제2장 영상의 표현

한밭대학교 전자공학과
곽수영

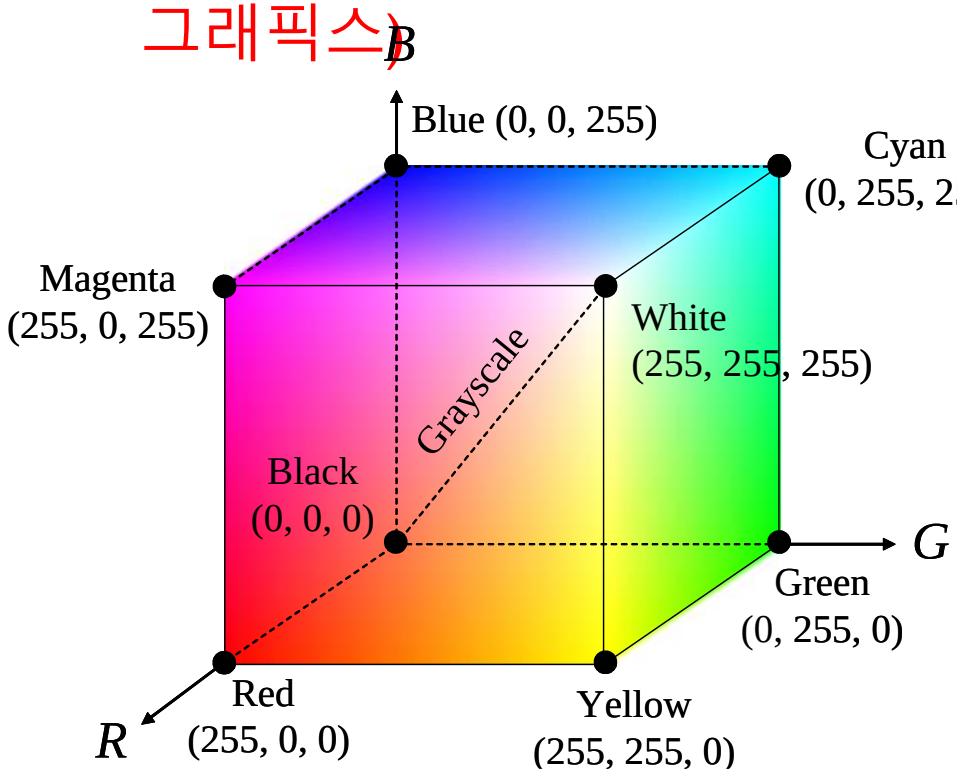
2.1 영상의 색상 표현 방법

컬러 모델

- 서로 다른 영상처리 시스템은 여러 이유 때문에 서로 다른 컬러 모델을 사용한다.
  - RGB, CMY(K), HSV, YCrCb,등이 있음
    - 컬러로 된 그림을 출판하는 기업은 CMY컬러 모델
    - 컬러 CRT모니터와 컴퓨터 그래픽스 시스템들은 RGB 컬러 모델
    - 색상, 채도, 명도를 각각 다루어야 하는 시스템들은 HSV컬러모델
    - 영상 시스템에서 사용되는YCrCb컬러모델

RGB 컬러 모델

- RGB컬러모델
  - 빛의 3원색인 Red, Green, Blue의 가산 조합으로 만들어 짐
  - 디스플레이 장치용 (개인용 PC의 CRT모니터, 컴퓨터 그래픽스)



RGB의 각 값은 0~1사이의 범위에 있는 소수를 사용하지만
그래픽 소프트웨어에서는 실제로 0~255까지의 값을 사용함

RGB 컬러 모델

- RGB컬러모델
  - 하나의 픽셀이 R, G, B 세 개의 색상 성분의 조합으로 표현되는 비트맵
  - 256^3 = 16,777,216 가지의 색상을 표현

표 1. RGB 성분을 이용한 색상 표현

색상 성분	빨간색 (Red)	녹색 (Green)	파란색 (Blue)	노란색 (Yellow)	하늘색 (Cyan)	보라색 (Magenta)	흰색 (White)	검정 (Black)
r	255	0	0	255	0	255	255	0
g	0	255	0	255	255	0	255	0
b	0	0	255	0	255	255	255	0

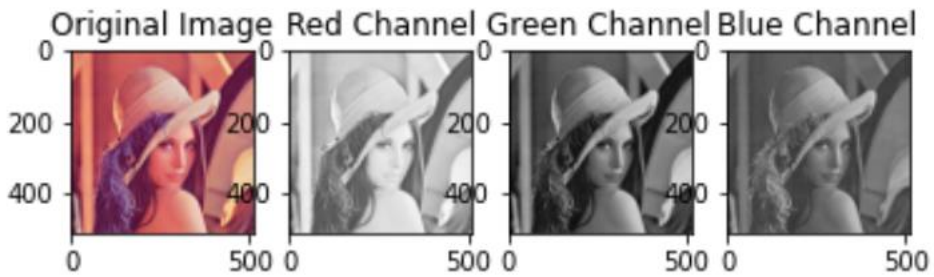
RGB 컬러 모델

- 연습2-1) R,G,B 각 채널로 분할하기

연습2-1) RGB 컬러 영상을 Red, Green, Blue 3 채널로 분리하기

```
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
origin_img = cv2.imread('/content/gdrive/My Drive/Image_Processing/lena.jpg')
# BGR 채널순서를 RGB 채널로 변경
RGB_img = cv2.cvtColor(origin_img, cv2.COLOR_BGR2RGB)
# R,G,B 채널로 분할하기
Red_img, Green_img, Blue_img = cv2.split(RGB_img)
#그림을 화면에 출력
plt.subplot(1, 4, 1) # 1행 4열에서 1번째 열
#원 영상 출력
plt.title("Original Image")
plt.imshow(RGB_img)
```

```
# R채널만 출력
RGB_img[:,:,0] = Red_img
RGB_img[:,:,1] = Red_img
RGB_img[:,:,2] = Red_img
plt.subplot(1, 4, 2) # 1행 4열에서 2번째 열
plt.title("Red Channel")
plt.imshow(RGB_img)
# G 채널만 출력
RGB_img[:,:,0] = Green_img
RGB_img[:,:,1] = Green_img
RGB_img[:,:,2] = Green_img
plt.subplot(1, 4, 3) # 1행 4열에서 3번째 열
plt.title("Green Channel")
plt.imshow(RGB_img)
# B 채널만 출력
RGB_img[:,:,0] = Blue_img
RGB_img[:,:,1] = Blue_img
RGB_img[:,:,2] = Blue_img
plt.subplot(1, 4, 4) # 1행 4열에서 4번째 열
plt.title("Blue Channel")
plt.imshow(RGB_img)
plt.show()
```



코드설명

import numpy as np: NumPy는 Numerical Python의 줄임말로 C언어로 구현된 파이썬 라이브러리로 고성능 수치계산을 위해 만들어졌다. NumPy는 array 단위로 벡터 및 행렬 연산에 있어서 매우 편리한 기능을 제공하며 데이터분석에 사용되는 라이브러리인 pandas와 matplotlib의 기반으로 사용되기도 한다.

import matplotlib.pyplot as plt: matplotlib라이브러리의 pyplot 모듈은 여러 그래프 관련 함수를 사용해서 그래프를 만들고 변화를 줄 수 있다.

RGB_img = cv2.cvtColor(origin_img, cv2.COLOR_BGR2RGB): 불러온 영상의 채널 순서를 BGR에서 RGB순으로 바꾼다. 영상은 기본적으로 한 픽셀이 24비트일 때 BGR 순서로 8비트씩 구성되어 있는데 영상처리에서 사용하기 편하도록 RGB순서대로 변화시켜 사용한다. cvtColor 함수에 의해 RGB_img는 영상의 (높이, 너비, 채널수)shape의 배열이 된다. 코드를 구현하고 마우스를 가져다 대면 `ndarray with shape (512, 512, 3)`와 같이 출력되는 것을 알 수 있다. 결국 512x512x3의 3차원 배열이 있다고 생각하면 된다.

Red_img[0:512,0:512,1] = 0: 0~512 높이, 0~512 너비까지, 채널 1번 (1번째 2차원 배열)에 해당하는 값들을 0으로 만든다. 따라서 Green 채널에 해당하는 2차원 배열값은 모두 0이 된다.

Green_img[:,:,0] = 0 : 영상의 너비, 폭에 아무것도 적지 않으면 디폴트로 0~512(영상 크기)로 세팅된다. 채널 0번 (Red)에 해당하는 2차원 배열값을 모두 0으로 변환한다.

plt.subplot(1, 4, 1): 그림 배치를 위한 테이블과 인덱스를 지정한다. 기본형은 `subplot(nrows, ncols, index, **kwargs)` 이다.

plt.imshow(RGB_img): 영상을 화면에 출력한다. RGB_img는 RGB로 변형된 원 영상이다.

plt.title("Original Image"): 영상위에 이름을 출력해준다.

채널 분리 & 병합

- 채널 분리 함수 cv2.split
  - 영상이나 이미지의 색상 공간의 채널을 분리
  - OPENCV의 가산 혼합의 삼원색 기본 배열 순서는 BGR
    - mv = cv2.split(src)는 입력 이미지(src)에서 채널을 분리해 단일 채널 이미지 배열(mv)을 생성
    - mv는 목록(list) 형식으로 반환되며, b, g, r 등으로 형태로 각 목록의 원소값을 변수로 지정할 수 있음
    - 분리된 채널의 순서에 맞게 각 변수에 할당
    - 분리된 채널들은 단일 채널이므로 흑백 색상으로 표현

```
from google.colab.patches import cv2_imshow
from google.colab import drive
import cv2
drive.mount('/content/gdrive')
src = cv2.imread('/content/gdrive/My Drive/Image_Processing/lena.jpg')
cv2_imshow(src)
b,g,r = cv2.split(src)
cv2_imshow(b)
cv2_imshow(g)
cv2_imshow(r)
```

채널 분리 & 병합

- 채널 병합 함수 cv2.merge
  - 분리된 채널을 병합해 하나의 이미지로 합칠수 있음.
    - dst = cv2.merge(mv)로 단일 채널 이미지 배열(mv)를 병합해 출력 이미지(dst)를 생성
    - 채널을 변형한 뒤에 다시 합치거나 순서를 변경해 병합할 수 있음.
    - 순서가 변경될 경우, 원본 이미지와 다른 색상으로 표현될 수 있음.

```
from google.colab.patches import cv2_imshow
from google.colab import drive
import cv2
drive.mount('/content/gdrive')
src = cv2.imread('/content/gdrive/My Drive/Image_Processing/lena.jpg')
cv2_imshow(src)
b,g,r = cv2.split(src)
inverse = cv2.merge((r,g,b))
cv2_imshow(inverse)
```

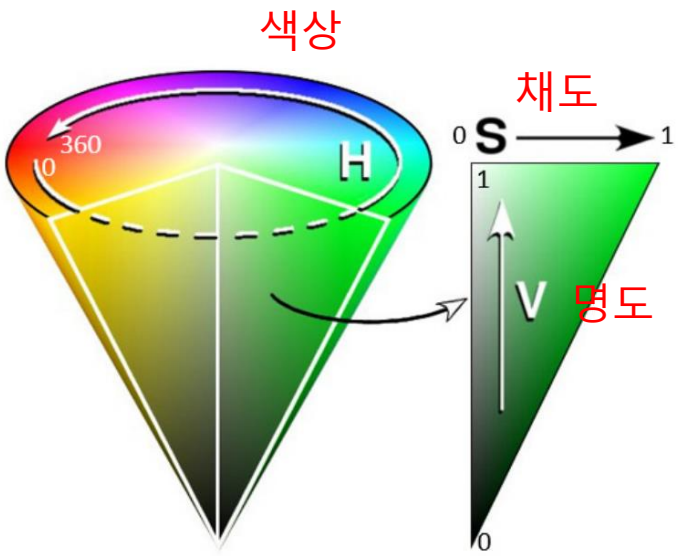
채널 분리 & 병합

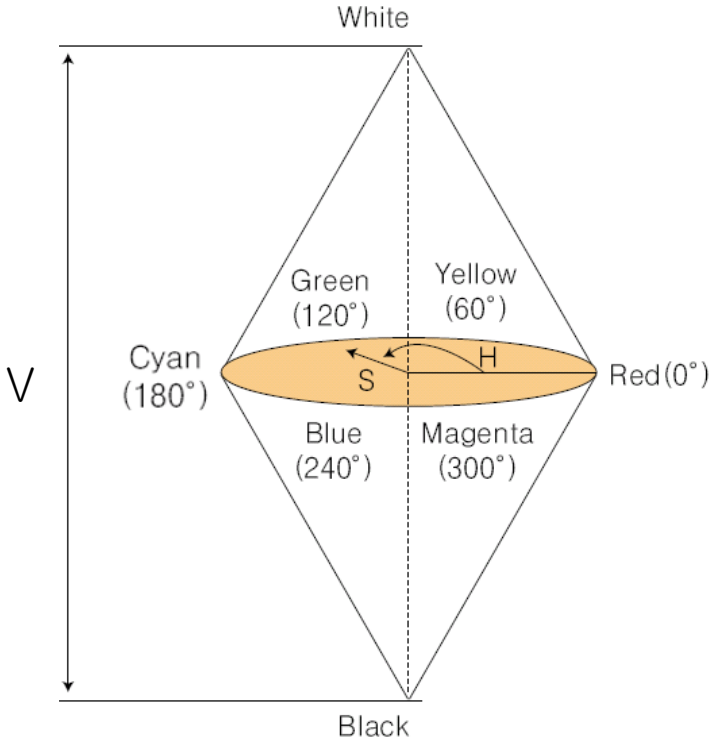
- numpy 형식 채널 분리
  - 이미지[높이, 너비, 채널]을 이용하여 특정 영역의 특정 채널만 불러올 수 있음.
  - :, :, n을 입력할 경우, 이미지 높이와 너비를 그대로 반환하고 n번째 채널만 반환하여 적용
  - src[..., n]의 형태로도 사용 가능

```
from google.colab.patches import cv2_imshow
from google.colab import drive
import cv2
drive.mount('/content/gdrive')
src = cv2.imread('/content/gdrive/My Drive/Image_Processing/lena.jpg')
cv2_imshow(src)
#b,g,r = cv2.split(src)
b = src[:, :, 0]
g = src[:, :, 1]
r = src[:, :, 2]
cv2_imshow(b)
cv2_imshow(g)
cv2_imshow(r)
```

HSV 컬러 모델

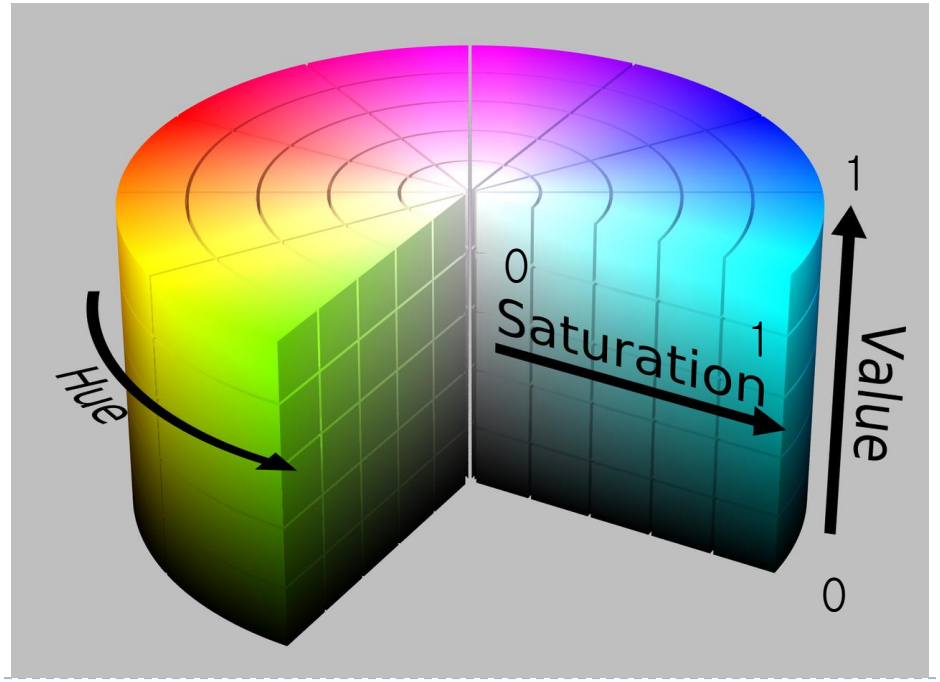
- HSV 컬러모델
  - HSV = Hue(색상), Saturation(채도), Value(명도)
  - 인간이 인식하는 색상과 흡사한 색상 모델

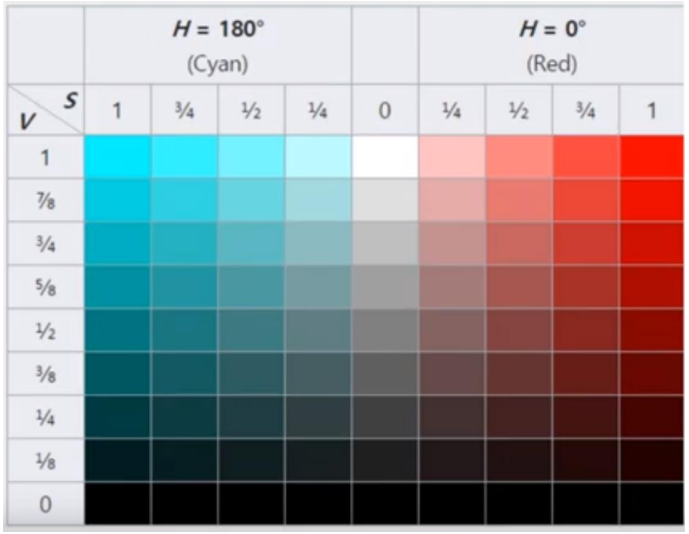




HSV 컬러 모델

- 채도(Saturation): 색의 순수도 또는 순수한 색이 흰빛에 의해 희석된 정보의 척도를 0-1사이의 값으로 나타냄 (S가 작을수록 흰색에 가까움)
- 밝기(Value): 색의 밝기 (V가 작을수록 검정색에 가까움)





HSV 컬러 모델

- RGB → HSV 공식
  - R,G,B값은 0~1사이로 정규화 된 값
  - V와 S는 0~1의 범위를 가지고 H는 0~360의 범위를 가짐

$$V = \max(R, G, B) \tag{2-1}$$

$$S = \begin{cases} \dfrac{V - \min(R, G, B)}{V} & \text{if } V \neq 0 \\ 0 & \text{if } V = 0 \end{cases} \tag{2-2}$$

$$H = \begin{cases} \dfrac{60(G - B)}{V - \min(R, G, B)} & \text{if } V = R \\ 20 + \dfrac{60(B - R)}{V - \min(R, G, B)} & \text{if } V = G \\ 240 + \dfrac{60(R - G)}{V - \min(R, G, B)} & \text{if } V = B \end{cases} \tag{2-3}$$

$$\text{if } H < 0, H = H + 360 \tag{2-4}$$

HSV 컬러 모델

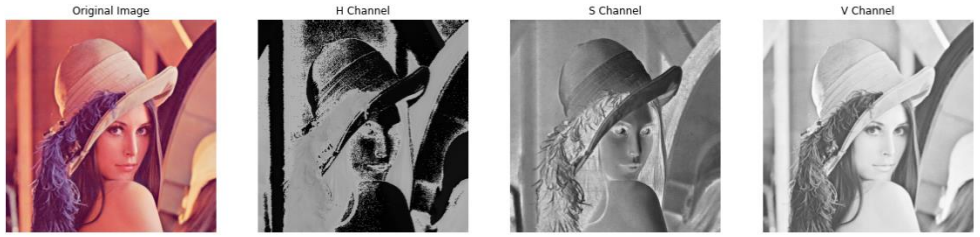
- HSV 값은 각각 최대값이 360, 1, 1의 값을 각각 가지므로 H/360.0× 255.0, S×255.0,V×255.0의 곱을 통해 화면에 표시

(그림 2-4) H, S, V 각 채널로 분할된 영상

연습 2-2) RGB 컬러 영상을 H, S, V 채널로 분리하기

```
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
origin_img = cv2.imread('/content/gdrive/My Drive/Image_Processing/lena.jpg')
# BGR채널순서를 RGB채널로 변경
RGB_img = cv2.cvtColor(origin_img, cv2.COLOR_BGR2RGB)
# RGB채널을 HSV채널로 변경
HSV_img = cv2.cvtColor(RGB_img, cv2.COLOR_RGB2HSV)
# H, S, V 채널로 나누기
H_img, S_img, V_img = cv2.split(HSV_img)
#그림을 화면에 출력
plt.figure( figsize=(20,20) ) # 영상의 크기를 키워주자
plt.subplot(1, 4, 1) # 1행 4열에서 1번째 열
plt.title("Original Image")
plt.imshow(RGB_img)
plt.axis("off")    #영상에 눈금이 생기는 것을 지우는 명령어
```

```
plt.subplot(1, 4, 2) # 1행 4열에서 2번째 열
RGB_img[:,:,0] = H_img
RGB_img[:,:,1] = H_img
RGB_img[:,:,2] = H_img
plt.title("H Channel")
plt.imshow(RGB_img)
plt.axis("off")
plt.subplot(1, 4, 3) # 1행 4열에서 3번째 열
RGB_img[:,:,0] = S_img
RGB_img[:,:,1] = S_img
RGB_img[:,:,2] = S_img
plt.title("S Channel")
plt.imshow(RGB_img)
plt.axis("off")
plt.subplot(1, 4, 4) # 1행 4열에서 4번째 열
RGB_img[:,:,0] = V_img
RGB_img[:,:,1] = V_img
RGB_img[:,:,2] = V_img
plt.title("V Channel")
plt.imshow(RGB_img)
plt.axis("off")
plt.show()
```



연습 2-2) RGB 컬러 영상을 H, S, V 채널로 분리하기

```
# BGR채널순서를 RGB채널로 변경
RGB_img = cv2.cvtColor(origin_img, cv2.COLOR_BGR2RGB)
# RGB채널을 HSV채널로 변경
HSV_img = cv2.cvtColor(RGB_img, cv2.COLOR_RGB2HSV)
# H, S, V 채널로 나누기
H_img, S_img, V_img = cv2.split(HSV_img)
```

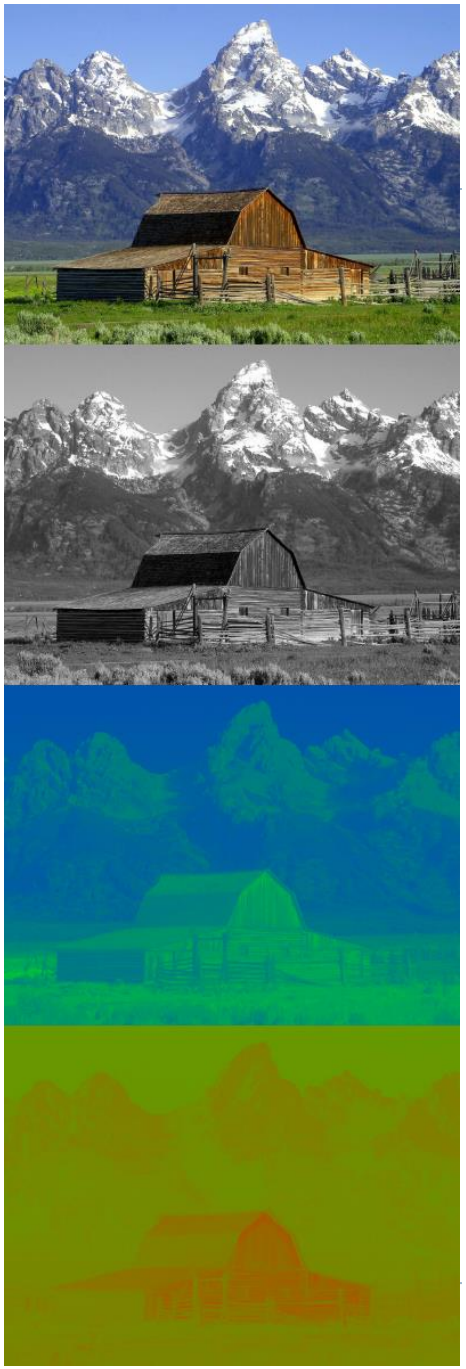
코드설명

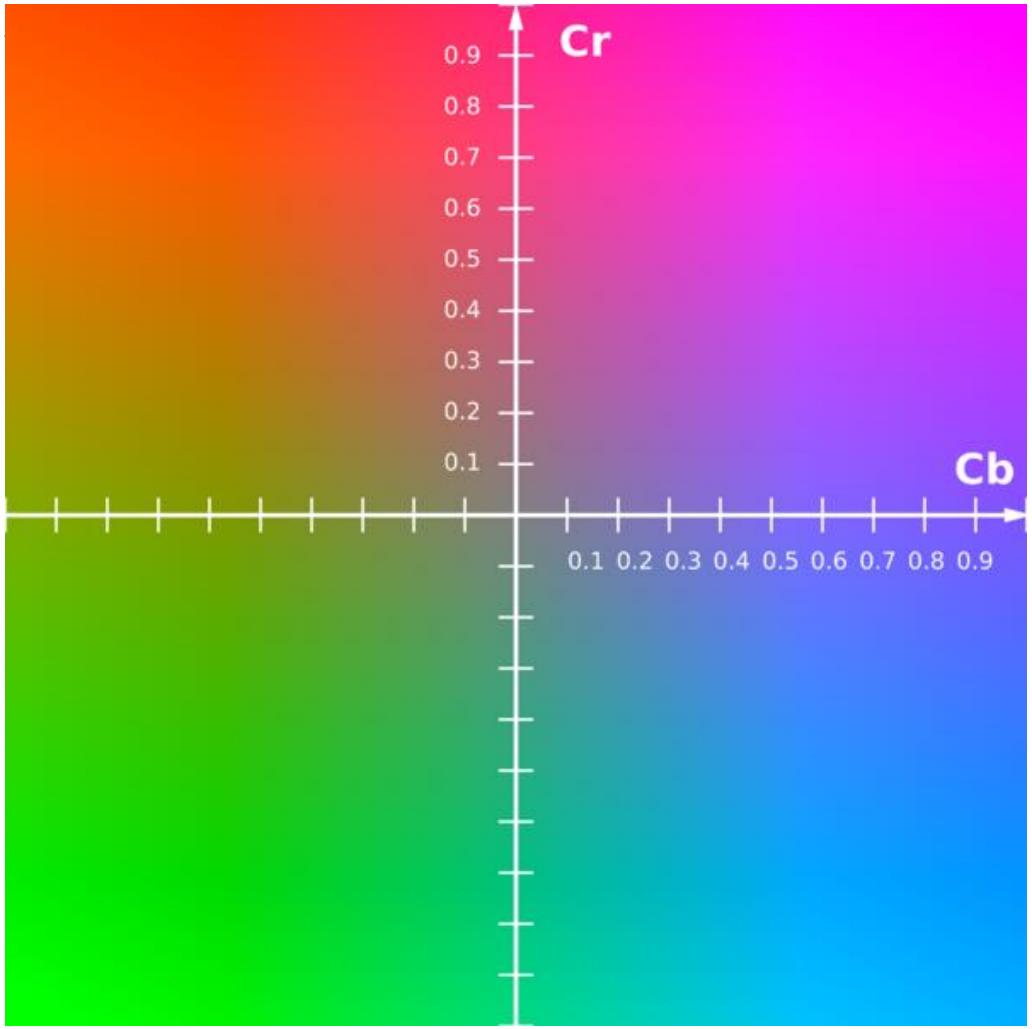
HSV_img = cv2.cvtColor(RGB_img, cv2.COLOR_RGB2HSV): 불러온 영상의 채널 순서를 RGB에서 HSV순으로 바꾼다. 따라서 HSV의 shape은 (512, 512, 3)이 되고 (*,*,0)은 H 채널 색상, (*,*,1)은 S채널 색상, (*,*,2)는 V채널 색상이 배정된다.

H_img, S_img, V_img = cv2.split(HSV_img): RGB 채널을 분할하는 것과 다르게 split() 함수를 사용해서 쉽게 각 채널별(8비트씩)로 분할이 가능하다. **split()**함수를 이용해서 채널을 분리한 경우와 연습 2-1처럼 분할하는 방법의 차이를 연습문제를 통해 알아보자. 반대로 cv2.merge((채널1, 채널2, 채널3))을 이용하여 나눠진 채널을 다시 원래 영상으로 병합할 수 있다.

YCbCr 컬러 모델

- YCbCr 컬러모델
  - 영상 시스템에서 사용되는 색공간의 일종
  - 컬러정보로부터 밝기값과 색차 신호를 분리하여 표현하는 칼라 모델
  - Y: 명도(밝기값)
  - Cb: B-Y (색차: 파랑색과의 차)
  - Cr: R-Y (색차: 빨강색과의 차)
  - 정지영상 압축 표준 방식인 JPEG와 동영상 압축 표준 방식인 MPEG에서 사용




Cr
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
Cb
0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9

YCbCr 컬러 모델

- RGB → YCbCr 공식

$$Y = 0.29900*R + 0.58700*G + 0.11400*B$$

$$Cr = 0.50000R - 0.41869G - 0.08131B$$

$$Cb = -0.16874R - 0.33126G + 0.50000B$$

- YCbCr→ RGB 공식

$$R = 1.0000*Y + 1.40200*Cr$$

$$G = 1.0000*Y - 0.34414*Cb - 0.71414*Cr$$

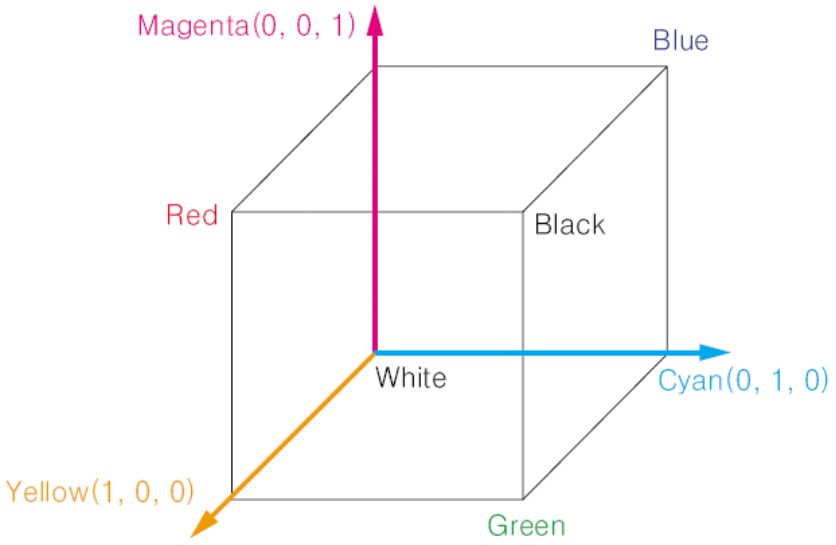
$$B = 1.0000*Y + 1.77200*Cb$$

연습 2-3) RGB 컬러 영상을 Y, Cr, Cb 채널로 분리하기

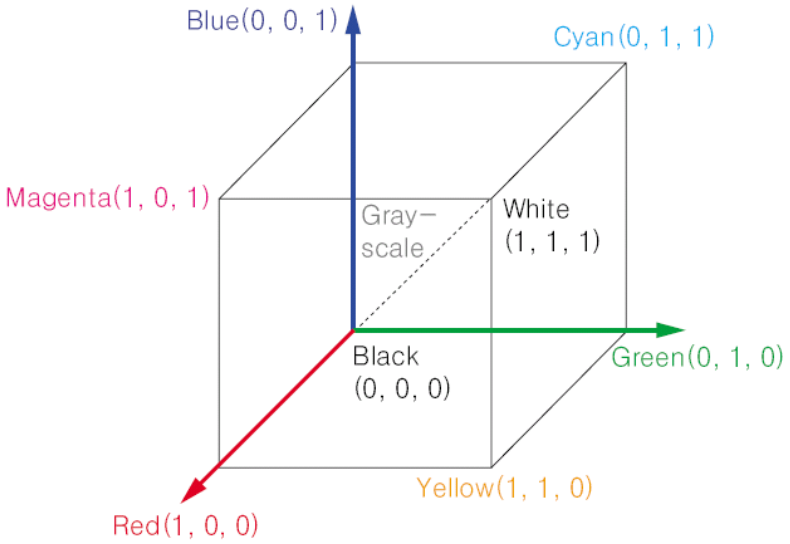
```
--------------------영상 읽기 코드 동일-----------------
# BGR채널순서를 RGB채널로 변경
RGB_img = cv2.cvtColor(origin_img, cv2.COLOR_BGR2RGB)
# RGB채널을 YCrCb채널로 변경
YCrCb_img = cv2.cvtColor(RGB_img, cv2.COLOR_RGB2YCrCb)
# Y, Cr, Cb 채널로 나누기
Y_img, Cr_img, Cb_img = cv2.split(YCrCb_img)
--------------------출력 코드 동일--------------------
```

CMY 컬러 모델

- CMY 컬러 모델
  - 청록색(Cyan), 자홍색(Magenta), 노랑색(Yellow)을 기본색으로 사용
  - RGB 컬러 모델에서 대각선으로 마주보는 색의 모양을 서로 바꿔 놓은 것처럼 보임
  - 프린트 등 잉크를 사용해 출력되는 장치에서 사용하는 색 (인쇄장치용)



[그림 2-7] CMY 컬러 모델



[그림 2-5] RGB 컬러 모델

CMY 컬러 모델

- CMY 컬러 모델
  - C, M, Y 세 가지 색을 더하면 검정색(1, 1, 1)이 되어 색의 밝기가 낮아지는 감산체계(Subtractive System) 사용

(a) CMY 컬러 영상 (b) Cyan 채널 영상 (c) Magenta 채널 영상 (d) Yellow 채널 영상

[그림 2-8] 컬러 영상에서 CMY 채널 분리

Cyan (청록)
Magenta (자홍)
Yellow (노랑)

CMY 컬러 모델

- CMY 컬러 모델
  - RGB 컬러와는 정반대 공간에 위치하므로, 청록색-빨강색, 자홍색-초록색, 노랑색-파란색은 보색(Complement) 관계
  - RGB → CMY상으로 변환

$$\begin{bmatrix} C \\ M \\ Y \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

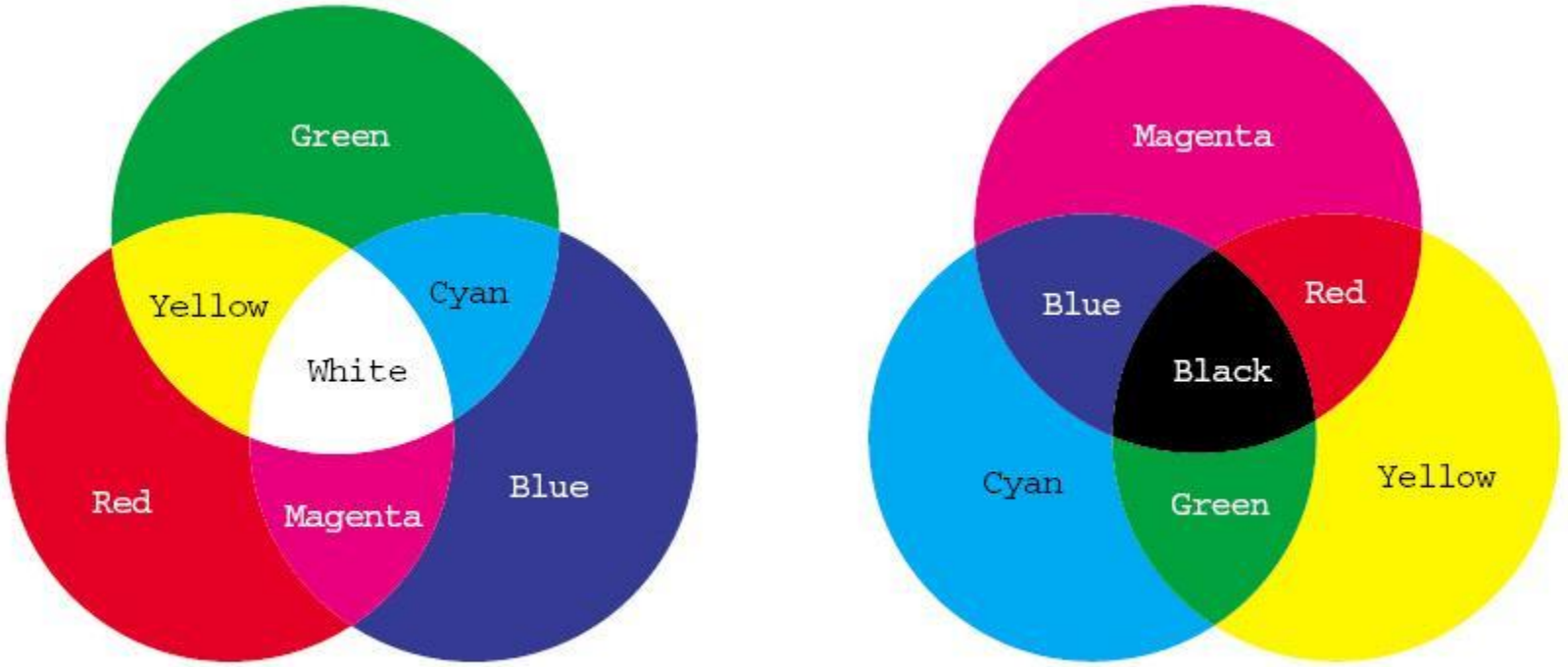
Cyan (청록)
Magenta (자홍)
Yellow (노랑)

  - CMY → RGB 상으로 변환

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} C \\ M \\ Y \end{bmatrix}$$

CMY 컬러 모델

- 컬러모델의 가산과 감산체계



[그림 2-9] 컬러 모델의 가산과 감산체계

2.2 디지털 영상의 구조

디지털 영상 종류

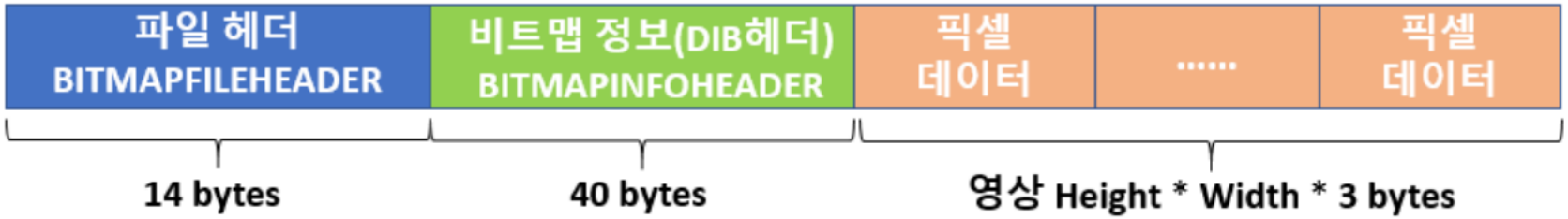
- RAW 영상
  - 영상 구성(Configuration)에 대한 헤더(Header)정보를 포함하지 않고 영상의 픽셀값만을 가지고 있는 파일
- BMP(Bitmap) 영상
  - MS에서 Windows기반 운영체제에서 사용할 목적으로 만든 영상 저장 방식으로 영상의 픽셀값을 그대로 저장 함. Header, palette, pixel값으로 구성
- JPEG(Joint Photographic Experts Group)
  - 정지 영상을 위해서 만들어진 손실 압축 방법 표준임
- PNG(Portable Network Graphics)
  - 최근의 소셜미디어 또는 고화질 그래픽이 필요한 포트폴리오 등에서 많이 사용
  - 무손실 압축방식, 파일의 용량이 커지는 단점

BMP 영상

- BMP 영상 저장 방식
  - 기본적으로 1~24bit의 색을 표현할 수 있지만 알파 채널(Alpha channel)을 포함한 32bit 포맷이 사용되기도 함
  - 장치 의존 비트맵(DDB: Dependent Bitmap)
    - 모니터(출력 장치)에 나타나는 영상이 모니터의 화면에 설정된 값에 의해 출력되는 방식
  - 장치 독립 비트맵(DIB: Device Independent Bitmap)
    - 출력 장치의 설정이 달라지더라도 자신의 비트맵 값이 그대로 출력되도록 하는 방식으로 대부분의 BMP 파일은 DIB 방식으로 저장되어 있음

BMP 영상

- BMP 영상 파일 구조



- BMP 헤더: BMP 파일에 대한 일반 정보를 담고 있음
- 비트맵 정보(DIB 헤더): 비트맵 그림에 대한 자세한 정보를 담고 있음
- 색 팔레트: 인덱스 컬러 비트맵에 쓰이는 색의 정의를 담고 있음
- 비트맵 데이터: 픽셀 단위의 실제 그림을 담고 있음

BMP 영상

- BMP헤더는 BITMAPFILEHEADER 구조체로 표현

```
typedef struct tagBITMAPFILEHEADER
{
    WORD      bfType;   // "BM"이라는 글자가 설정됨 (16진수 0x42 or 0x4D)
    DWORD     bfSize;   // 비트맵 파일의 전체 크기 (바이트 단위)
    WORD      bfReserved1;   // 예약변수(0으로 설정함)
    WORD      bfReserved2;   // 예약변수(0으로 설정함)
    DWORD     bfOffBits;      // 파일에서 비트맵 데이터가 있는 위치 (바이트 단위의
                                시작 주소)
} BITMAPFILEHEADER;
// 총 14 BYTES = 2+4+2+2+4 (byte)
```

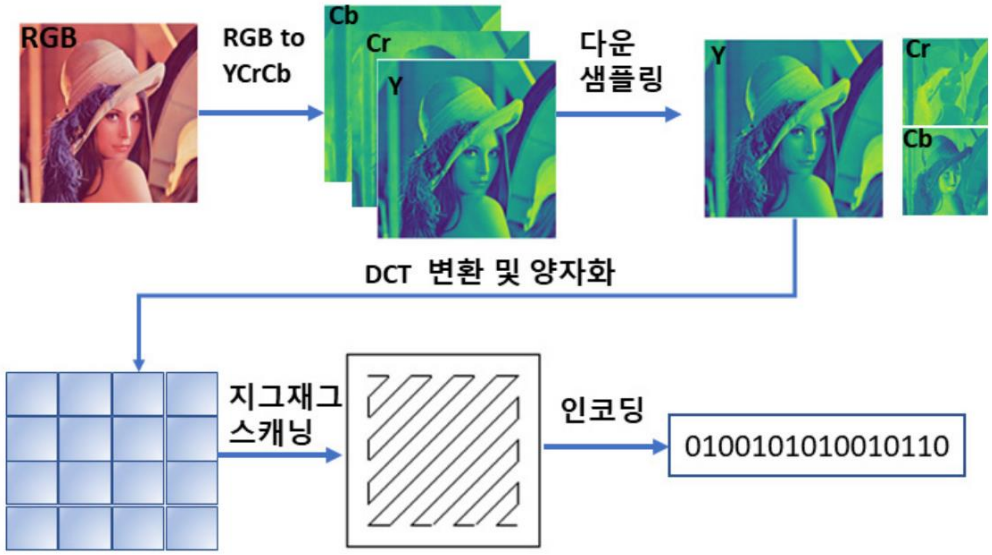
BMP 영상

- 비트맵 정보는 BITMAPINFOHEADER 구조체로 표현

```
typedef struct tagBITMAPINFOHEADER
{
    DWORD biSize;  // 이 구조체를 저장하기 위해 필요한 바이트 수
    LONG biWidth;  // 비트맵의 가로 크기 (픽셀단위)
    LONG biHeight;  // 비트맵의 세로 크기 (픽셀단위)
    WORD biPlanes;  // 비트맵을 화면에 보여줄 때 필요한 Plane수(1로 설정함)
    WORD biBitCount;  // 한 픽셀 당 비트수 (1,4,8,16,24,32)
    DWORD biCompression;  // 압축 유무 플래그(압축하지 않으면 BI_RGB)
    DWORD biSizeImage;  // 영상 데이터의 크기를 바이트 단위로 표시
    LONG biXPelsPerMeter;  // 미터 당 가로 픽셀 수 (영상의 가로 해상도)
    LONG biYPelsPerMeter;  // 미터 당 세로 픽셀 수 (영상의 세로 해상도)
    DWORD biClrUsed;  // 그림에서 실제 사용되는 컬러 수
    DWORD biClrImportant;  // 중요하게 사용되는 컬러
} BITMAPINFOHEADER;
// 총 40 BYTES = 4+4+4+2+2+4+4+4+4+4+4 (byte)
```

JPEG 영상

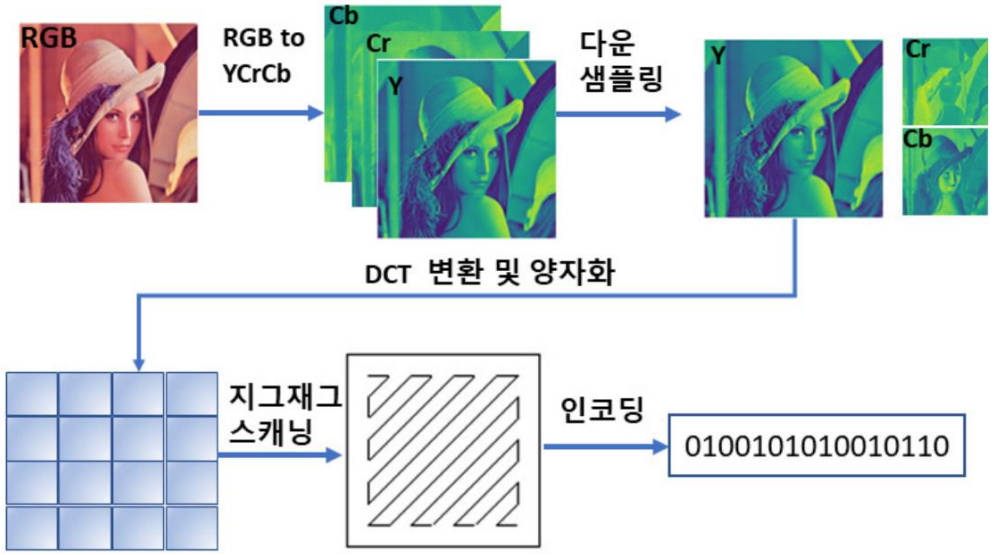
- JPEG 영상 압축 단계
  - 1) 화소값들을 RGB에서 YCrCb 방식으로 표현
  - 2) 크로마 서브샘플링(chroma subsampling, down sampling) 과정
    - 사람은 색상의 변화보다는 밝기 변화에 더 민감하기 때문에 색상변환 후에 각 색상별 로 압축률을 다르게 적용
    - Y채널은 그대로 두고 Cr과 Cb 채널은 압축률에 따라 선택되는 픽셀의 개수를 줄임



(그림 2-7) JPEG 파일 형식 변환 과정 [13)]

JPEG 영상

- JPEG 영상 압축 단계
  - 3) DCT변환 및 양자화 적용하여 저주파 성분은 유지하고 고주파 성분을 줄임
  - 4) Zig-Zag 스캐닝 방법을 이용하여 DCT행렬값들을 벡터에 저장
  - 5) DC계수는 허프만 코딩 방식을 이용하여 부호화하고,AC성분은 Run-Length와 허프만 코딩을 이용하여 부호화



(그림 2-7) JPEG 파일 형식 변환 과정 [13)]