화소 처리
Pixel-based Image Processing

목차

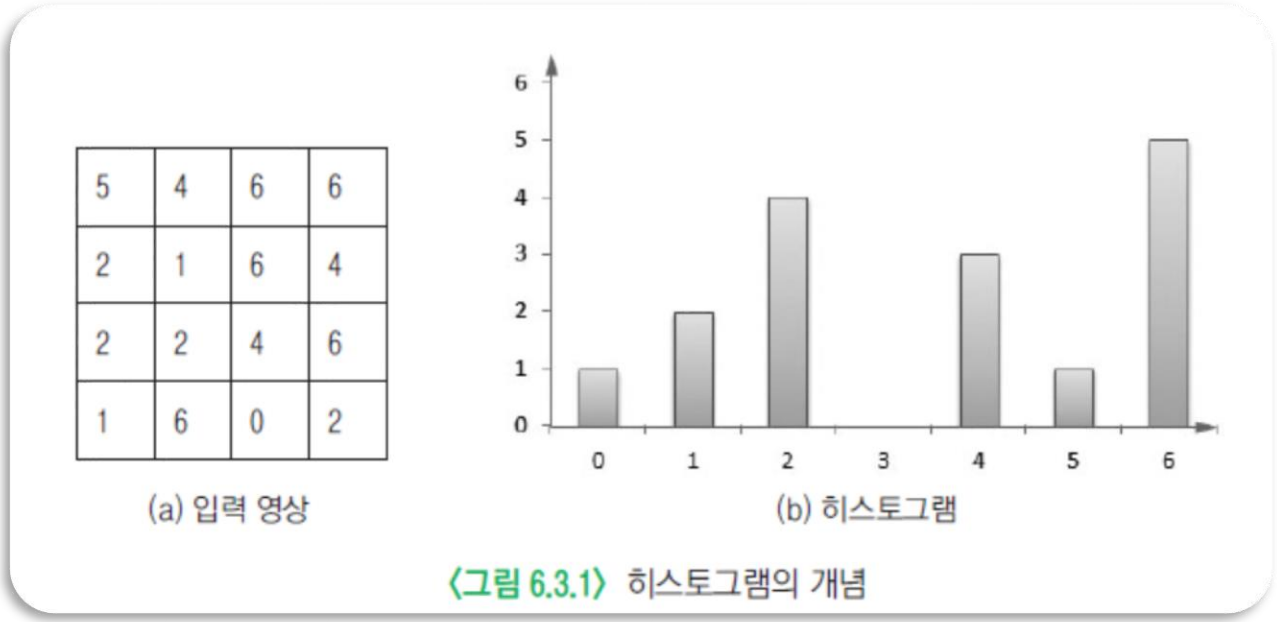
3. 히스토그램(histogram)

- ❖히스토그램 개념
- ❖히스토그램 계산
- ❖OpenCV 함수 활용
- ❖히스토그램 스트레칭
- ❖히스토그램 평활화
- ❖히스토그램의 응용

히스토그램의 개념

❖히스토그램

- 영상내 화소 값에 따라 그 빈도를 그래프로 표현한 것
- 영상의 화소 분포 상태를 쉽게 이해
- 영상의 특성을 판단하는 도구로 사용



〈그림 6.3.1〉 히스토그램의 개념

예제 6.3.1 영상 히스토그램 계산 – 08.calc_histogrm.opencv.py(일부)

```
01  import numpy as np, cv2
02
03  def calc_histo(image, histSize, ranges=[0, 256] ):      # 행렬 원소의 1차원 히스토그램 계산
04      hist = np.zeros((histSize, 1), np.float32)          # 히스토그램 누적 행렬
05      gap = ranges[1] / histSize                          # 계급 간격
06
07      for row in ima
08          for pix i
09              idx =
10              hist[
11      return hist
```

histSize = Bin의 개수

함수 및 인수 구조

cv2.calcHist (images , channels , mask , histSize , ranges [, hist [, accumulate]]) → ret

■설명: 행렬의 원속값의 빈도를 계산한다.

인수 설명	■images	원본 배열들 – CV_8U 혹은 CV_32F 형으로 크기가 같아야 함
	■channels	히스토그램 계산에 사용되는 차원 목록
	■mask	특정 영역만 계산하기 위한 마스크 행렬 – 입력 영상과 같은 크기의 8비트 배열
	■histSize	각 차원의 히스토그램 배열 크기 – 계급(bin)의 개수
	■ranges	각 차원의 히스토그램의 범위
	■accumulate	누적 플래그– 여러 배열에서 단일 히스토그램을 구할 때 사용

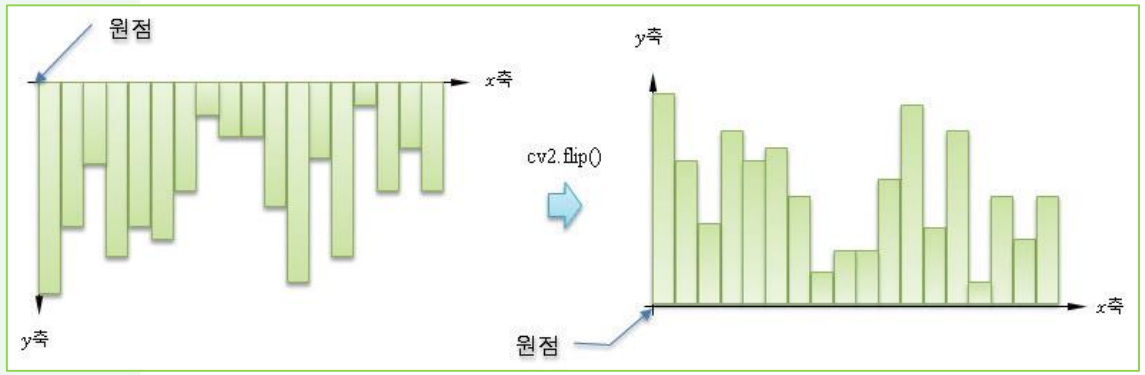
예제 6.3.1 영상 히스토그램 계산 – 08.calc_histogram_opencv.py

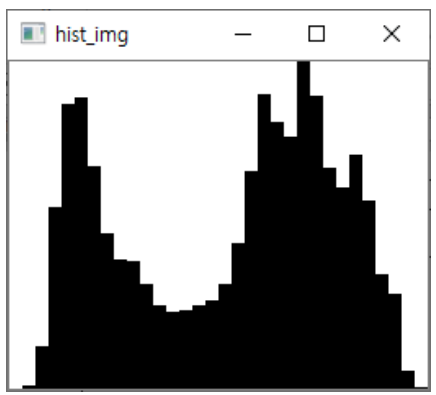
```
01  import numpy as np, import cv2
02
03  def calc_histo(image, histSize, ranges=[0, 256]): …          # 소스 내용 생략
12
13  image = cv2.imread("images/pixel_test.jpg", cv2.IMREAD_GRAYSCALE)          # 영상 읽기
14  if image is None: raise Exception("영상파일 읽기 오류")
15
16  histSize, ranges = [32], [0, 256]                    # 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 함
21  hist3, bins = np.histogram(image, ranges_gap )                    # numpy 모듈
22
23  print("User 함수: \n", hist1.flatten())                           # 1차원 행렬
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]
```

예제 6.3.3 히스토그램 그래프 그리기 – 09.draw_histogram.py

```
01  import numpy as np, cv2
02
03  def draw_histo(hist, shape=(200, 256)):
04      hist_img = np.full(shape, 255, np.uint8)
05      cv2.normalize(hist, hist, 0, shape[0], cv2.NORM_MINMAX)     # 정규화
06      gap = hist_img.shape[1]/hist.shape[0]                       # 한 계급 너비
07
08      for i, h in enumerate(hist):
09          x = int(round(i * gap))                         # 막대 사각형 시작 x 좌표
10          w = int(round(gap))
11          cv2.rectangle(hist_img, (x, 0, w, int(h)), 0, cv2.FILLED)
12
13      return cv2.flip(hist_img, 0)                        # 영상 상하 뒤집기 후 반환
14
15  image = cv2.imread("images/pixel.jpg", cv2.IMREAD_GRAYSCALE)  # 영상 읽기
16  if image is None: raise Exception("영상파일 읽기 오류")
17
18  hist = cv2.calcHist([image], [0], None, [32], [0, 256])
19  hist_img = draw_histo(hist)
20
21  cv2.imshow("image", image)
22  cv2.imshow("hist_img", hist_img)
23  cv2.waitKey(0)
```

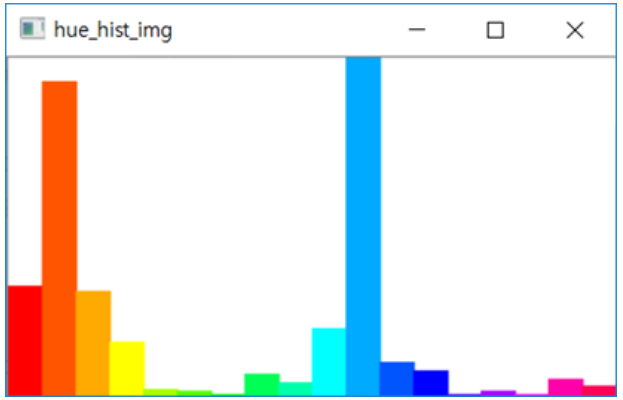




심화예제 6.3.4 색상 히스토그램 그리기 – hue_histogram.py

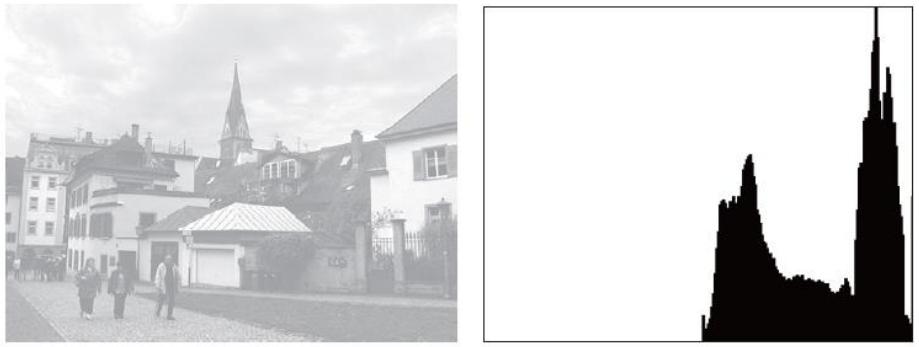
```python
import numpy as np, cv2

def make_palette(rows):                                  # hue 채널 팔레트 행렬 생성 함수
    ## 리스트 생성 방식
    hue = [round(i * 180 / rows) for i in range(rows)] # hue 값 리스트 계산
    hsv = [[[h, 255, 255]] for h in hue]                 # (hue, 255,255) 화소값 계산
    hsv = np.array(hsv, np.uint8)                        # 정수(uint8)형 행렬 변환
    ## 반복문 방식
    # hsv = np.full((rows, 1, 3), (255, 255, 255), np.uint8)
    # for i in range(0, rows):                           # 행수만큼 반복
    #     hue = round(i / rows * 180 )                   # 색상 계산
    #     hsv[i] = (hue, 255, 255)                       # HSV 컬러 지정

    return cv2.cvtColor(hsv, cv2.COLOR_HSV2BGR)          # HSV 컬러→BGR 컬러

def draw_hist_hue(hist, shape=(200, 256, 3)):            # 색상 히스토그램 그리기 함수
    hsv_palette = make_palette(hist.shape[0])            # 색상 팔레트 생성
    hist_img = np.full(shape, 255, np.uint8)
    cv2.normalize(hist, hist, 0, shape[0], cv2.NORM_MINMAX) # 영상 높이값으로 정규화

    gap = hist_img.shape[1] / hist.shape[0]                  # 한 계급 크기
    for i, h in enumerate(hist):
        x, w = int(round(i * gap)), int(round(gap))
        color = tuple(map(int, hsv_palette[i][0]))           # 정수형 튜플로 변환
        cv2.rectangle(hist_img, (x, 0, w, int(h)), color, cv2.FILLED)  # 팔레트 색으로 그리기

    return cv2.flip(hist_img, 0)

```

```python
image = cv2.imread("images/hue_hist.jpg", cv2.IMREAD_COLOR)  # 영상 읽기
if image is None: raise Exception("영상파일 읽기 오류")

hsv_img = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)              # BGR 컬러→HSV 컬러
hue_hist = cv2.calcHist( [hsv_img], [0], None, [18], [0,180]) # Hue 채널 히스토그램 계산
hue_hist_img = draw_hist_hue(hue_hist, (200, 360, 3))         # 히스토그램 그래프

cv2.imshow("image", image)
cv2.imshow("hue_hist_img", hue_hist_img)
cv2.waitKey(0)
```



히스토그램 스트레칭

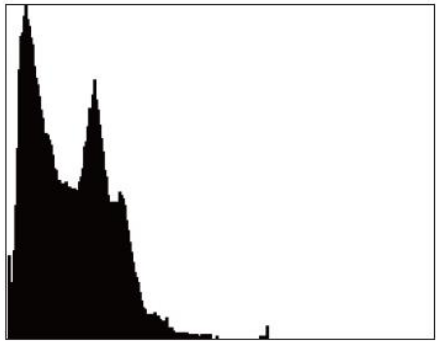
❖히스토그램의 분포가 좁아서 영상의 대비가 좋지 않은 영상의 화질을 개선할 수 있는 알고리즘



밝은 부분을 많이 분포하는 영상

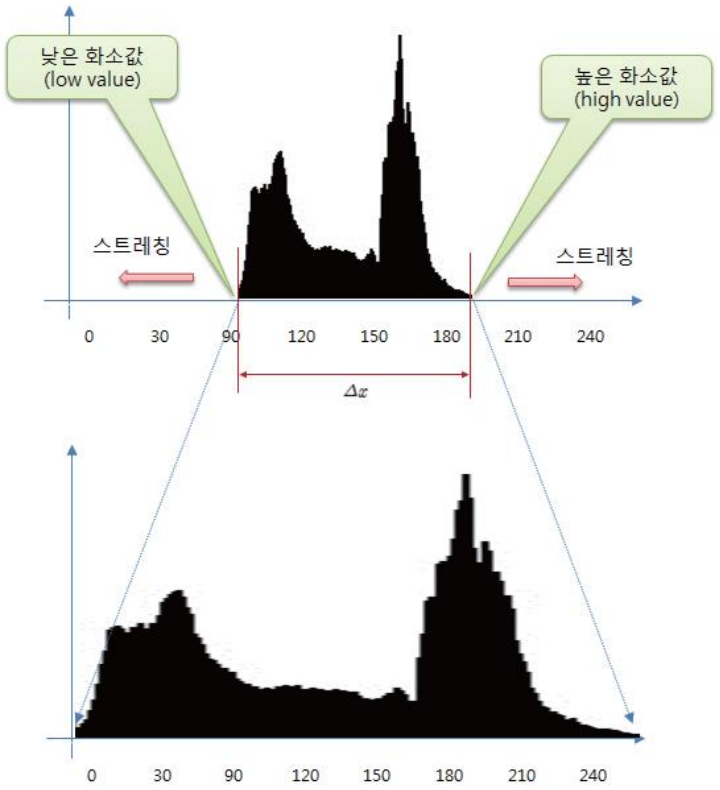
히스토그램

어두운 부분을 많이 분포하는 영상



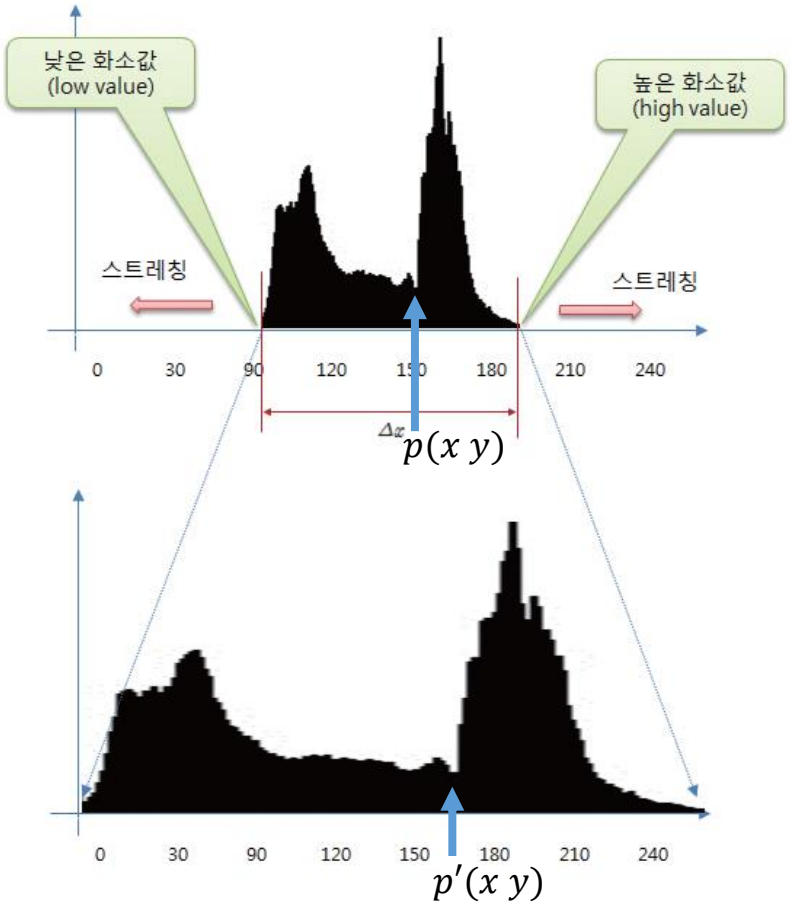
히스토그램

〈그림 6.3.3〉 영상에 따른 히스토그램 예



히스토그램 스트레칭

❖히스토그램의 분포가 좁아서 영상의 대비가 좋지 않은 영상의 화질을 개선할 수 있는 알고리즘

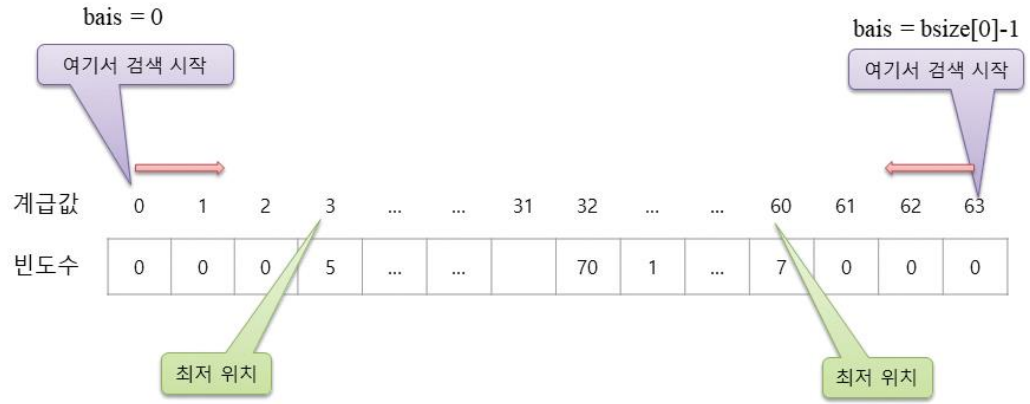


$$(high - low) : 255 = (p(x,y) - low) : p'(x,y)$$

$$p'(x,y) = (p(x,y) - low) \times \frac{255}{(high - low)}$$

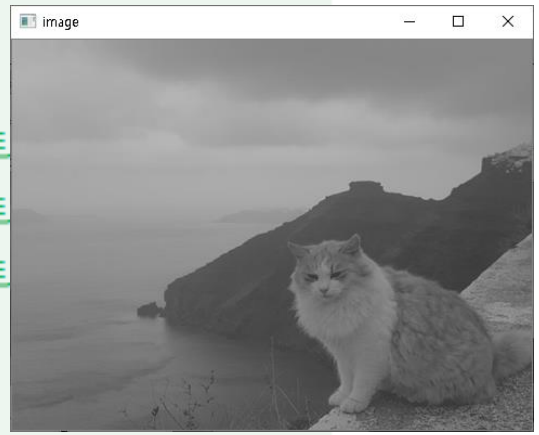
예제 6.3.5 히스토그램 스트래칭 – histogram_stretching.py

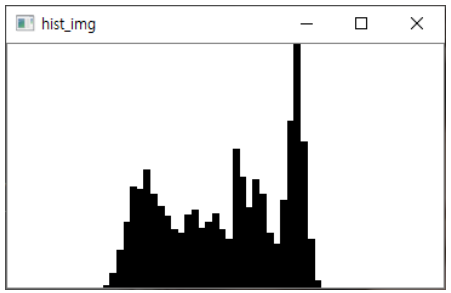
```
01  import numpy as np, cv2
02  from Common.histogram import draw_histo              # 함수 재사용 위한 임포트
03
04  def search_value_idx(hist, bias=0):                  # 값있는 첫 계급 검색 함수
05      for i in range(hist.shape[0]):
06          idx = np.abs(bias - i)                       # 검색 위치(처음 또는 마지막)
07          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
```

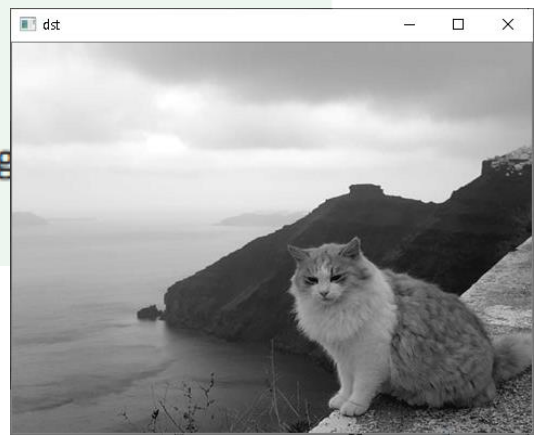


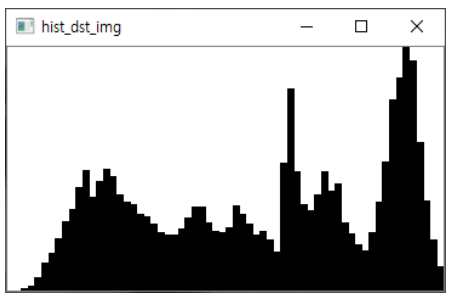
```
25  dst = cv2.LUT(image, idx.astype('uint8'))                    # 룩업 테이블 사용
26  ## 룩업 테이블 사용하지 않고 직접 구현
27  # dst = np.zeros(image.shape, dtype=image.dtype)
28  # for i in range(dst.shape[0]):
29  #         for j in range(dst.shape[1]):
30  #                 dst[i,j] = idx[image[i,j]]
31
32  hist_dst = cv2.calcHist([dst], [0], None, bsize, ranges)# 결과 영상 히스토
33  hist_img = draw_histo(hist, (200,360))                       # 원본 영상 히스토
34  hist_dst_img = draw_histo(hist_dst, (200,360))               # 결과 영상 히스토
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)
```

```
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

- 한쪽으로 치우친 명암 분포를 가진 영상을 히스토그램의 재분배 과정을 거쳐서 균등한 히스토그램 분포를 갖게 하는 알고리즘

- 히스토그램 평활화 알고리즘

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

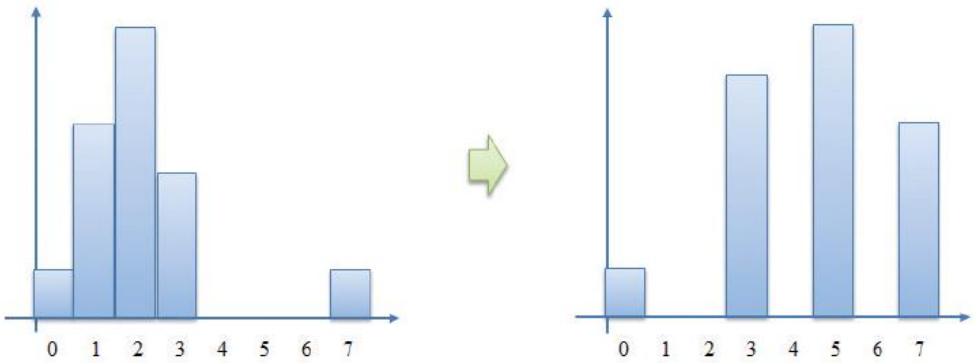
0	5	5	3
3	5	7	5
3	5	7	5
3	7	3	7

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
평활화 결과	0	3	5	7	7	7	7	7

〈그림 6.3.6〉 평활화 계산 과정 예시

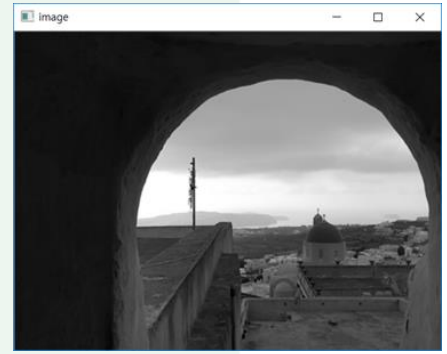
① 영상의 히스토그램을 계산한다.
② 히스토그램 빈도값에서 누적 빈도수(누적합)를 계산한다.
③ 누적 빈도수를 정규화(정규화 누적합)한다.
④ 결과 화소값 = 정규화 누적합 * 최대 화소값

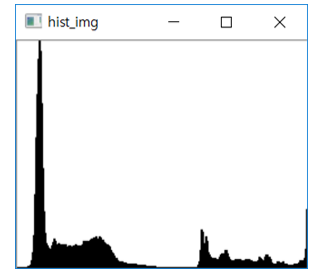


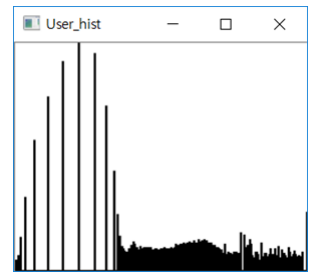
예제 6.3.6 히스토그램 평활화 – histogram_equalize.py

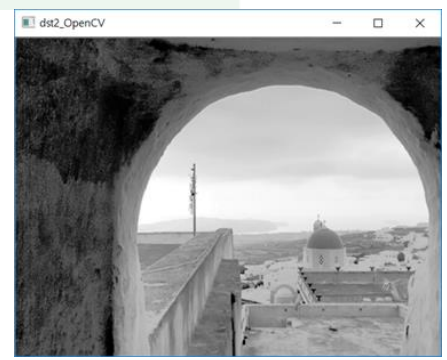
```
01  import numpy as np, cv2
02  from Common.histogram import draw_histo              # 히스토그램 그리기 함수 임포트
03
04  image = cv2.imread("images/equalize_test.jpg", cv2.IMREAD_GRAYSCALE) # 영상 읽기
05  if image is None: raise Exception("영상파일 읽기 오류")
06
07  bins, ranges = [256], [0, 256]
08  hist = cv2.calcHist([image], [0], None, bins, ranges)              # 히스토그램 계산
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                        # 누적합의 정규화
17  dst1 = [[accum_hist[val] for val in row] for row in image]         # 화소값 할당
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
```

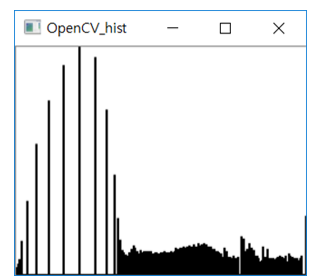
```
25  dst2 = cv2.equalizeHist(image)                          # OpenCV 히스토그램 평활화
26  hist1 = cv2.calcHist([dst1], [0], None, bins, ranges)   # 히스토그램 계산
27  hist2 = cv2.calcHist([dst2], [0], None, bins, ranges)
28  hist_img = draw_histo(hist)
29  hist_img1 = draw_histo(hist1)
30  hist_img2 = draw_histo(hist2)
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. 컬러 공간 변환

- ❖컬러 및 컬러 공간
- ❖RGB 컬러 공간
- ❖CMY(K) 컬러 공간
- ❖HSI 컬러 공간
- ❖기타 컬러 공간

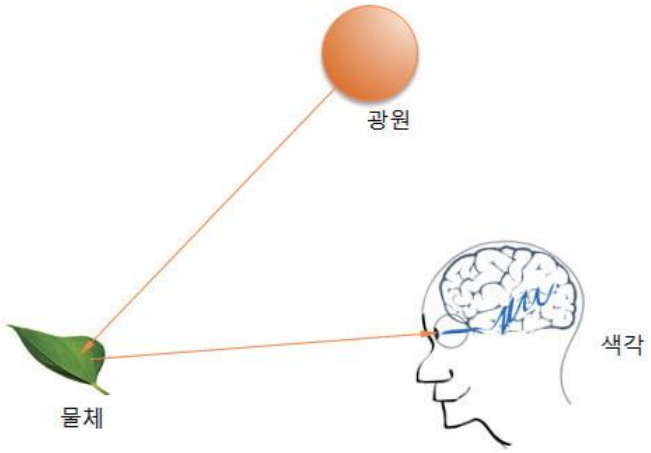
컬러 및 컬러 공간

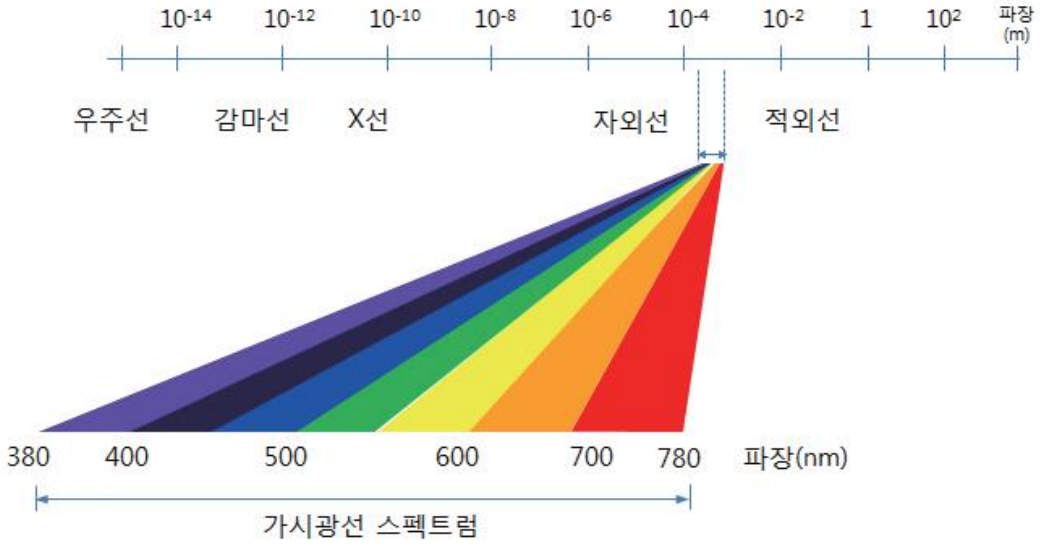
- ❖개와 고양이가 보는 세상?
  - ▪동물들은 다양한 색상을 인지하지 못함
- ❖영장류에 대한 가설
  - ▪어린 잎을 골라 먹기 위해 원래 흑백이었던 시각
    - 1차로 파랑색과 노랑색을 구분하는 2색형 색각을 갖게 됨
  - ▪그 뒤에 노랑색을 감지하는 시세포
    - 각각 빨간색과 녹색에 민감한 시세포로 분화해 3색형 색각을 갖게 됨

컬러 및 컬러 공간

❖색(color)
- 빛에서 주파수의 차이에 따라 다르게 느껴지는 색상들

❖물체에 닿는 빛은 일부 흡수, 일부 반사
- 흡수되지 않고 반사된 빛을 사람의 눈이 인지하는 것이 그 물체의 색
- 가시광선: 인간이 인지하는 파장의 빛 (380~780 nm)

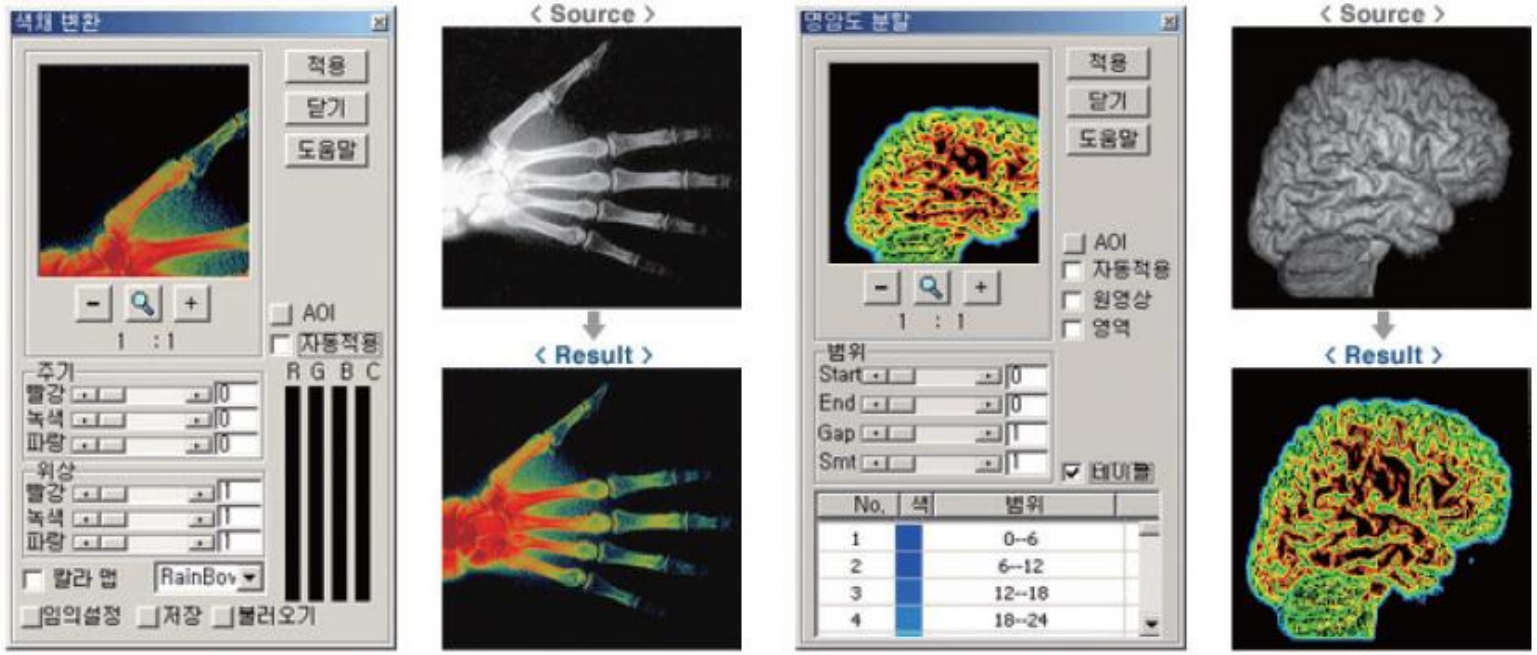




컬러 및 컬러 공간

❖컬러 공간(color space)

- 색 표시계의 모든 색들을 색 공간에서 3차원 좌표로 표현한 것
- 공간상의 좌표로 표현
  - 어떤 컬러와 다른 컬러들 간의 관계를 표현하는 논리적인 방법 제공



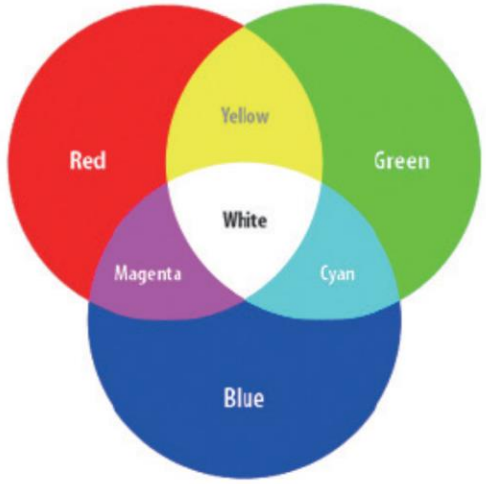
RGB 컬러 공간

❖빛을 이용해서 color를 표현

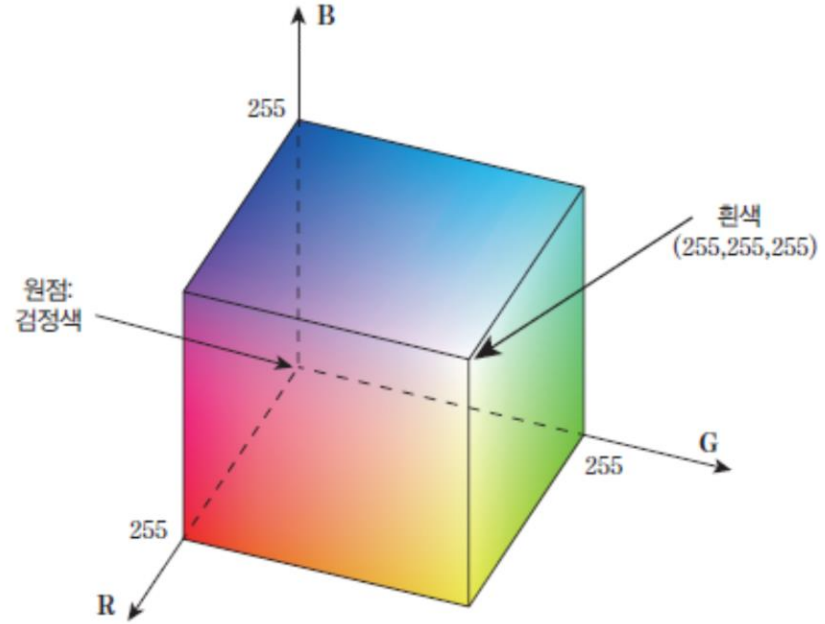
- 빨강 빛, 초록 빛, 파랑 빛 필요
  → 빛의 삼원색 (가산혼합)
- 모니터, 텔레비전, 빔 프로젝터와 같은 디스플레이 장비들에서 사용

❖RGB 컬러 공간

- R, G, B를 축으로 구성하여 입방체를 만들어 3차원 좌표계 형성
- OpenCV에서 컬러의 채널 순서
  - Blue, Green, Red



〈그림 6.4.5〉 가산 혼합



〈그림 6.4.6〉 RGB 컬러 공간

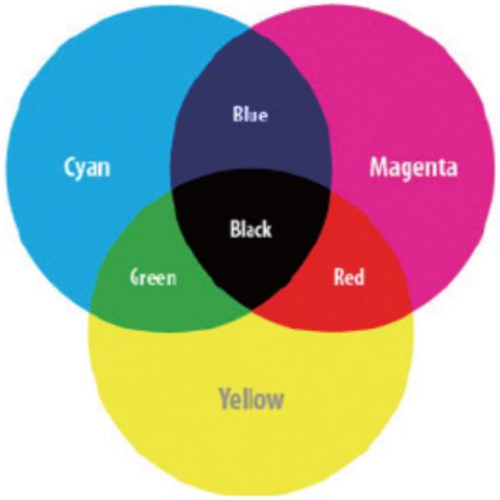
RGB 컬러 공간

〈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

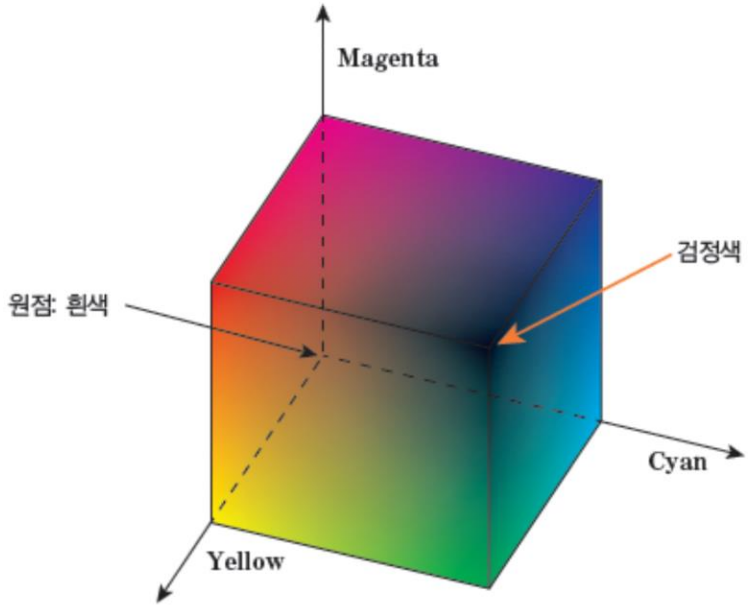
CMY(K) 컬러 공간

❖색을 이용해서 color를 표현
- 청록색, 자홍색, 노랑색 필요
- 색의 3원색 (감산 혼합)
  - 색은 섞으면 섞을수록 어두워짐



〈그림 6.4.7〉 감산 혼합

❖CMY 컬러 공간
- 색의 삼원색을 3개의 축으로 구성하여 입방체를만들어 3차원 좌표계를 형성



〈그림 6.4.8〉 CMY 컬러 공간

CMY(K) 컬러 공간

❖CMY(K) 컬러 공간 수식

- RGB 컬러와 보색관계

$$C = 255 - R \qquad R = 255 - C$$
$$M = 255 - G \qquad G = 255 - M$$
$$Y = 255 - R \qquad B = 255 - Y$$

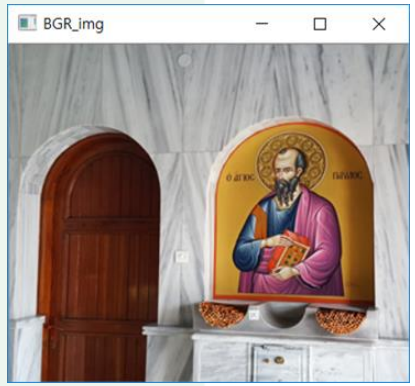
❖CMYK 컬러 공간

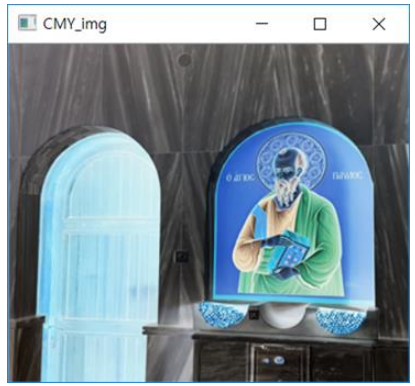
- 아무리 많은 색들을 섞어도 순수한 검은색이 되지 않음
- 순수한 검은색을 출력하기 위해 검은색 컬러 채널 분리

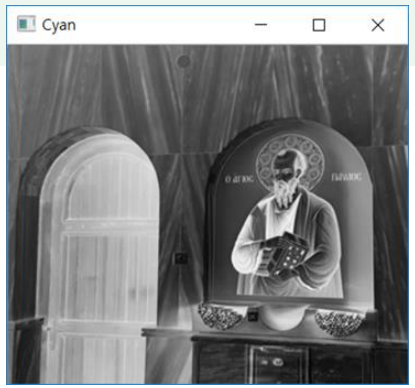
$$blacK = \min(Cyan, Magenta, Yellow)$$
$$Cyan = Cyan - blacK$$
$$Magenta = Magenta - blacK$$
$$Yellow = Yellow - blacK$$

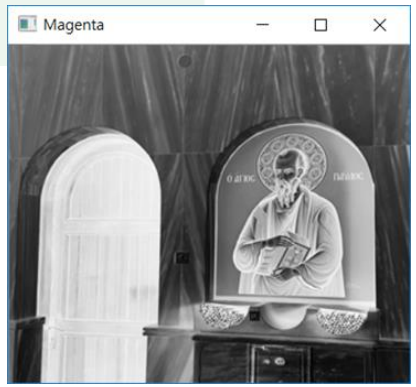
예제 6.4.1 컬러 공간 변환(BGR→CMY) - convert_CMY.py

```
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)
```



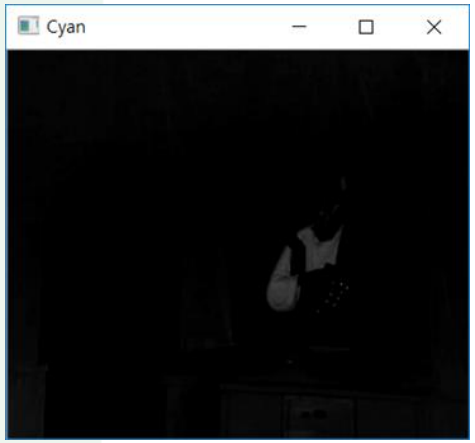


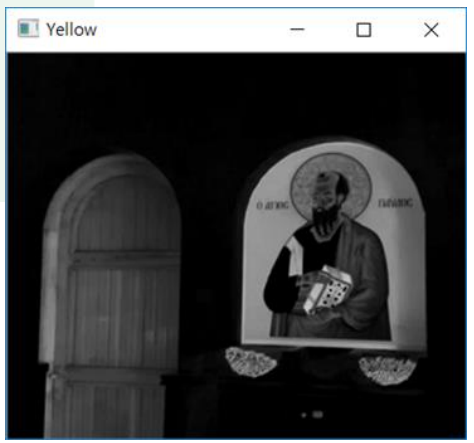


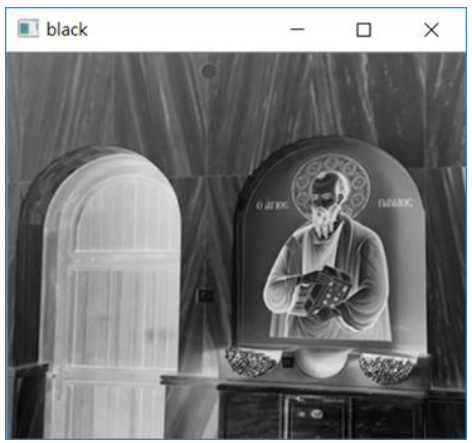


예제 6.4.2 컬러 공간 변환(BGR→CMYK) - 14.conver_CMYK.py

```
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)
```



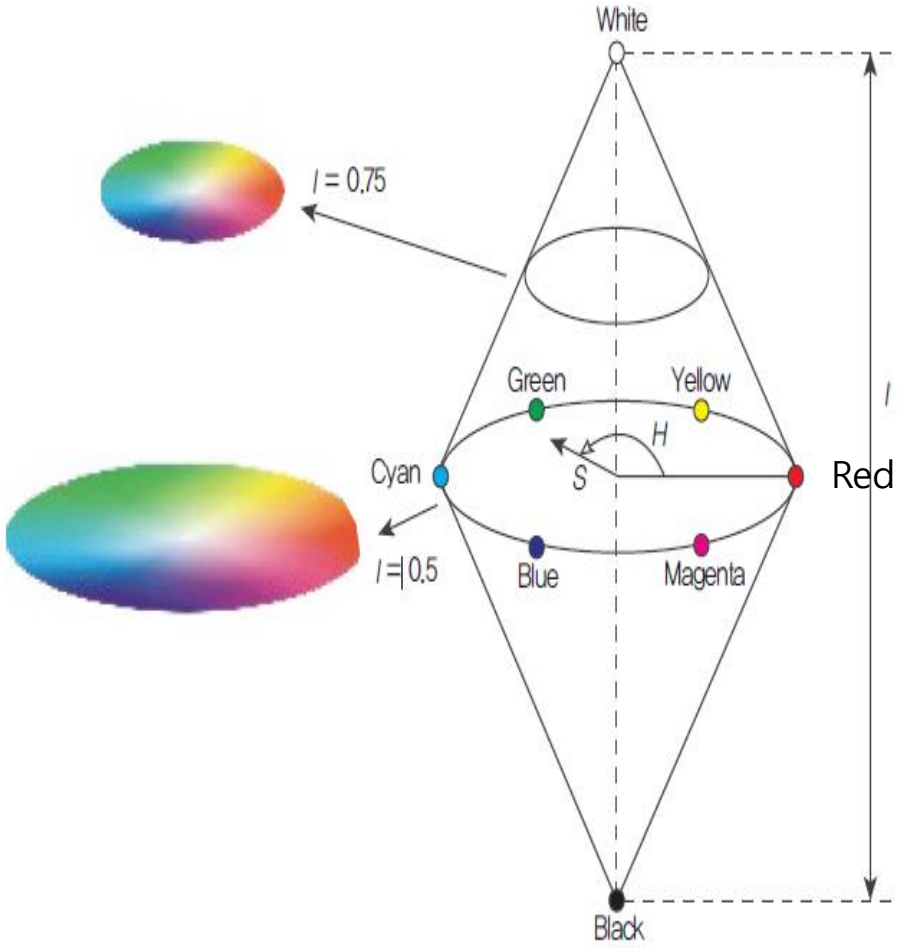




HSI 컬러 공간

❖인간의 색상인지 3요인
- 색상(Hue), 채도(Saturation), 명도(Intensity)

❖HSI 컬러 공간
- 색상, 채도, 명도를 컬러 공간에 표시
- 인간 시각 시스템 특성과 가장 유사
  - 색상: 원판의 0~360도까지 회전
    (R 0도, G 120도, B 240도)
  - 채도: 색의 순수한 정도(0~100)
  - 명도: 빛의 세기(black~white)



〈그림 6.4.9〉 HSI 컬러 공간

HSI 컬러 공간

RGB → HSI 변환 수식

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

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

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

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

OpenCV HSV 변환 수식

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

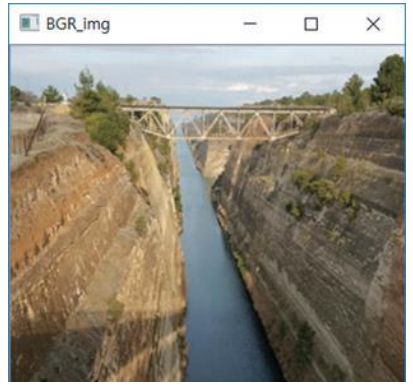
$$S = \begin{cases} V - \frac{\min(R,G,B)}{V}, & \text{if } V \neq 0 \\ 0 & , \ otherwise \end{cases}$$

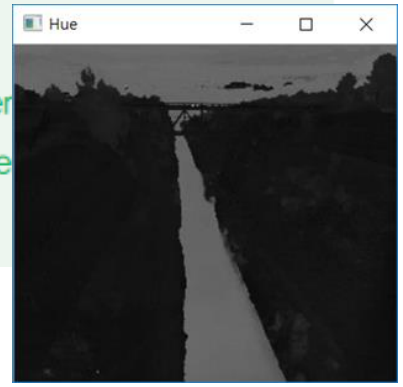
$$V = \max(R,G,B)$$

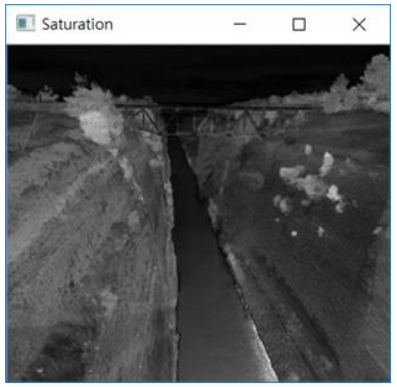
예제 6.4.3 컬러 공간 변환(BGR→HSV) – 15.conver_HSV.py

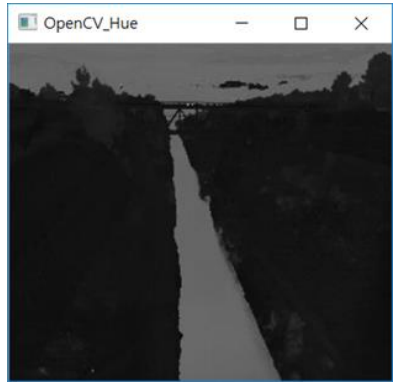
```
01  import numpy as np, cv2, math
02
03  def calc_hsi(bgr):                                          # 한 화소 hsi 계산 함수
04      # B, G, R = bgr.astype(float)                           # float 형 변환
05      B, G, R = float(bgr[0]), float(bgr[1]), float(bgr[2])   # 속도면에 유리
06      bgr_sum = (R + G + B)
07      ##    색상 계산
08      tmp1 = ((R - G) + (R - B)) * 0.5
09      tmp2 = math.sqrt((R - G) * (R - G) + (R - B) * (G - B))
10      angle = math.acos(tmp1 / tmp2) * (180 / np.pi) if tmp2 else 0      # 각도
11
12      H = angle if B <= G else 360 - angle                    # 색상
13      S = 1.0 - 3 * min([R, G, B]) / bgr_sum if bgr_sum else 0  # 채도
14      I = bgr_sum / 3                                         # 명도
15      return (H/2, S*255, I)                                  # 3 원소 튜플로 반환
16
17  ## BGR 컬러→HSI 컬러 변환 함수
18  def bgr2hsi(image):
19      hsv = [[calc_hsi(pixel) for pixel in row] for row in image]     # 2차원 배열 순회
20      return cv2.convertScaleAbs(np.array(hsv))
21
```

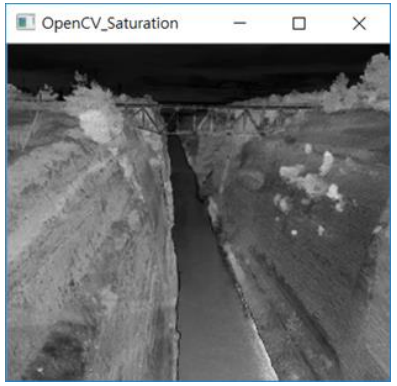
```python
BGR_img = cv2.imread("images/color_space.jpg", cv2.IMREAD_COLOR)        # 컬러 영상 읽기
if BGR_img is None: raise Exception("영상파일 읽기 오류")

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
[cv2.imshow('OpenCV_'+t, eval(t+'2')) for t in titles[1:]]              # Ope
cv2.waitKey(0)
```











기타 컬러 공간

❖YCbCr컬러 공간

- 영상 시스템에서 사용되는 색 공간의 일종
- Y(휘도 성분), Cb Cr(색차 성분)
- 인간의 시각은 밝기에는 민감하지만, 색상에는 덜 민감
  - 색차 신호(Cr, Cb 성분)를 Y성분보다 낮은 해상도로 구성

    → 인간의 시각에서 화질의 큰 저하 없이 영상 데이터 용량 감소

❖RGB →YCbCr

$$Y = +0.299 \cdot R + 0.587 \cdot G + 0.114 \cdot B$$
$$Cb = (R - Y) \cdot 0.564 + 128$$
$$Cr = (B - Y) \cdot 0.713 + 128$$

$$R = Y + 1.403 \cdot (Cr - 128)$$
$$G = Y - 0.714 \cdot (Cb - 128) - 0.344 (Cb - 128)$$
$$B = Y + 1.773 \cdot (Cb - 128)$$

기타 컬러 공간

❖YUV 컬러 공간

- TV 방송 규격에서 사용하는 컬러 표현 방식
- PAL 방식의 아날로그 비디오를 위해 개발
- 디지털 비디오에서도 유럽의 비디오 표준으로 사용

❖RGB→ YUV

$$Y = +0.2160 \cdot R + 0.7152 \cdot G + 0.0722 \cdot B$$
$$U = -0.0999 \cdot R - 0.3360 \cdot G + 0.4360 \cdot B$$
$$V = +0.6150 \cdot R - 0.5586 \cdot G - 0.05639 \cdot B$$

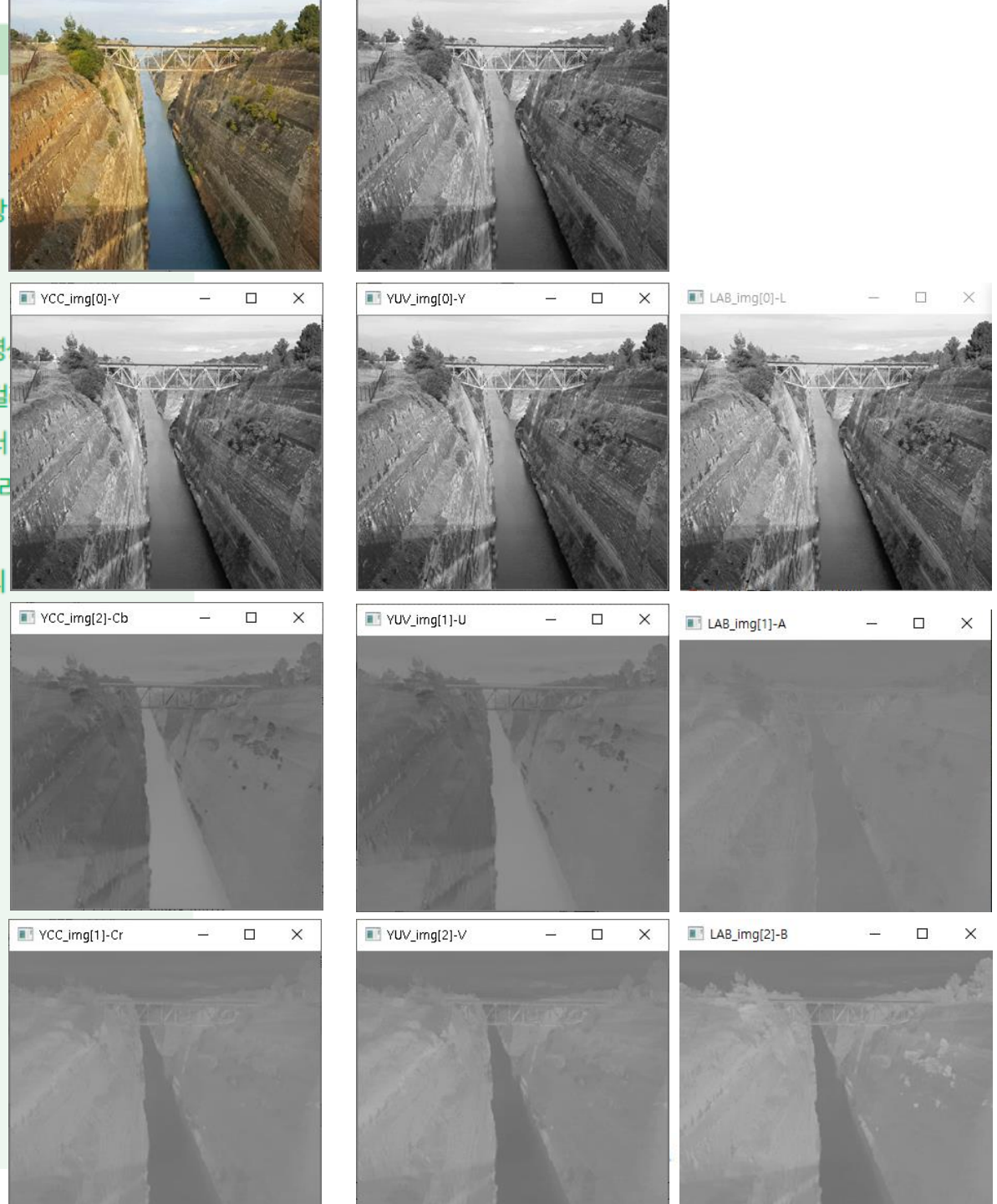
$$R = Y + 1.28033 \cdot V$$
$$G = Y - 0.21482 \cdot U - 0.38059 \cdot V$$
$$B = Y + 2.12798 \cdot U$$

〈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

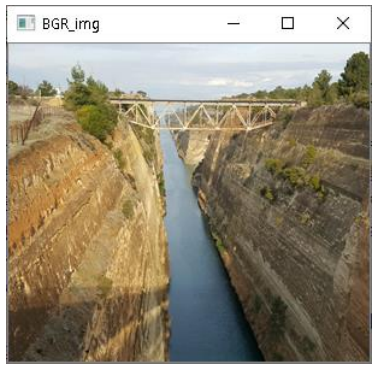
예제 6.4.4 다양한 컬러 공간 변환 – convert_others.py

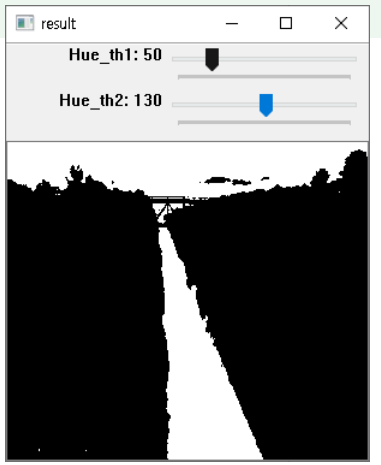
```
01  import cv2
02
03  BGR_img = cv2.imread("images/color_space.jpg", cv2.IMREAD_COLOR) # 컬러 영상
04  if BGR_img is None: raise Exception("영상파일 읽기 오류")
05
06  Gray_img = cv2.cvtColor(BGR_img, cv2.COLOR_BGR2GRAY)          # 명암도 영
07  YCC_img = cv2.cvtColor(BGR_img, cv2.COLOR_BGR2YCrCb)          # YCbCr 컬
08  YUV_img = cv2.cvtColor(BGR_img, cv2.COLOR_BGR2YUV)            # YUV 컬러
09  LAB_img = cv2.cvtColor(BGR_img, cv2.COLOR_BGR2LAB)            # La*b* 컬러
10
11  YCC_ch = cv2.split(YCC_img)                                   # 채널 분리
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)
```



실화예제 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] = 255

    ## 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)
```





Summary

❖영상내 화소 접근

- Img[r, c] img.item(r, c) img.itemset(r, c)

❖명도 영상 (gray-scale image)

- 하나의 화소값은 0~255의 값을 가지는데 0은 검은색을, 255는 흰색을 의미

❖영상의 밝기 값 변환

- 영상에 스칼라값을 가감하면 영상이 밝기가 밝게/어둡게 됨
- 행렬에 스칼라값을 곱하면 영상의 대비를 조절할 수 있다.

❖히스토그램

- 영상내의 화수의 빈도 분포를 그래프로 나타낸
- 히스토그램 계산하는 opencv 함수 → cv2.calcHist()

Summary

❖히스토그램 스트레칭 (histogram stretching)
- 히스토그램의 분포가 한쪽으로 치우쳐서 분포가 좁아서 영상의 대비가 좋지 않은 영상의 화질을 개선할 수 있는 알고리즘

❖히스토그램 평활화 (histogram equalization)
- 불균등한 명암 분포를 가진 영상을 히스토그램의 재분배 과정을 거쳐서 균등한 히스토그램 분포를 갖게 하는 알고리즘

❖컬러 공간(color space)
- 모든 색들을 색 공간에서 3차원 좌표로 표현한 것
- 모니터(RGB), 프린터(CMYK), 인간(HSI)
- JPEG 압축(YCrCb), 방송시스템(YUV)
- 컬러 공간을 변환: cv2.**cvtColor() 함수** 이용