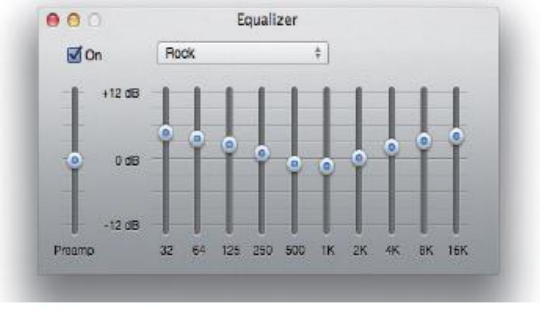
CHAPTER 06
화소처리 (2)

contents

- 6.3 히스토그램 평활화
- 6.4 컬러 공간 변환

6.3.5 히스토그램 평활화

- 이퀄라이저
  - 주파수 특성을 균등하게 보정하는 기기
  - 주파수 특성을 어느 특정의 목적에 맞추어 임의로 변화시켜 원하는 음색



〈그림 6.3.4〉 MP3 플레이어의 이퀄라이저의 예

- 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

6.3.5 히스토그램 평활화

- 평활화 과정 예시
  - 균일하지 않은 분포의 히스토그램을 균일하게 만드는 것
  - 히스토그램의 비선형적인 누적 분포 함수를 선형적인 형태로 변형

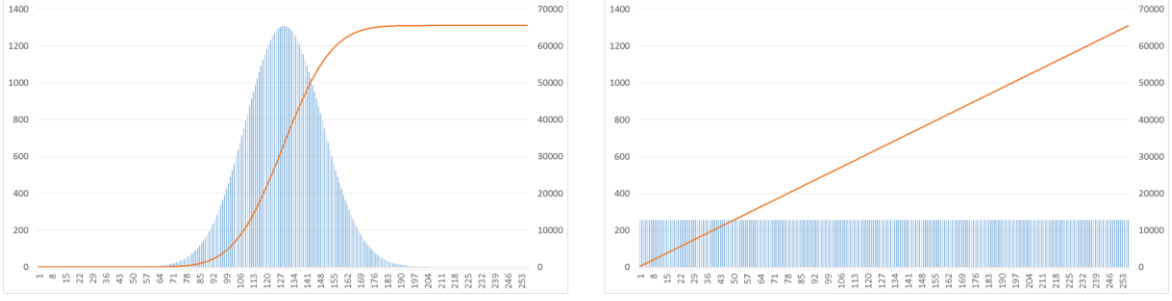


그림 3-9. 히스토그램 평활화의 개념

6.3.5 히스토그램 평활화

- 히스토그램 평활화 과정
  - i번째 화소 값 x의 히스토그램 H는 x_i의 발생 빈도 n_i로 정의

$$H(x_i) = n_i$$

    - 비트심도가 8인 경우 i의 범위는 0~255
  - x_i의 확률 밀도 함수 p (정규화)

$$p(x_i) = \frac{n_i}{n}, i = 0, 1, 2, \dots, 255$$

  - x_i에 대한 히스토그램 평활화 결과 값 s_i

$$s_k = 255 \times \sum_{i=0}^{k} p(x_i) = 255 \times \sum_{i=0}^{k} \frac{n_i}{n}, k = 0, 1, 2, \dots, 255$$

6.3.5 히스토그램 평활화

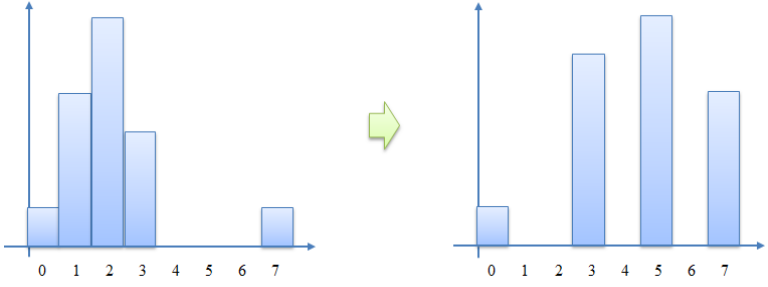
- 평활화 과정 예시

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

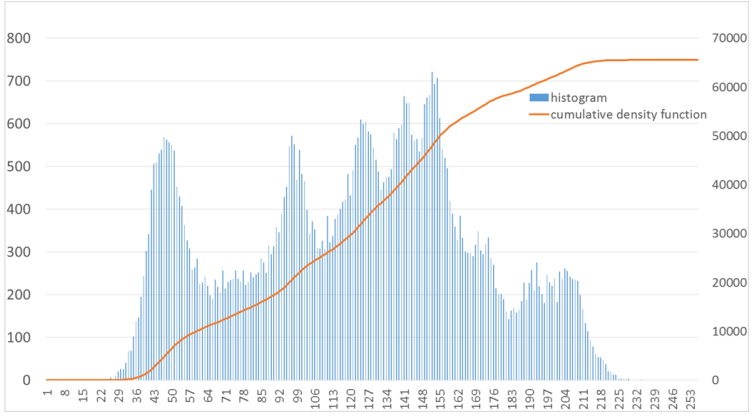
평활화 완료 영상 화소값

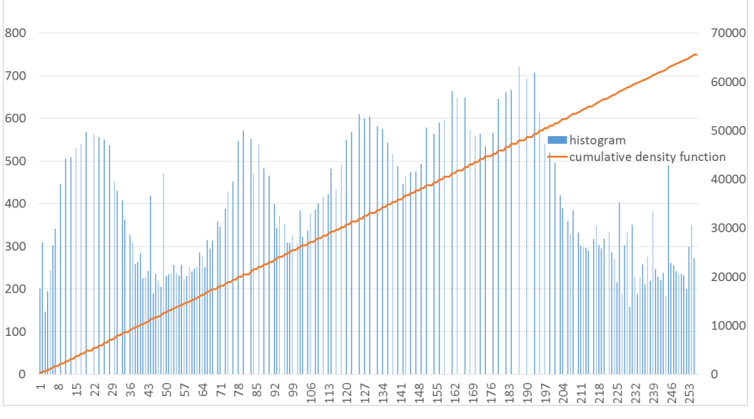


화소값	0	1	2	3	4	5	6	7
빈도수	1	5	6	3	0	0	0	1
누적 빈도수	1	6	12	15	15	15	15	16
정규화누적합	1/16	6/16	12/16	15//16	15//16	15/16	15/16	16/16
	0.0625	0.375	0.75	0.9375	0.9375	0.9375	0.9375	1
누적합 *최댓값	0.4375	2.625	5.25	6.5625	6.5625	6.5625	6.5625	7
평활화 결과	0	3	5	7	7	7	7	7

<그림 6.3.5> 평활화 계산 과정 예시

6.3.5 히스토그램 평활화





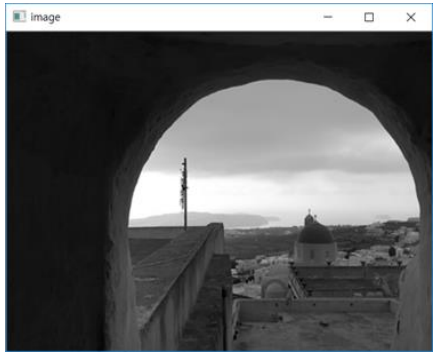
6.3.5 히스토그램 평활화

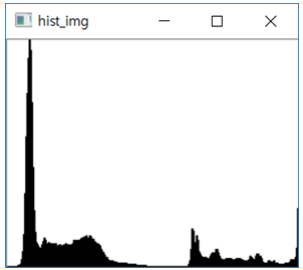
예제 6.3.6 히스토그램 평활화 – histogram_equalize.py

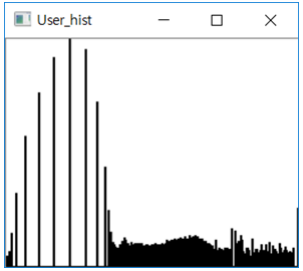
```
01  import numpy as np, cv2
02  from Common.histogram import draw_histo              # 히스토그램 그리기 함수 임포트
03
04  image = cv2.imread("images/equalize_test.jpg", cv2.IMREAD_GRAYSCALE) # 영상 읽기
05  if image is None: raise Exception("영상파일 읽기 오류")
06
07  bins, ranges = [256], [0, 256]
08  hist = cv2.calcHist([image],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

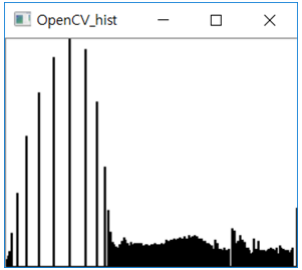
6.3.5 히스토그램 평활화

```
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)
```







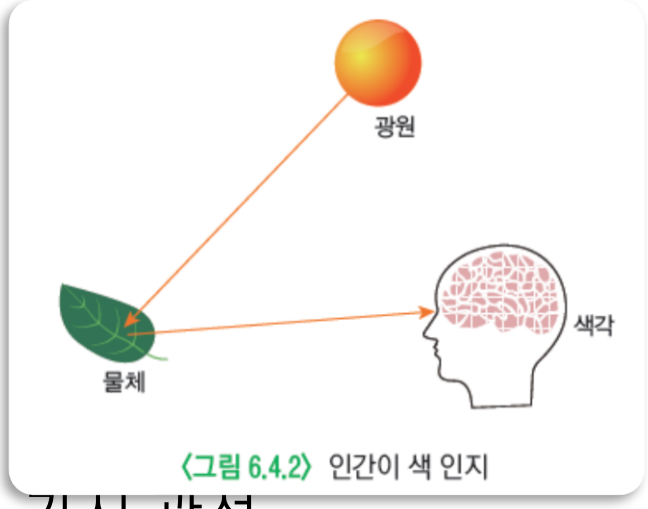


6.4 컬러 공간 변환

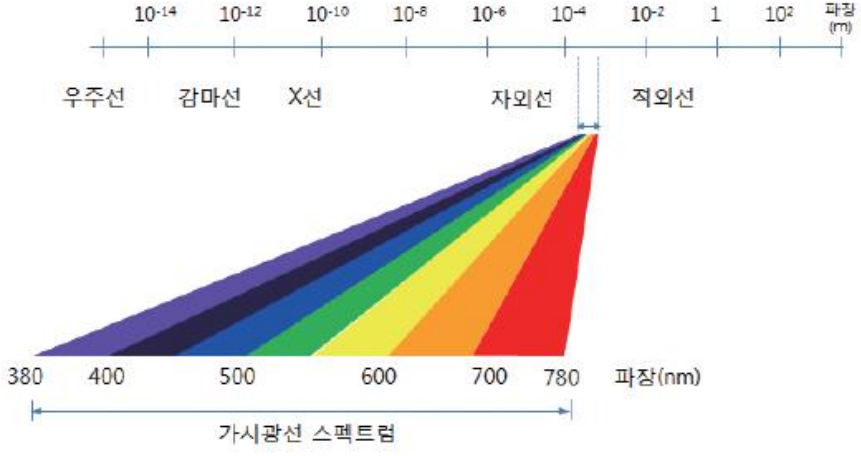
- 6.4.1 컬러 및 컬러 공간
- 6.4.2 RGB 컬러 공간
- 6.4.3 CMY(K) 컬러 공간
- 6.4.4 HSI 컬러 공간
- 6.4.5 기타 컬러 공간

6.4.1 컬러 및 컬러 공간

- 색
  - 색각으로 느낀 빛에서 주파수(파장)의 차이에 따라 다르게 느껴지는 색상



〈그림 6.4.2〉 인간이 색 인지



〈그림 6.4.3〉 가시광선의 분포

- 가시 광선
  - 전체 전자기파 중에서 인간이 인지할 수 있는 약 380 nm~780 nm사이의 파장

6.4.1 컬러 및 컬러 공간

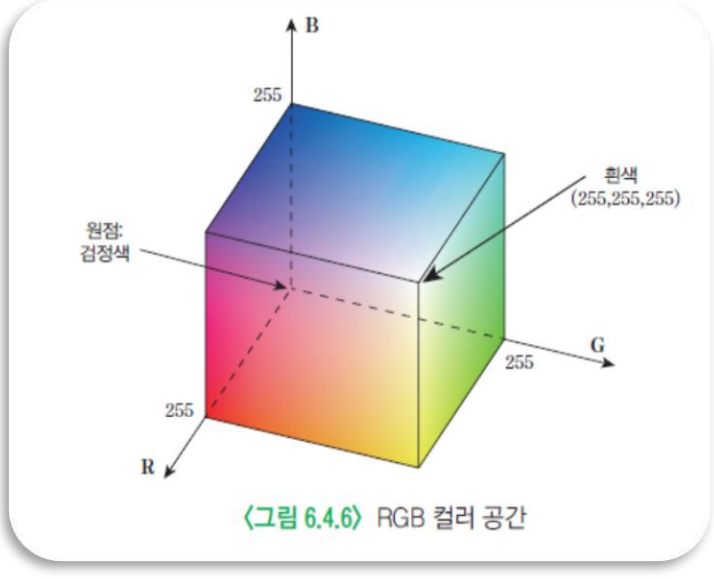
- 컬러 공간(color space)
  - 색 표시계(color system)의 모든 색들을 색 공간에서 3차원 좌표로 표현한 것
    - 색 표시계 - RGB, CMY, HSV, LAB, YUV 등의 색 체계
  - 공간상의 좌표로 표현 → 어떤 컬러와 다른 컬러들과의 관계를 표현하는 논리적인 방법제공

- 영상처리에서 컬러 공간 다양하게 활용
  - 어떤 영상에서 각각의 색상이 다른 객체를 분리하기 위해서 적절한 컬러 공간을 이용해 전체 영상을 컬러 영역별 분리
  - 전선의 연결 오류 검사를 위해 기준 색상과 비슷한 색상의 전선인지를 체크위해 사용
  - 내용 기반 영상 검색에서 색상 정보를 이용하여 원하는 물체를 검색하기 위해 특정 컬러 공간 이용

6.4.2 RGB 컬러 공간

- RGB 컬러 공간
  - 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

6.4.2 RGB 컬러 공간

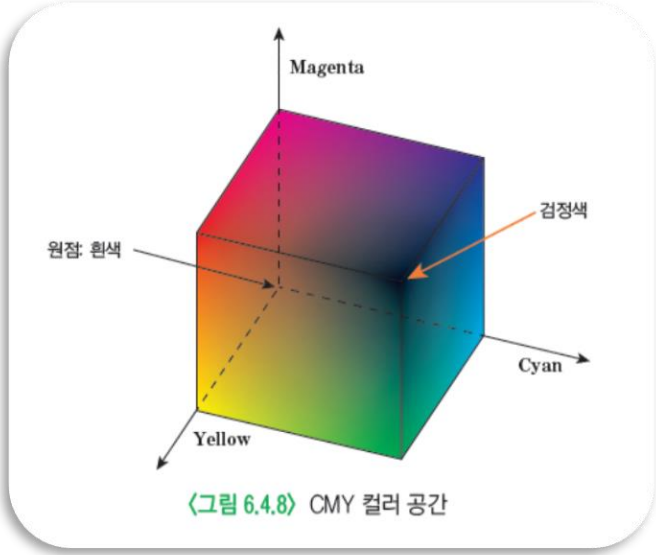
〈표 6.4.1〉 대표적인 색상에 대한 RGB 표현

RGB 화소값	색상	RGB 화소값	색상
255, 255, 255	white	240, 230, 140	khaki
0, 0, 0	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

6.4.3 CMY(K) 컬러 공간

- CMY 컬러 공간
  - 감산 혼합(subtractive mixture) – 섞을 수록 어두워지는 방식
  - 색의 삼원색(청록색(Cyan), 자홍색(Magenta), 노랑색(Yellow))으로 색 만듦
  - RGB 컬러 공간과 보색 관계

〈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

6.4.3 CMY(K) 컬러 공간

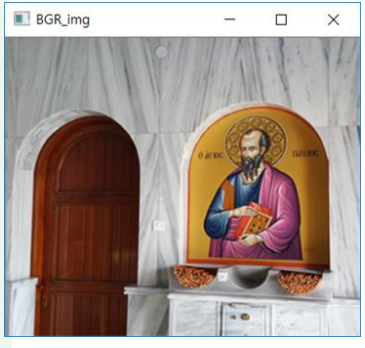
- CMYK 컬러공간
  - 순수한 검정색 사용
    - 뛰어난 대비를 제공, 검정 잉크가 컬러 잉크보다 비용이 적은 장점

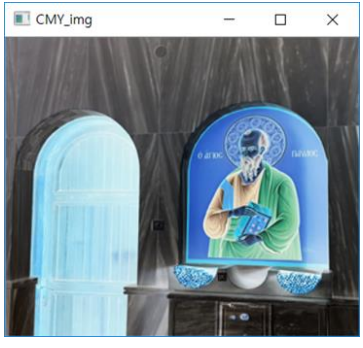
blacK	= min(Cyan, Magenta, Yellow)
Cyan	= Cyan - blacK
Magenta	= Magenta- blacK
Yellow	= Yellow - blacK

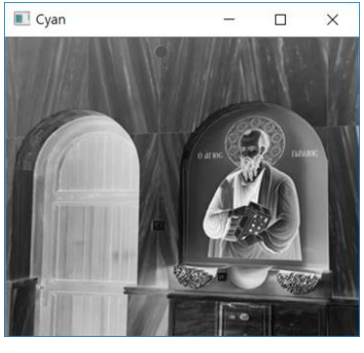
6.4.3 CMY(K) 컬러 공간

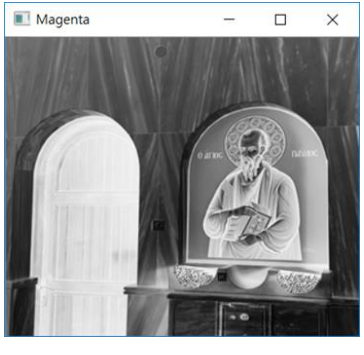
예제 6.4.1 컬러 공간 변환(BGR→CMY) – convert_CMY.py

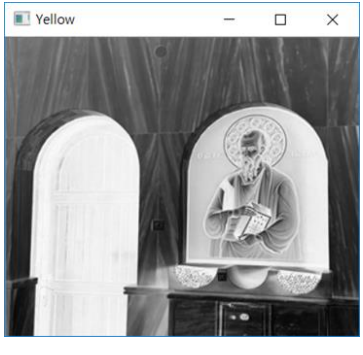
```
01  import numpy as np, cv2
02
03  BGR_img = cv2.imread("images/color_model.jpg", cv2.IMREAD_COLOR) # 컬러 영상 읽기
04  if BGR_img is None: raise Exception("영상파일 읽기 오류")
05
06  white = np.array([255, 255, 255], np.uint8)
07  CMY_img = white - BGR_img
08  Yellow, Magenta, Cyan = cv2.split(CMY_img)                    # 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)
```







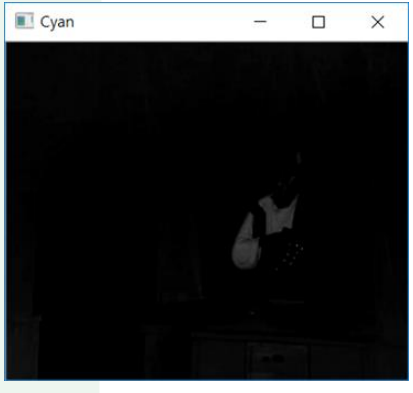


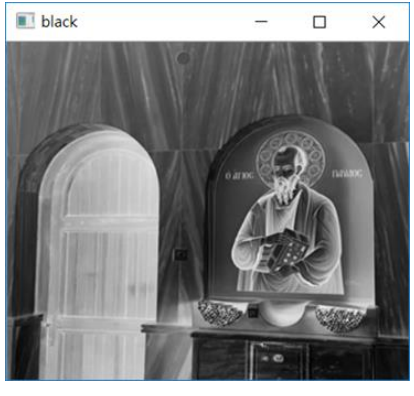


6.4.3 CMY(K) 컬러 공간

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

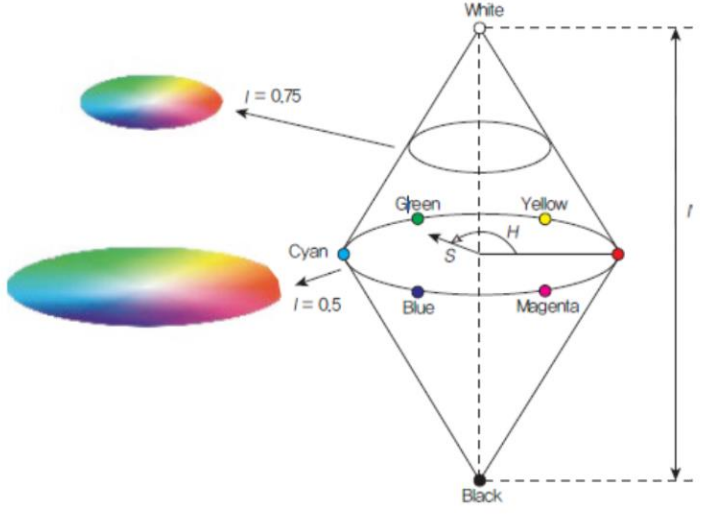
```
01  import numpy as np, cv2
02
03  BGR_img = cv2.imread("images/color_model.jpg", cv2.IMREAD_COLOR)      # 컬러 영상 읽기
04  if BGR_img is None: raise Exception("영상파일 읽기 오류")
05
06  white = np.array([255, 255, 255], np.uint8)
07  CMY_img = white - BGR_img
08  CMY = cv2.split(CMY_img)                                              # 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)
```





6.4.4 HSI 컬러 공간

- 인간이 컬러 영상 정보를 인지하는 방법
  - 색상(Hue), 채도(Saturation), 명도(Intensity,Value)라는 세 가지 지각 변수로 분류
- HSI 컬러 공간
  - 인간이 색상을 인식하는 3가지 요인인 색상, 채도, 명도를 컬러 공간으로 옮긴 것
  - 색상 - 원판의 0~360도까지 회전
    - 0도: 빨간색, 60도: 노란색,
    - 120도: 녹색, 180도: 청록(Cyan),
    - 240도: 파란색, 300도: 다홍(Magenta)
  - 채도 - 색의 순수한 정도
    - 흰색의 혼합 비율에 따라서 0~100까지의 값
  - 명도 - 빛의 세기, 색의 밝고 어두운 정도
    - 가장 아래쪽 0이 검은색, 가장 위쪽이 100으로 흰색



〈그림 6.4.9〉 HSI 컬러 공간

6.4.4 HSI 컬러 공간

• RGB → HSI 변환 수식

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

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

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

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

• OpenCV HSV 변환 수식

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

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

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

6.4.4 HSI 컬러 공간

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

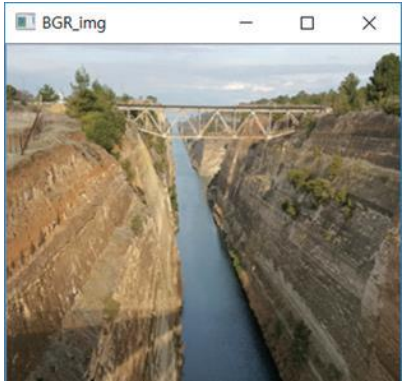
```
01 import numpy as np, cv2, math
02
03 def calc_hsi(bgr):                                          # 한 화소 hsi 계산 함수
04     # B, G, R = bgr.astype(float)                           # float 형 변환
05     B, G, R = float(bgr[0]), float(bgr[1]), float(bgr[2])   # 속도면에 유리
06     bgr_sum = (R + G + B)
07     ##    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
```

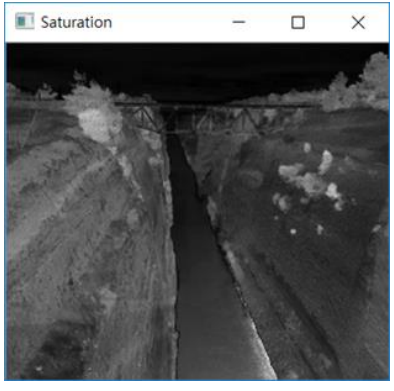
6.4.4 HSI 컬러 공간

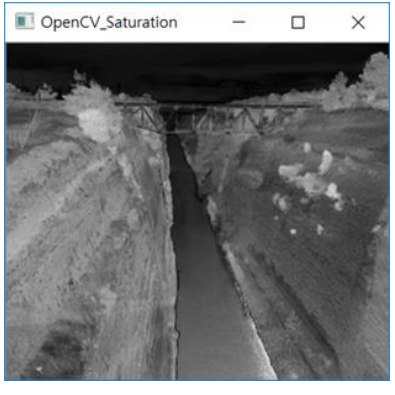
```
22  BGR_img = cv2.imread("images/color_space.jpg", cv2.IMREAD_COLOR)       # 컬러 영상 읽기
23  if BGR_img is None: raise Exception("영상파일 읽기 오류")
24
25  HSI_img = bgr2hsi(BGR_img)                                             # BGR→ HSI 변환
26  HSV_img = cv2.cvtColor(BGR_img, cv2.COLOR_BGR2HSV)                     # OpenCV 함수
27  Hue, Saturation, Intensity = cv2.split(HSI_img)                        # 채널 분리
28  Hue2, Saturation2, Intensity2 = cv2.split(HSV_img)                     # 채널 분리
29
30  titles = ['BGR_img', 'Hue', 'Saturation', 'Intensity']
31  [cv2.imshow(t, eval(t)) for t in titles]                               # User 구현 결과 영상표시
32  [cv2.imshow('OpenCV_'+t, eval(t+'2')) for t in titles[1:]]             # OpenCV 결과 영상 표시
33  cv2.waitKey(0)
```

6.4.4 HSI 컬러 공간

- 실행결과







6.4.5 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

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

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

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

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

$$
\begin{aligned}
R &= Y + 1.28033 \cdot V \\
G &= Y - 0.21482 \cdot U - 0.38059 \cdot V \\
B &= Y + 2.12798 \cdot U
\end{aligned}
$$

6.4.5 기타 컬러 공간

〈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

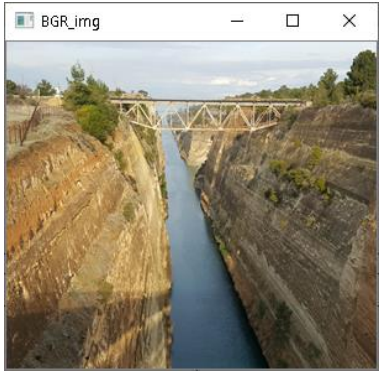
6.4.5 기타 컬러 공간

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)                                   # 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)
```

6.4.5 기타 컬러 공간

- 실행 결과



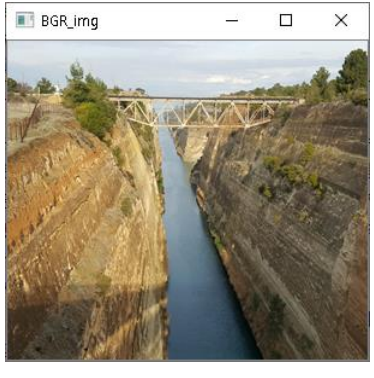
6.4.5 기타 컬러 공간

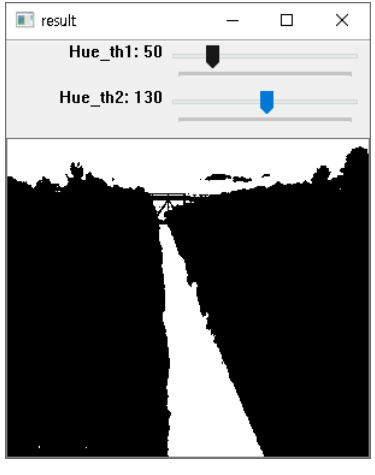
심화예제 6.4.5 Hue 채널을 이용한 객체 검출 – 17.hue_threshold.py

```
01  import numpy as np, cv2
02
03  def onThreshold(value):
04      th[0] = cv2.getTrackbarPos("Hue_th1", "result")
05      th[1] = cv2.getTrackbarPos("Hue_th2", "result")
06
07      ## 이진화– 화소 직접 접근 방법
08      # result = np.zeros(hue.shape, np.uint8)
09      # for i in range(result.shape[0]):
10      #    for j in range(result.shape[1]):
11      #       if th[0] <= hue[i, j] < th[1] : result[i, j] = 255
12
13      ## 이진화– 넘파이 함수 활용 방식
14      # result = np.logical_and(hue < th[1], hue >= th[0])
15      # result = result.astype('uint8') * 255
16
17      ## OpenCV 이진화 함수 이용– 상위 값과 하위 값 제거
18      _, result = cv2.threshold(hue, th[1], 255, cv2.THRESH_TOZERO_INV)
19      cv2.threshold(result, th[0], 255, cv2.THRESH_BINARY, result)
20      cv2.imshow("result", result)
21
```

6.4.5 기타 컬러 공간

```
22  BGR_img = cv2.imread("images/color_space.jpg", cv2.IMREAD_COLOR)      # 컬러 영상 읽기
23  if BGR_img is None: raise Exception("영상파일 읽기 오류")
24
25  HSV_img = cv2.cvtColor(BGR_img, cv2.COLOR_BGR2HSV)     # 컬러 공간 변환
26  hue = np.copy(HSV_img[:, :, 0])                        # hue 행렬에 색상 채널 복사
27
28  th = [50, 100]                                         # 트랙바로 선택할 범위 변수
29  cv2.namedWindow("result")
30  cv2.createTrackbar("Hue_th1", "result", th[0], 255, onThreshold)
31  cv2.createTrackbar("Hue_th2", "result", th[1], 255, onThreshold)
32  onThreshold(th[0])                                     # 이진화 수행
33  cv2.imshow("BGR_img", BGR_img)
34  cv2.waitKey(0)
```





단원 핵심 요약

- 1. 히스토그램 평활화(histogram equalization)는 특정 부 분에서 한쪽으로 치우친 명암 분포를 가진 영상을 히스토그램의 재분배 과정을 거쳐서 균등한 히스토그램 분포를 갖게 하는 알고리즘이다.

- 2. 컬러 공간이란 색 표시계의 모든 색들을 색 공간에서 3차원 좌표로 표현한 것이다. 따라서 컬러 공간은 공간상의 좌표로 표현되기 때문에 어떤 컬러와 다른 컬러들과의 관계를 표현하는 논리적인 방법을 제공한다.

- 3.컬러 공간에서는 모니터에서 주로 사용하는 RGB, 프린터에서 사용하는 CMY, 인간시각 시스템과 유사한 HSI 컬러공간이 있다. 또한 JPEG 등의 압축에 주로 사용하는 YCbCr, 방송시스템에 많이 사용하는 YUV 컬러공간이 있다.

6. 실습 과제

- (실습 과제) 히스토그램 평활화 구현
  - 피피티와 교과서에 나온 코드를 보지말고 아래 수식만을 이용하여 히스토그램 평활화를 구현하시오.

    - 히스토그램 평활화 과정
      - i번째 화소 값 x의 히스토그램 H는 x_i의 발생 빈도 n_i로 정의

$$H(x_i) = n_i$$

        - 비트심도가 8인 경우 i의 범위는 0~255
      - x_i의 확률 밀도 함수 p (정규화)

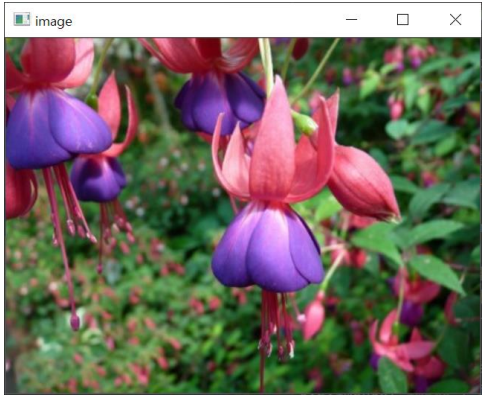
$$p(x_i) = \frac{n_i}{n}, i = 0, 1, 2, \dots, 255$$

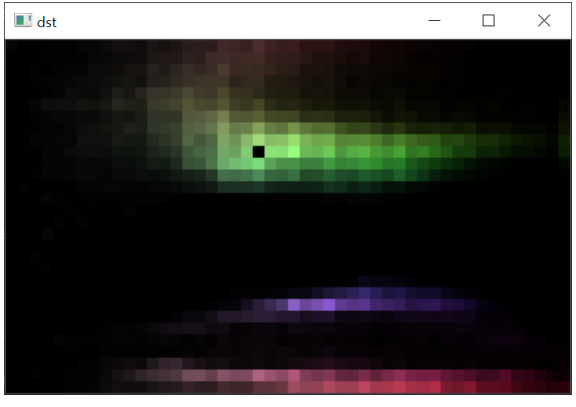
      - x_i에 대한 히스토그램 평활화 결과 값 s_i

$$s_k = 255 \times \sum_{i=0}^{k} p(x_i) = 255 \times \sum_{i=0}^{k} \frac{n_i}{n}, k = 0, 1, 2, \dots, 255$$

6. 실습 과제 part 2.

- (보너스) draw_histo() 함수 수정
  - 영상 파일을 읽어들여서 HSV 컬러 공간으로 변환하고, Hue와 Saturation 채널로 2차원 히스토그램을 구하시오. 즉, 2차원 히스토그램의 Hue(세로)와 Saturation(가로)를 2개의 축으로 구성하고, 빈도값을 밝기로 표현해서 오른쪽 그림과 같이 2차원 그래프를 그리시오.





6. 실습 규칙

- 실습 과제는 실습 시간내로 해결해야 합니다.
  - 해결 못한경우 실습 포인트를 얻지 못합니다.
  - -> 집에서 미리 예습하고 오길 권장합니다.

- 코드 공유/보여주기 금지. 의논 가능.

- 보너스문제까지 해결한 학생은 조기 퇴실 가능