2.imread(filename, cv2.IMREAD_GRAYSCALE)
    cv2.imshow('image', img)
    # Create Histogram
    Hist = np.zeros((256)) # 0으로 초기화
    ysize = img.shape[0]
    xsize = img.shape[1]
    for y in range(ysize):
        for x in range(xsize):
            Hist[img.item(y,x)] = Hist[img.item(y,x)] + 1
    # Calculate low & high values of Histogram range
    low = 0
    high = 255
    for i in range(256):
        if Hist[i] != 0:
            low = i
            break
```

```
    for i in range(255, -1, -1):
        if Hist[i] != 0:
            high = i
            break

    # Convert the image using normHist
    for y in range(ysize):
        for x in range(xsize):
            value = round((img.item(y, x) - low)/(high - low) * 255)
            img.itemset((y, x), value)

    cv2.imshow('result', img)
    cv2.waitKey(0)
    cv2.destroyAllWindows()

main()
```



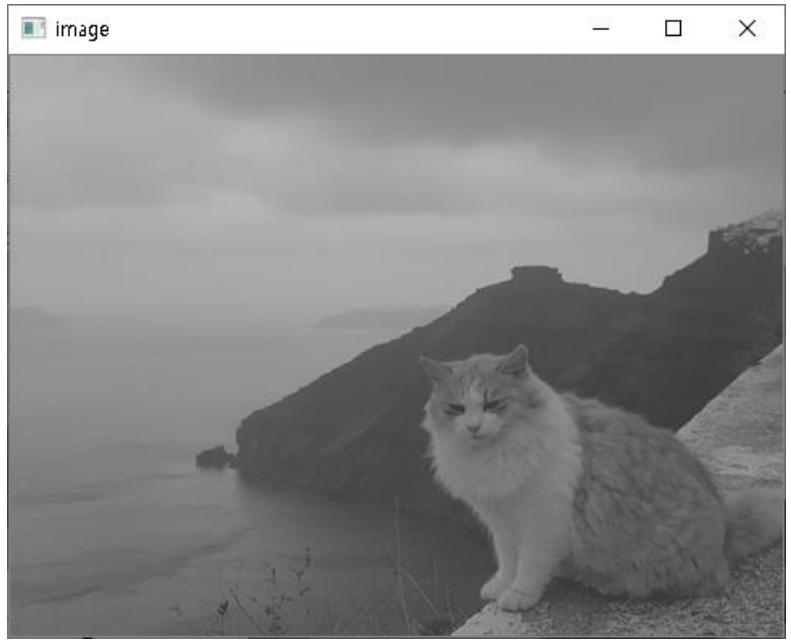
예제 6.3.5 히스토그램 스트래칭 – histogram_stretching.py

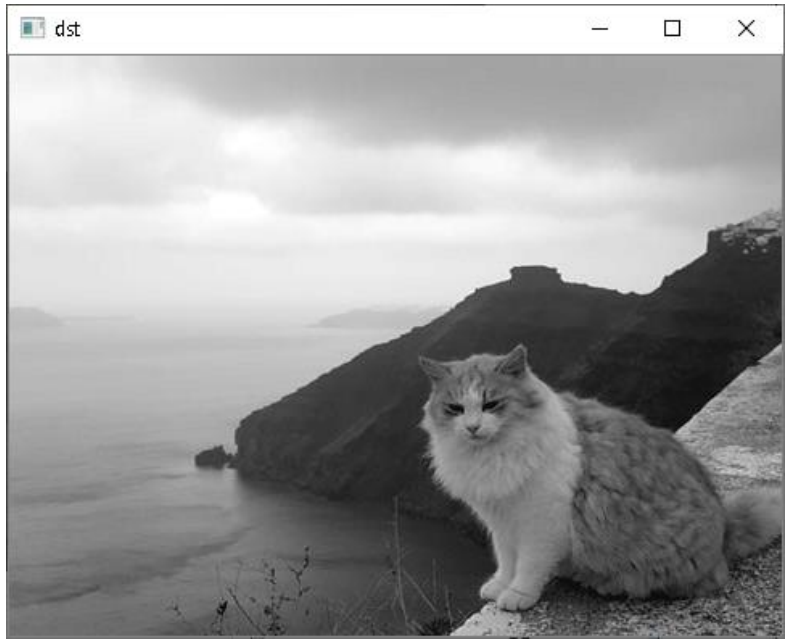
```
01  import numpy as np, cv2
02  from Common.histogram import draw_histo              # 함수 재사용 위한 임포트
03
04  def search_value_idx(hist, bias=0):                  # 값있는 첫 계급 검색 함수
05      for i in range(hist.shape[0]):
06          idx = np.abs(bias - i)                       # 검색 위치(처음 또는 마지막)
07          if hist[idx] > 0: return idx                 # 위치 반환
08      return -1                                        # 대상 없으면 반환
09
10  image = cv2.imread("images/hist_stretch.jpg", cv2.IMREAD_GRAYSCALE)
11  if image is None: raise Exception("영상파일 읽기 오류")
12
13  bsize, ranges = [64], [0,256]                        # 계급 개수 및 화소 범위
14  hist = cv2.calcHist([image], [0], None, bsize, ranges)
15
16  bin_width = ranges[1]/bsize[0]                       # 한 계급 너비
17  low = search_value_idx(hist, 0) * bin_width          # 최저 화소값
18  high= search_value_idx(hist, bsize[0] - 1) * bin_width   # 최고 화소값
19
20  idx = np.arange(0, 256)                              # 룩업 인덱스(0~255) 생성
21  idx = (idx - low)/(high- low) * 255                  # 수식 적용하여 룩업 인덱스 완성
22  idx[0:int(low)] = 0                                  # 히스토그램 하위 부분
23  idx[int(high+1):] = 255                              # 히스토그램 상위 부분
24
25  dst = cv2.LUT(image, idx.astype('uint8'))            # 룩업 테이블 사용
26  ## 룩업 테이블 사용하지 않고 직접 구현
27  # dst = np.zeros(image.shape, dtype=image.dtype)
28  # for i in range(dst.shape[0]):
29  #         for j in range(dst.shape[1]):
30  #             dst[i,j] = idx[image[i,j]]
31
32  hist_dst = cv2.calcHist([dst], [0], None, bsize, ranges)# 결과 영상 히스토그램 재계산
33  hist_img = draw_histo(hist, (200,360))               # 원본 영상 히스토그램 그리기
34  hist_dst_img = draw_histo(hist_dst, (200,360))       # 결과 영상 히스토그램 그리기
35
36  print("high_vlue =", high)
37  print("low_vlue =", low)
38  cv2.imshow("image", image);     cv2.imshow("hist_img", hist_img)
39  cv2.imshow("dst", dst);         cv2.imshow("hist_dst_img", hist_dst_img)
40  cv2.waitKey(0)
```

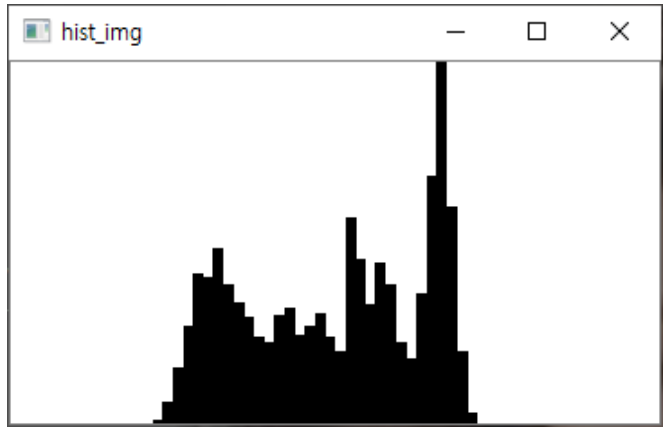


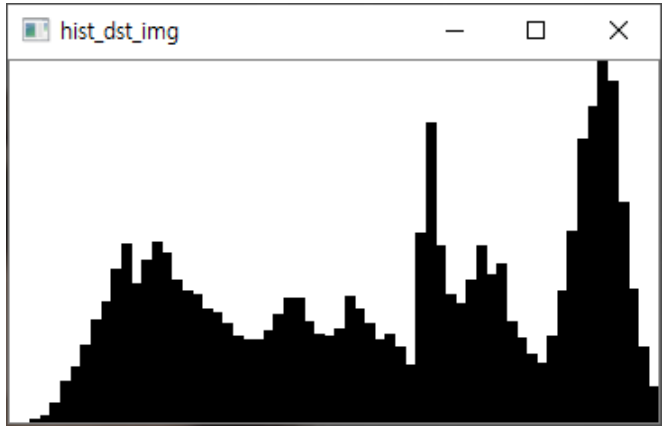
❖ 실행결과

```
Run:  11.histogram_stretching
C:\Python\python.exe D:/source/chap06/11.histogram_stretching.py
high_value =  180.0
low_value =  52.0
```









히스토그램 평활화(Histogram Equalization)

❖ 이퀄라이저

- 주파수 특성을 균등하게 보정하는 기기
- 주파수 특성을 어느 특정의 목적에 맞추어 임의로 변화시켜 원하는 음색을 얻어냄

❖ 평활화 알고리즘

- 히토그램 평활화의 사전적 의미인 “분포의 균등”이라는 방법을 이용해 명암 대비 증가
- 영상의 인지도 높이며, 영상의 화질 개선

① 영상의 히스토그램을 계산한다.

② 히스토그램 빈도값에서 누적 빈도수(누적합)를 계산한다.

③ 누적 빈도수를 정규화(정규화 누적합) 한다.

④ 결과 화소값 = 정규화 누적합 * 최대 화소값

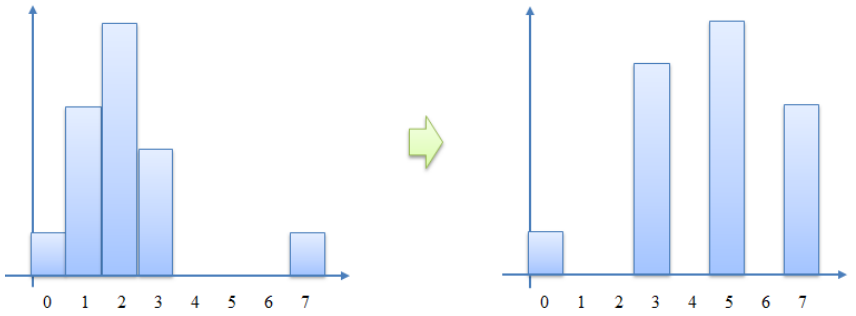
❖ 평활화 과정 예시

0	2	2	1
1	2	3	2
1	2	3	2
1	3	1	7

입력 영상 화소값

0	5	5	3
3	5	7	5
3	5	7	5
3	7	3	7

평활화 완료 영상 화소값



화소값	0	1	2	3	4	5	6	7
빈도수	1	5	6	3	0	0	0	1
누적 빈도수	1	6	12	15	15	15	15	16
정규화누적합	1/16	6/16	12/16	15//16	15//16	15/16	15/16	16/16
	0.0625	0.375	0.75	0.9375	0.9375	0.9375	0.9375	1
누적합 *최댓값	0.4375	2.625	5.25	6.5625	6.5625	6.5625	6.5625	7
평활화 결과	0	3	5	7	7	7	7	7

<그림 6.3.5> 평활화 계산 과정 예시

H(i): 누적 히스토그램
h(i): 정규화합 히스토그램

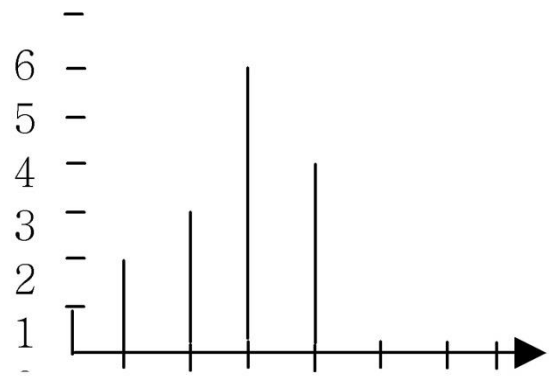
N_t : 전체 픽셀수
G_{max} : 최대밝기값

$$h(i) = \frac{G_{\max}}{N_t} H(i)$$

❖ 평활화 과정 예시(2)

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

src

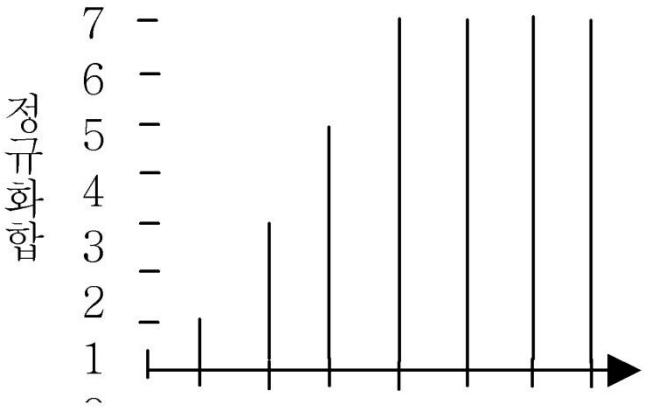


H(i): 누적 히스토그램
h(i): 정규화합 히스토그램

N_t : 전체 픽셀수
G_{max} : 최대밝기값

$$h(i) = \frac{G_{max}}{N_t} H(i)$$

	누적합	정규화합
0	1	0.44
1	3	1.31
2	6	2.62
3	12	5.25
4	16	7.0
5	16	7.0
6	16	7.0
7	16	7.0



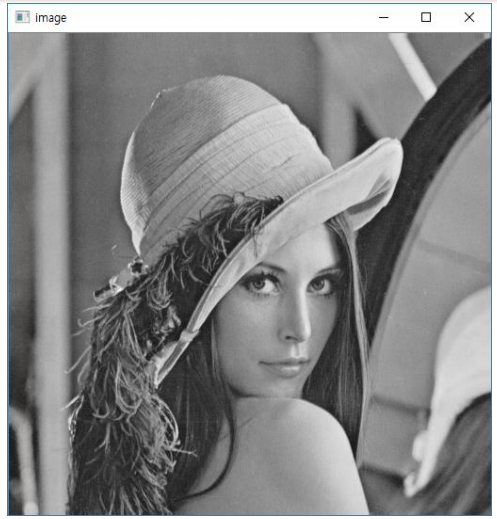
5	7	5	7
7	5	5	3
7	3	1	5
1	0	3	5

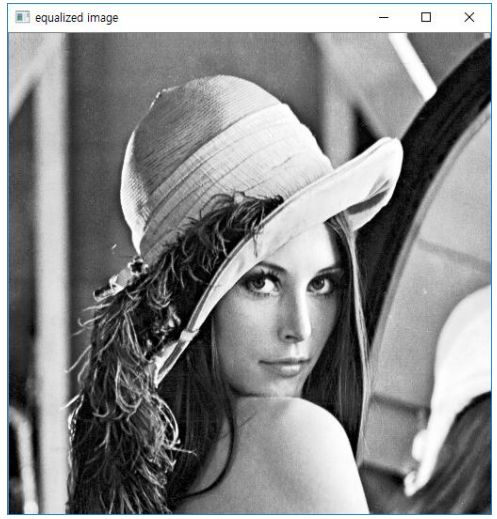
dst

cv2.equalizeHist()

```
import cv2
import numpy as np

def showImage():
    filename = "lena.jpg"
    img = cv2.imread(filename, cv2.IMREAD_GRAYSCALE)
    cv2.imshow('image', img)

    equ_img = cv2.equalizeHist(img)
    cv2.imshow('equalized image', equ_img)

    cv2.waitKey(0)
    cv2.destroyAllWindows()

showImage()
```

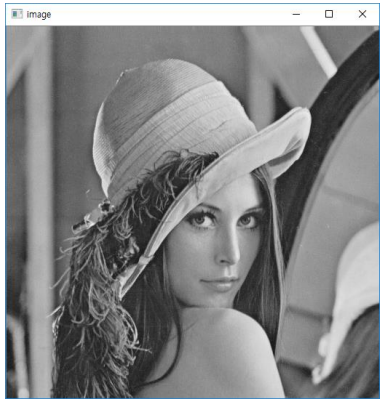
또는....

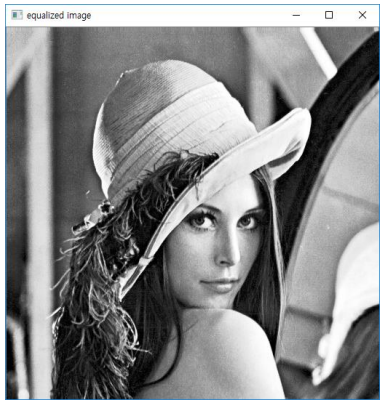




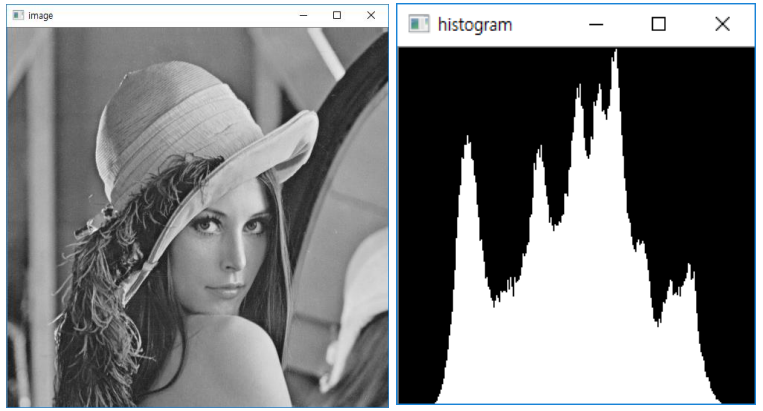
continue…

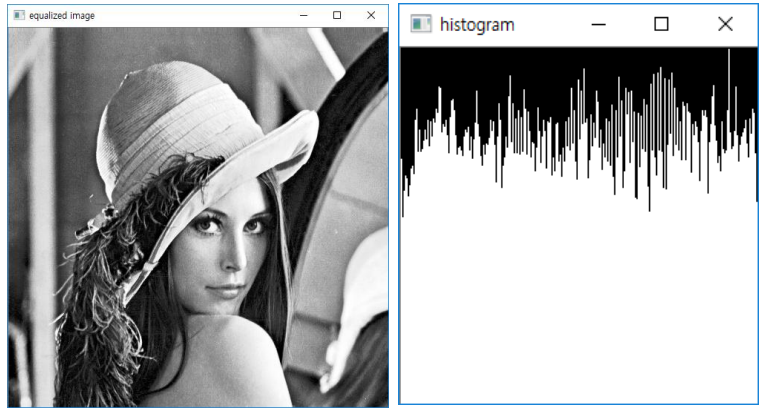
```
import cv2
import numpy as np

def main():
    filename = "lena.jpg"
    img = cv2.imread(filename, cv2.IMREAD_GRAYSCALE)
    cv2.imshow('image', img)

    # Create Histogram
    Hist = np.zeros((256)) # 0으로 초기화
    ysize = img.shape[0]
    xsize = img.shape[1]
    for y in range(ysize):
        for x in range(xsize):
            Hist[img.item(y,x)] = Hist[img.item(y,x)] + 1

    # Calculate Normalized Sum
    normHist = np.empty((256))
    sum = 0.0
    factor = 255.0 / (ysize * xsize)
    for i in range(256):
        sum = sum + Hist[i]
        normHist[i] = round(sum * factor)
```





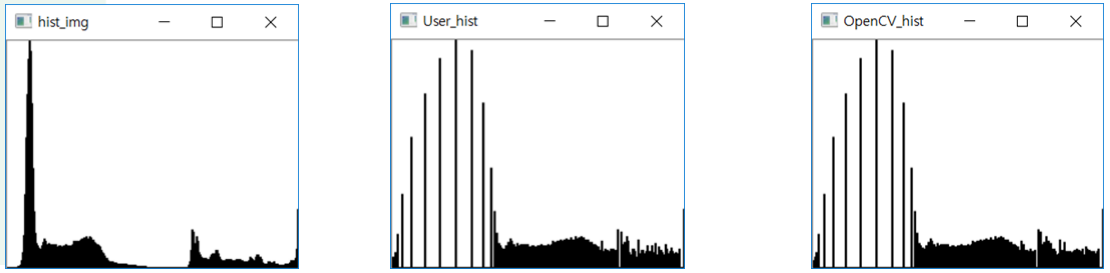
```
    # Convert the image using normHist
    for y in range(ysize):
        for x in range(xsize):
            img.itemset((y, x), normHist[img.item(y, x)])

    cv2.imshow('result', img)
    # cv2.imwrite('lena_eq.jpg', img)
    cv2.waitKey(0)
    cv2.destroyAllWindows()

main()
```





예제 6.3.6 히스토그램 평활화 – histogram_equalize.py

```
01  import numpy as np, cv2
02  from Common.histogram import draw_histo                    # 히스토
03
04  image = cv2.imread("images/equalize_test.jpg", cv2.IMREAD_GRAY
05  if image is None: raise Exception("영상파일 읽기 오류")
06
07  bins, ranges = [256], [0, 256]
08  hist = cv2.calcHist([image], [0], None, bins, ranges)
09
10  ## 히스토그램 누적합 계산
11  accum_hist = np.zeros(hist.shape[:2], np.float32)
12  accum_hist[0] = hist[0]
13  for i in range(1, hist.shape[0]):
14      accum_hist[i] = accum_hist[i - 1] + hist[i]
15
16  accum_hist = (accum_hist / sum(hist)) * 255                # 누적합
17  dst1 = [[accum_hist[val] for val in row] for row in image] # 화소값
18  dst1 = np.array(dst1, np.uint8)
19
20  ##numpy 함수 및 OpenCV 룩업 테이블 사용
21  # accum_hist = np.cumsum(hist)                             # 누적합 계산
22  # cv2.normalize(accum_hist, accum_hist, 0, 255, cv2.NORM_MINMAX)   # 정규화
23  # dst1 = cv2.LUT(image, accum_hist.astype('uint8'))        # 룩업 테이블로 화소값 할당
24
25  dst2 = cv2.equalizeHist(image)                             # OpenCV 히스토그램 평활화
26  hist1 = cv2.calcHist([dst1], [0], None, bins, ranges)      # 히스토그램 계산
27  hist2 = cv2.calcHist([dst2], [0], None, bins, ranges)
28  hist_img = draw_histo(hist)
29  hist_img1 = draw_histo(hist1)
30  hist_img2 = draw_histo(hist2)
31
32  cv2.imshow("image", image);          cv2.imshow("hist_img", hist_img)
33  cv2.imshow("dst1_User", dst1);       cv2.imshow("User_hist", hist_img1)
34  cv2.imshow("dst2_OpenCV", dst2);     cv2.imshow("OpenCV_hist", hist_img2)
35  cv2.waitKey(0)
```



4

컬러공간 변환

Color의 개요

❖ 컬러 표현

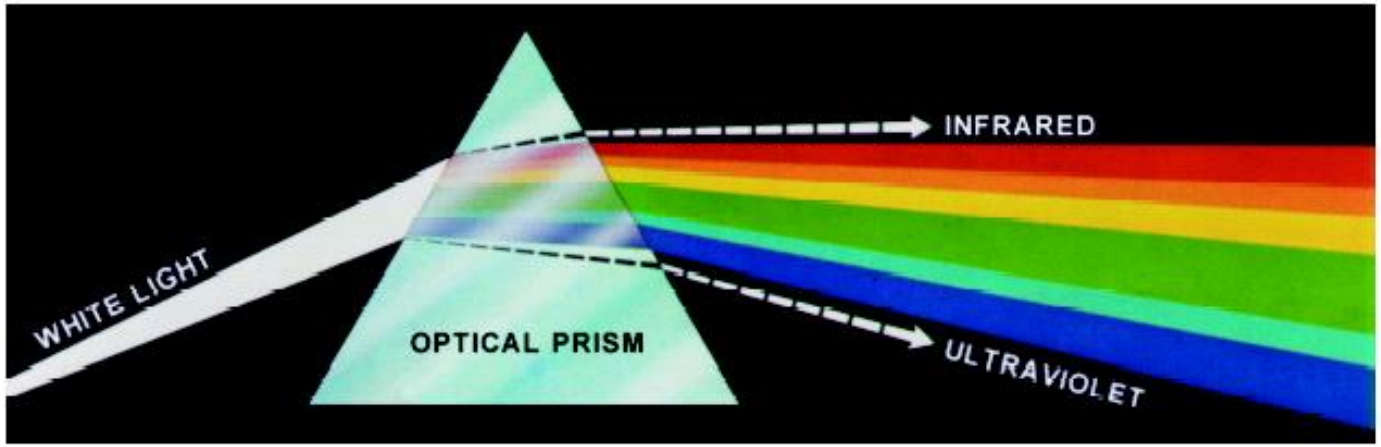
- 1802년, Tomas Young이 제안한 3색 이론
- 컬러는 세가지 기본 컬러를 적당한 비율로 조합하여 만들어진다고 주장(사람의 색인식구조와 일치)

$$C = aC_1 + bC_2 + cC_3$$

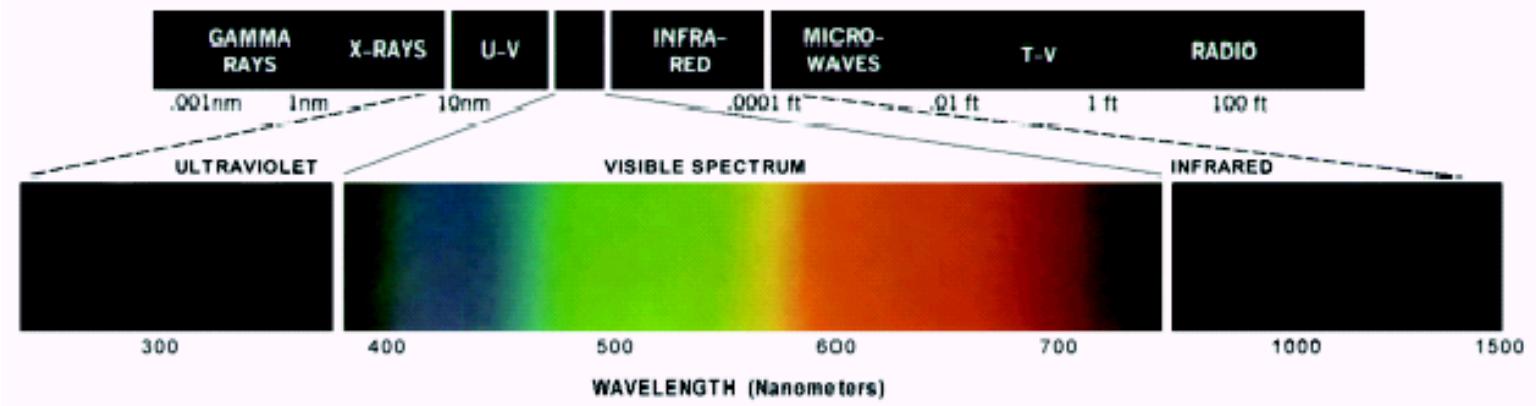
❖ 삼중자극(tristimulus)

- 인간의 컬러 인지 능력은 세가지 추상체들의 반응에 의해 나타남
  - 휴먼 망막(retina) 내부: 색을 감지하는 추상체(cone)이 3개 존재
  - 추상체: 빛을 감지하는 3개의 센서로 각각 적(red),녹(green),청(blue)영역 감지

❖ 컬러 스펙트럼(spectrum)



❖ 가시광선 영역(약 380 nm~780 nm사이의 파장)



컬러모델

❖ 컬러모델

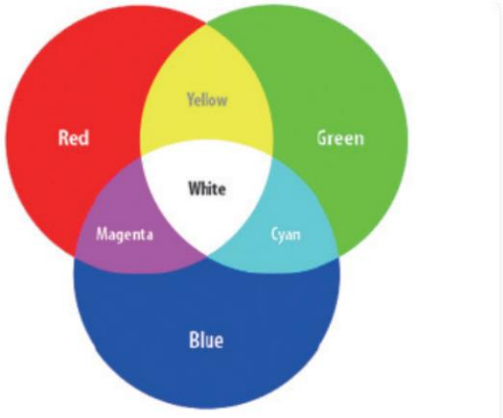
- 색상을 표현하는 체계
- 색 표시계(color system)의 모든 색들을 색 공간에서 3차원 좌표로 표현한 것
- 색 표시계 - RGB, CMY, HSV, LAB, YUV 등의 색 체계
- 공간상의 좌표로 표현 -> 어떤 컬러와 다른 컬러들과의 관계를 표현하는 논리적인 방법제공

❖ 종류

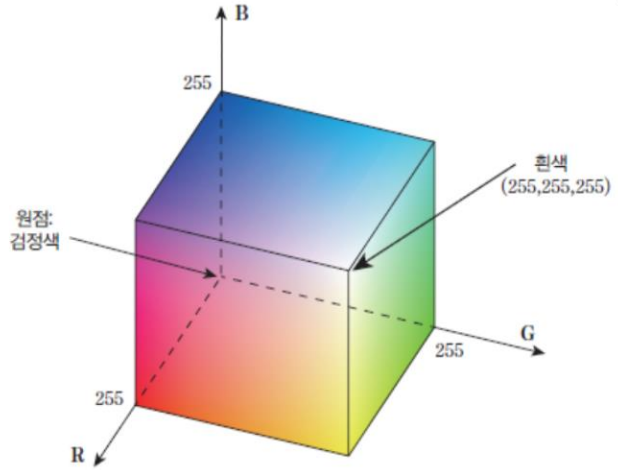
- OpenCV에서는 150여개의 컬러모델을 지원
- 주요 컬러모델
  - RGB
  - CMY
  - HSI
  - YIQ

RGB모델

- 가산 혼합(additive mixture) – 빛을 섞을 수록 밝아짐
- 빛을 이용해서 색 생성
- 빛의 삼원색(빨강 빛, 초록 빛, 파랑 빛) 사용
- 모니터, 텔레비전, 빔 프로젝터와 같은 디스플레이 장비들에서 기본 컬러 공간으로 사용

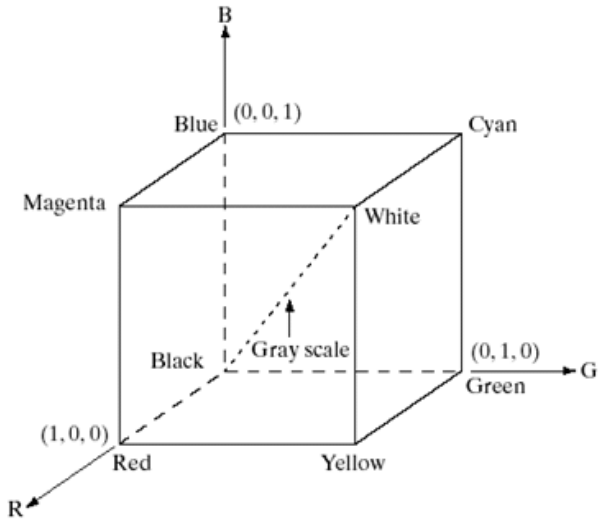


〈그림 6.4.5〉 가산 혼합

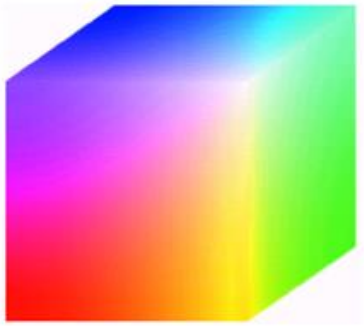


〈그림 6.4.6〉 RGB 컬러 공간

- 각각의 좌표축은 R,G,B축을 나타냄
- 좌표 (0,0,0): 검은색
- 좌표 (1,0,0): 빨강
- (0,0,0)-(1,1,1) 연결 대각선: R,G,B 비율이 동일한 회색(gray) 등급



명암도 = 0.299R + 0.587G + 0.114B
명암도 = 0.333R + 0.333G + 0.333B

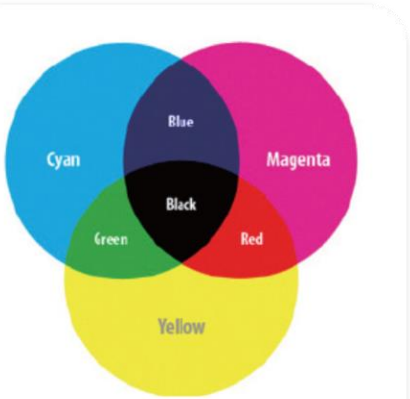


〈표 6.4.1〉 대표적인 색상에 대한 RGB 표현

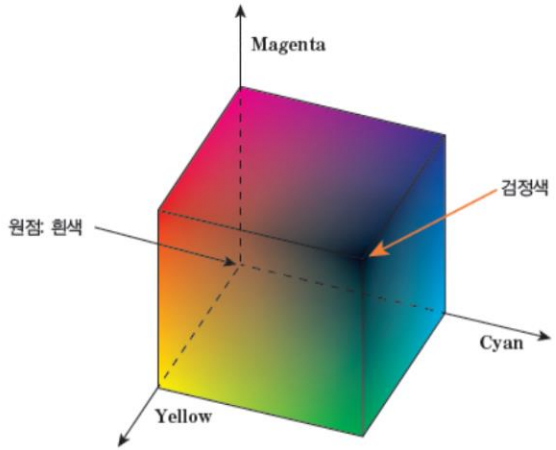
RGB 화소값	색상	RGB 화소값	색상
0, 0, 0	white	240, 230, 140	khaki
255, 255, 255	black	238, 130, 238	violet
128, 128, 128	gray	255, 165, 0	orange
192, 192, 192	silver	255, 215, 0	gold
255, 0, 0	red	0, 0, 128	navy
0, 255, 0	green	160, 32, 240	purple
0, 0, 255	blue	0, 128, 128	olive
255, 255, 0	yellow	75, 0, 130	indigo
255, 0, 255	magenta	255, 192, 203	pink
0, 255, 255	cyan	135, 206, 235	skyblue

CMY 모델

- ‘색’의 삼원색이며, 청록(Cyan), 자홍(Magenta), 노랑(Yellow)로 구성
- RGB모형과 반대의 공간, C,M,Y는 각기 R,G,B의 보색(complement)
- 빼기 삼원색(subtractive primaries): 백색광에서 특정색을 뺌에 의해 원하는 색깔을 만듬
- 컬러 복사기, 프린트와 같은 출력장치에 사용



〈그림 6.4.7〉 감산 혼합



〈그림 6.4.8〉 CMY 컬러 공간

C = 255 − R	R = 255 − C
M = 255 − G	G = 255 − M
Y = 255 − R	B = 255 − Y

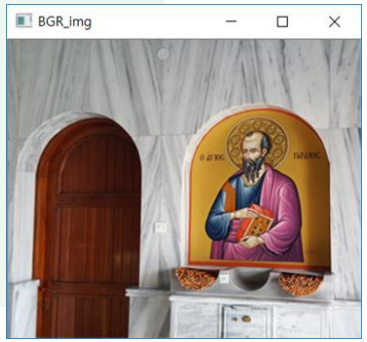
❖ CMYK 모델

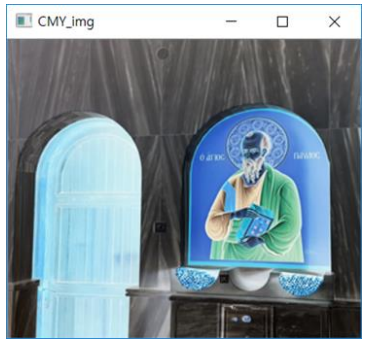
- CMY 모델 + 검정색(black) K = CMYK
- 검정색을 만들기 위해 CMY를 조합하는 것은 문제
- 비용 증가, 검정색의 질적 수준이 떨어지는 것을 방지
- 실용적 프린트 모델

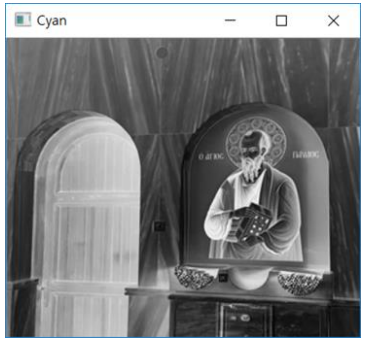
$$C = 1-R$$
$$M = 1-G$$
$$Y = 1-B$$
$$K = \min(C,M,Y)$$

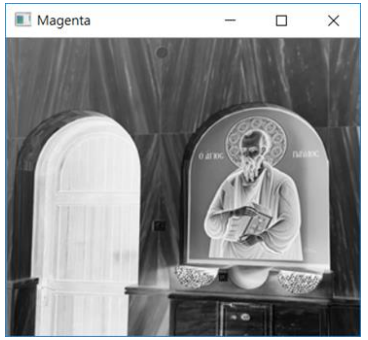
예제 6.4.1 컬러 공간 변환(BGR→CMY) - convert_CMY.py

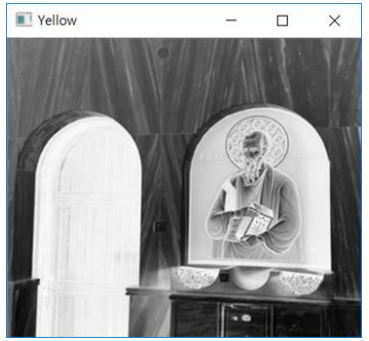
```
01  import numpy as np, cv2
02
03  BGR_img = cv2.imread("images/color_model.jpg", cv2.IMREAD_COLOR) # 컬러 영상 읽기
04  if BGR_img is None: raise Exception("영상파일 읽기 오류")
05
06  white = np.array([255, 255, 255], np.uint8)
07  CMY_img = white - BGR_img
08  Yellow, Magenta, Cyan = cv2.split(CMY_img)                    # 채널 분리
09
10  titles = ['BGR_img', 'CMY_img', 'Yellow', 'Magenta', 'Cyan']
11  for t in titles: cv2.imshow(t, eval(t))
12  cv2.waitKey(0)
```











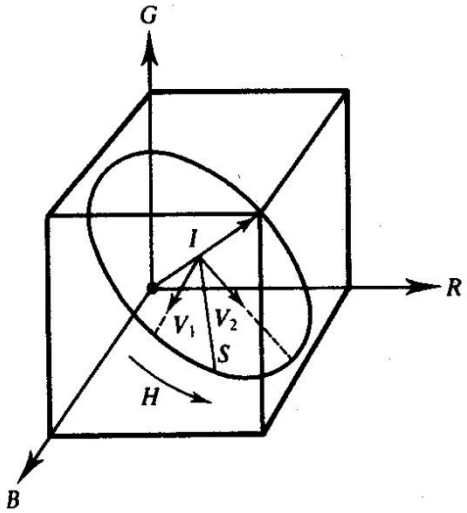
예제 6.4.2 컬러 공간 변환(BGR→CMYK) – 14.conver_CMYK.py

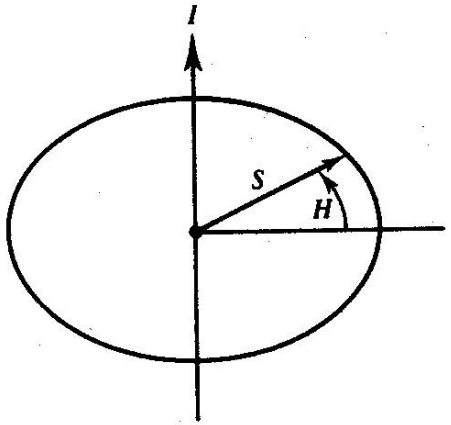
```
01  import numpy as np, cv2
02
03  BGR_img = cv2.imread("images/color_model.jpg", cv2.IMREAD_COLOR)      # 컬러 영상 읽기
04  if BGR_img is None: raise Exception("영상파일 읽기 오류")
05
06  white = np.array([255, 255, 255], np.uint8)
07  CMY_img = white - BGR_img
08  CMY = cv2.split(CMY_img)                                              # 채널 분리
09
10  black = cv2.min(CMY[0], cv2.min(CMY[1], CMY[2]))                      # 원소 간의 최솟값 저장
11  Yellow, Magenta, Cyan = CMY - black                                   # 3개 행렬 화소값 차분
12
13  titles = ['black', 'Yellow', 'Magenta', 'Cyan']
14  [cv2.imshow(t, eval(t)) for t in titles]                              # 리스트 생성 방식 활용
15  cv2.waitKey(0)
```

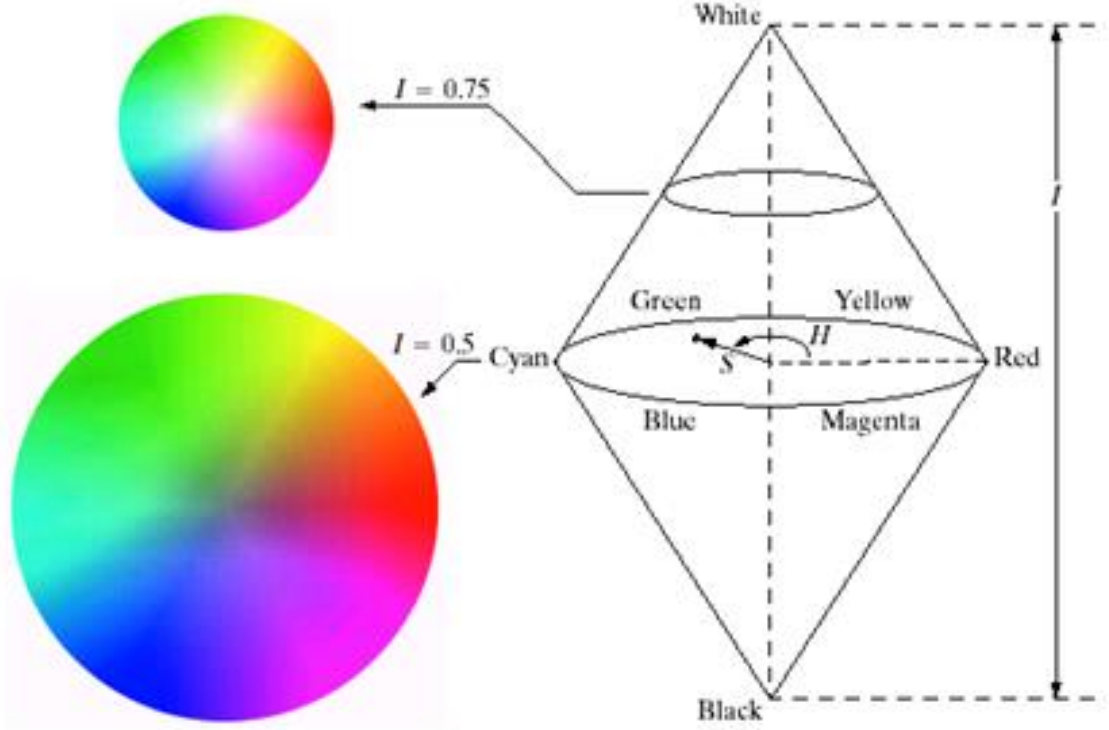
HSI모델

- 인간의 색인지에 기반한 모델
- (RGB,YIQ,CMY,CMYK는 시스템이나 하드웨어에서 사용을 위한 모델)
- 색상(Hue: H), 채도(Saturation: S), 명도(Intensity: I) 모델
- 구체적 컬러를 만들기 위해 색 조합이 불필요(H 좌표값 자체가 바로 색상값)

  - H(색상) : 0~360° 범위, 빨강,파랑, 노랑 등의 색을 부별하는 축
  - S(채도) : 0~1 범위, 순색에 첨가된 백색광의 비율
  - I(명도) : 0~1의 범위,0은 검정, 1은 흰색













원본 hue saturation intensity

❖ RGB → HSI 변환 수식

$$\theta = \cos -1\left[\frac{((R-G)+(R-B))*0.5}{\sqrt{(R-G)^2+(R-B)\cdot(G-B)}}\right]$$

$$H = \begin{Bmatrix} \theta & , \text{ if } B \le G \\ 360-\theta, & otherwise \end{Bmatrix}$$

$$S = 1 - \frac{3\cdot \min(R,G,B)}{(R+G+B)}$$

$$I = \frac{1}{3}(R+G+B)$$

❖ OpenCV HSV 변환 수식

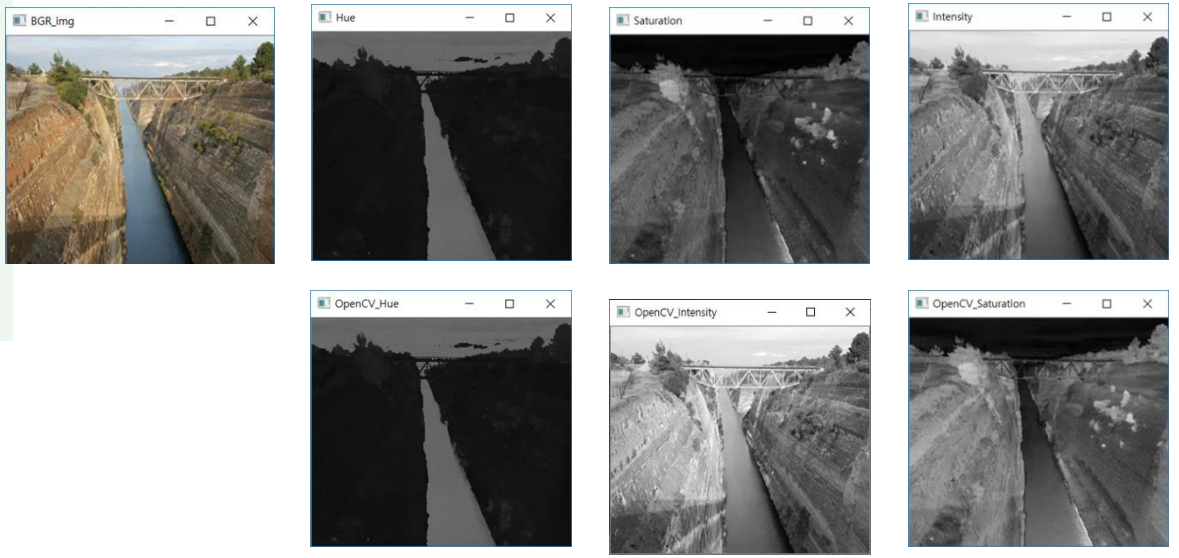
$$H = \begin{cases} \frac{(G-B)*60}{S} & , \text{ if } V = R \\ \frac{(G-B)*60}{S}+120, & \text{if } V = G \\ \frac{(G-B)*60}{S}+240, & \text{if } V = B \end{cases}$$

$$S = \begin{cases} V - \frac{\min(R,G,B)}{V}, & \text{if } V \neq 0 \\ 0 & , otherwise \end{cases}$$

$$V = \max(R,G,B)$$

예제 6.4.3 컬러 공간 변환(BGR→HSV) – 15.conver_HSV.py

```python
import numpy as np, cv2, math

def calc_hsi(bgr):                                          # 한 화소 hsi 계산 함수
    # B, G, R = bgr.astype(float)                           # float 형 변환
    B, G, R = float(bgr[0]), float(bgr[1]), float(bgr[2])   # 속도면에 유리
    bgr_sum = (R + G + B)
    ##    색상 계산
    tmp1 = ((R - G) + (R - B)) * 0.5
    tmp2 = math.sqrt((R - G) * (R - G) + (R - B) * (G - B))
    angle = math.acos(tmp1 / tmp2) * (180 / np.pi) if tmp2 else 0       # 각도

    H = angle if B <= G else 360 - angle                    # 색상
    S = 1.0 - 3 * min([R, G, B]) / bgr_sum if bgr_sum else 0  # 채도
    I = bgr_sum / 3                                         # 명도
    return (H/2, S*255, I)                                  # 3 원소 튜플로 반환

## BGR 컬러→HSI 컬러 변환 함수
def bgr2hsi(image):
    hsv = [[calc_hsi(pixel) for pixel in row] for row in image]      # 2차원 배열 순회
    return cv2.convertScaleAbs(np.array(hsv))

BGR_img = cv2.imread("images/color_space.jpg", cv2.IMREAD_COLOR)        # 컬러 영상 읽기
if BGR_img is None: raise Exception("영상파일 읽기 오류")

HSI_img = bgr2hsi(BGR_img)                                  # BGR→ HSI 변환
HSV_img = cv2.cvtColor(BGR_img, cv2.COLOR_BGR2HSV)          # OpenCV 함수
Hue, Saturation, Intensity = cv2.split(HSI_img)             # 채널 분리
Hue2, Saturation2, Intensity2 = cv2.split(HSV_img)          # 채널 분리

titles = ['BGR_img', 'Hue', 'Saturation', 'Intensity']
[cv2.imshow(t, eval(t)) for t in titles]                    # User 구현 결과 영상표시
[cv2.imshow('OpenCV_'+t, eval(t+'2')) for t in titles[1:]]  # OpenCV 결과 영상 표시
cv2.waitKey(0)
```



기타 컬러 공간

❖ YCbCr 컬러 공간

- 색차 신호(Cr, Cb) 성분을 휘도(Y) 성분보다 상대적으로 낮은 해상도로 구성
  - 인간의 시각에서 화질의 큰 저하 없이 영상 데이터의 용량 감소, 효과적인 영상 압축 가능
  - JPEG이나 MPEG에서 압축을 위한 기본 컬러 공간

$$
\begin{aligned}
Y &= 0.299 \cdot R + 0.587 \cdot G + 0.114 \cdot B \\
Cb &= (R - Y) \cdot 0.564 + 128 \\
Cr &= (B - Y) \cdot 0.713 + 128 \\
\\
R &= Y + 1.403 \cdot (Cr - 128) \\
G &= Y - 0.714 \cdot (Cr - 128) - 0.344 (Cb - 128) \\
B &= Y + 1.773 \cdot (Cb - 128)
\end{aligned}
$$

❖ YUV 컬러 공간

- TV 방송 규격에서 사용하는 컬러 표현 방식
- PAL 방식의 아날로그 비디오를 위해 개발

$$
\begin{aligned}
Y &= +0.2160 \cdot R + 0.7152 \cdot G + 0.0722 \cdot B \\
U &= -0.0999 \cdot R - 0.3360 \cdot G + 0.4360 \cdot B \\
V &= +0.6150 \cdot R - 0.5586 \cdot G - 0.05639 \cdot B \\
\\
R &= Y + 1.28033 \cdot V \\
G &= Y - 0.21482 \cdot U - 0.38059 \cdot V \\
B &= Y + 2.12798 \cdot U
\end{aligned}
$$

컬러모델 바꾸기

❖ OpenCV

- cv2.cvtColor()함수를 이용

❖ cv2.cvtColor()함수의 사용 예

- `hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)`

- 주요모드
  - **cv2.COLOR_BGR2GRAY** **cv2.COLOR_GRAY2BGR**
  - **cv2.COLOR_BGR2HSV** **cv2.COLOR_HSV2BGR**
  - **cv2.COLOR_BGR2LAB** **cv2.COLOR_LAB2BGR**
  - cv2.COLOR_BGR2RGB
  - cv2.COLOR_GRAY2RGB
  - cv2.COLOR_HSV2RGB

〈표 6.4.2〉 컬러 공간 변환을 위한 옵션 상수(cv2. 생략)

옵션 상수	값	옵션 상수	값	옵션 상수	값
COLOR_BGR2BGRA	0	COLOR_BGR2YCrCb	36	COLOR_HSV2BGR	54
COLOR_BGRA2BGR	1	COLOR_RGB2YCrCb	37	COLOR_HSV2RGB	55
COLOR_BGRA2RGB	2	COLOR_YCrCb2BGR	38	COLOR_LAB2BGR	56
COLOR_RGBA2BGR	3	COLOR_YCrCb2RGB	39	COLOR_LAB2RGB	57
COLOR_BGR2RGB ,	4	COLOR_BGR2HSV	40	COLOR_LUV2BGR	58
COLOR_BGRA2RGBA	5	COLOR_RGB2HSV	41	COLOR_LUV2RGB	59
COLOR_BGR2GRAY	6	COLOR_BGR2LAB	44	COLOR_HLS2BGR	60
COLOR_RGB2GRAY	7	COLOR_RGB2LAB	45	COLOR_HLS2RGB	61
COLOR_GRAY2BGR	8	COLOR_BayerBG2BGR	46	COLOR_BGR2YUV	82
COLOR_GRAY2BGRA	9	COLOR_BayerGB2BGR	47	COLOR_RGB2YUV	83
COLOR_BGRA2GRAY	10	COLOR_BayerRG2BGR	48	COLOR_YUV2BGR	84
COLOR_RGBA2GRAY	11	COLOR_BayerGR2BGR	49	COLOR_YUV2RGB	85
COLOR_BGR2XYZ	32	COLOR_BGR2LUV	50	COLOR_BayerBG2GRAY	86
COLOR_RGB2XYZ	33	COLOR_RGB2LUV	51	COLOR_BayerGB2GRAY	87
COLOR_XYZ2BGR	34	COLOR_BGR2HLS	52	COLOR_BayerRG2GRAY	88
COLOR_XYZ2RGB	35	COLOR_RGB2HLS	53	COLOR_BayerGR2GRAY	89

예제 6.4.4 다양한 컬러 공간 변환 – convert_others.py

```
01  import cv2
02
03  BGR_img = cv2.imread("images/color_space.jpg", cv2.IMREAD_COLOR) # 컬러 영상 읽기
04  if BGR_img is None: raise Exception("영상파일 읽기 오류")
05
06  Gray_img = cv2.cvtColor(BGR_img, cv2.COLOR_BGR2GRAY)          # 명암도 영상 변환
07  YCC_img = cv2.cvtColor(BGR_img, cv2.COLOR_BGR2YCrCb)          # YCbCr 컬러 공간 변환
08  YUV_img = cv2.cvtColor(BGR_img, cv2.COLOR_BGR2YUV)            # YUV 컬러 공간 변환
09  LAB_img = cv2.cvtColor(BGR_img, cv2.COLOR_BGR2LAB)            # La*b* 컬러 공간 변환
10
11  YCC_ch = cv2.split(YCC_img)                                   # 채
12  YUV_ch = cv2.split(YUV_img)
13  Lab_ch = cv2.split(LAB_img)
14
15  cv2.imshow("BGR_img", BGR_img)
16  cv2.imshow("Gray_img", Gray_img)
17
18  sp1, sp2, sp3 = ['Y', 'Cr', 'Cb'] , ['Y', 'U', 'V'], ['L', 'A', 'B']
19  for i in range(len(ch1)):
20      cv2.imshow("YCC_img[%d]-%s" %(i, sp1[i]), YCC_ch[i])
21      cv2.imshow("YUV_img[%d]-%s" %(i, sp2[i]), YUV_ch[i])
22      cv2.imshow("LAB_img[%d]-%s" %(i, sp3[i]), Lab_ch[i])
23  cv2.waitKey(0)
```

심화예제 6.4.5 Hue 채널을 이용한 객체 검출 – 17.hue_threshold.py

```python
import numpy as np, cv2

def onThreshold(value):
    th[0] = cv2.getTrackbarPos("Hue_th1", "result")
    th[1] = cv2.getTrackbarPos("Hue_th2", "result")

    ## 이진화– 화소 직접 접근 방법
    # result = np.zeros(hue.shape, np.uint8)
    # for i in range(result.shape[0]):
    #     for j in range(result.shape[1]):
    #         if th[0] <= hue[i, j] < th[1] : result[i, j] = 25

    ## 이진화– 넘파이 함수 활용 방식
    # result = np.logical_and(hue < th[1], hue >= th[0])
    # result = result.astype('uint8') * 255

    ## OpenCV 이진화 함수 이용– 상위 값과 하위 값 제거
    _, result = cv2.threshold(hue, th[1], 255, cv2.THRESH_TOZERO_INV)
    cv2.threshold(result, th[0], 255, cv2.THRESH_BINARY, result)
    cv2.imshow("result", result)

BGR_img = cv2.imread("images/color_space.jpg", cv2.IMREAD_COLOR)      # 컬러 영상 읽기
if BGR_img is None: raise Exception("영상파일 읽기 오류")

HSV_img = cv2.cvtColor(BGR_img, cv2.COLOR_BGR2HSV)      # 컬러 공간 변환
hue = np.copy(HSV_img[:, :, 0])                         # hue 행렬에 색상 채널 복사

th = [50, 100]                                          # 트랙바로 선택할 범위 변수
cv2.namedWindow("result")
cv2.createTrackbar("Hue_th1", "result", th[0], 255, onThreshold)
cv2.createTrackbar("Hue_th2", "result", th[1], 255, onThreshold)
onThreshold(th[0])                                      # 이진화 수행
cv2.imshow("BGR_img", BGR_img)
cv2.waitKey(0)
```