chap 10 컬러 영상 처리

컬러 영상 다루기

컬러 영상의 픽셀 값 참조

- openCV에서 영상 파일을 불러와 객체 생성시 imread()함수 사용. ->이때 imread()함수의 두 번째 인자를 IMREAD_COLOR로 설정하면 영상을 3채널 컬러 영상 형식으로 불러옴
- 일반적으로 컬러영상은 흔히 RGB로 표현하지만 OpenCV에서는 BGR 색상 순서로 픽셀값을 표현 -> imread()로 불러오면 색상 값이 BGR순서로 저장된 객체가 생성

- 컬러 영상에서 각각의 B,G,R성분은 0부터 255사이의 값을 가짐. 0인 경우에는 해당 색 성분이 전혀 없고, 255면 가득 찬 상태.
- 컬러영상에서 1개의 픽셀은 3개의 색상 성분을 가지고 있음

컬러 영상의 픽셀값 참조 예제

```
def color_op():
    src = cv.imread()

print('The pixel value [B, G, R] at (0,0) is ', src[0,0])
```

컬러 영상 반전 예제

```
def color_inverse():
    src = imread()

    dst = np.zeros(src.shape, src.dtype)

    for j in range(src.shape[0]):
        for i in range(src.shape[1]):
            p1 = src[j,i]
            p2 = dst[j,i]

            p2[0] = 255 - p1[0] # B
            p2[1] = 255 - p1[1] # G
            p2[2] = 255 - p1[2] # R
```

- B,G,R 세개의 색상 성분을 각각 255에서 빼는 연산을 수행.
- 결과 이미지를 저장할 공간을 np.zeros로 초기화 하고 한 픽셀 단위로 loop을 돌면서 반전 영상 수행

색 공간 변환

- 빨간색, 녹색, 파란색 세 가지 색 성분의 조합으로 색을 표현하는 방식을 RGB 색 모델 또는 RGB 색 공간이라 함.
- 컬러 영상 처리에서는 보통 색상 구분이 용이한 HSV, HSL 색 공간을 사용.
- 휘도 성분이 구분되어 있는 YCrCb, YUV등 색 공간을 사용하는 것도 유리
- OpenCVd에서는 BGR순서의 컬러영상을 다른 색공간으로 변환하는 인터페이스를 제공

cvtColor()함수

-> 인자로 다양한 색 공간 기능 제공

->BGR2GRAY

- BGR2GRAY 컬러영상을 그레이스케일 영상으로 변환하기 위해 주로 사용. (연산속도와 메모리 사용량을 줄이기 위한 목적)
- 색상정보의 활용도가 높지 않은 경우 그레이스케일 영상 변환하여 처리하는것이 효율적
- BGR영상을 그레이스케일로 변환할때

-> GRAY2BGR

- 그레이스케일 영상을 BGR컬러 영상으로 변환할 떄 사용
- 색상 성분값은 B,G,R에 동일하게 값을 주어 처리함.

-> BGR2HSV

- HSV 색 모델은 색상, 채도, 명도로 색을 표현하는 방식.
- 색상은 색의 종류를 의미하고, 채도는 색의 순도를 나타냄. 명도는 빛의 세기를 나타냄.
- HSV 색 공간 모형에서 색상은 원뿔을 가로로 잘랐을때 나타나는 원형에서의 각도로 정의됨. 각도가 0도부터 시작 노,초,파,보를 거쳐 360도에 가까워지면 다시 빨간색
- 채도는 원뿔을 가로로 잘랐을 때 나타나는 원 모양의 중심에서 최솟값을 갖음.
- 명도는 원뿔 아래쪽 꼭지점에서 최솟값을 갖고, 원뿔의 축을 따라 올라가면서 증가.
- BGR영상을 HSV영상으로 변환할 경우 H값은 0부터 179사이 정수로 표현, S,V는 0부터 255사이의 정수로 표현 -> 색상값은 0부터 360까지 각도로 표현하지만, opencv에서는 자료형이 256이상을 표현 하지 못해서 2로 나눈값으로 H값 사용.
    - 그러나 float자료형이 입력으로 온 경우에는 표현 가능

-> BGR2YCrCb

- YCrCb 색 공간에서 Y성분은 밝기 또는 휘도 정보를 나타냄.
- Cr과 Cb성분은 색상 또는 색차 정보를 나타냄.

- RGB색상 성분으로부터 Y성분을 계산하는 공식은 그레이스케일 계산 공식과 완전히 같고, Cr과 Cb성분은 밝기에 대한 정보는 포함되지 않고, 오직 색상에 대한 정보만 갖고 있음.
- YCrCb 색 공간은 영상을 그레이스케일 정보와 색상 정보로 분리하여 처리할 때 유용함.
- BGR2YCrCb 시에 Y,Cr,Cb 각각의 성분 값이 0부터 255사이의 값으로 표현됨.
- HSV 색 공간에서는 H 값만을 이용하여 색 종류를 구분할 수 있었지만, YCrCb 색 공간에서는 Cr과 Cb를 함께 조합하여 색을 구분할 수 있음.

그레이스케일영상 변환 예제

```
def color_grayscale():
    src = cv.imread(IMGpath, cv.IMREAD_COLOR)

    dst = cv.cvtColor(src, cv.COLOR_BGR2GRAY)
```

- 그레이스케일영상을 컬러로 읽고, 그레이스케일영상으로 변환

색상 채널 나누기

- imread()함수로부터 생성된 컬러 영상은 하나의 픽셀이 3가지의 색상정보를 갖고 있음.
- 컬러영상을 다루다 보면 한가지의 색상 성분을 이용하거나 HSV색공간으로 변환한 후 H성분만 이용하는 경우가 종종 발생
- 이경우에는 3채널을 1채널로 분리해 다루는것이 효율적

색상 채널 나누기 예제

```
def color_split():
    src = cv.imread(IMG_PATH, cv.IMREAD_COLOR)

    bgr_planes = cv.split(src)

    bgr_planes[0]
    # B
    bgr_planes[1]
    # G
    bgr_planes[2]
    # R
```

- split()함수를 통해 색상 채널을 분리 할 수 있음.
- 추가적으로 merge()함수를 통해 다시 합치는것도 가능.

컬러 영상 처리 기법

- openCV에서는 equalizeHist()함수를 통해 히스토그램 평활화를 수행할 수 있지만, 이는 그레이스케일 영상만 입력으로 받을 수 있음.
- 각각 스플릿한 채널로 히스토그램 평활화 함수를 수행하고, 이후 다시 합칠 수 있지만 각 색 성분이 연계되어 있어서 독립적 수행시 결과가 달라질 수 있다는 단점 존재.
- 컬러영상에서 색감은 변경하지 않고, 명암비만 높이려면 영상의 밝기 정보만을 사용해야 함.
- 보통 컬러영상에 대해 히스토그램 평활화를 수행하려면 입력영상에 밝기정보와 색상정보를 분리함.
- 이후 밝기 정보에 대해서 히스토그램 평활화를 수행.
- 예) YCrCb에서 Y성분만 히스토그램 평활화 수행. 이후 다시 합치기.

컬러 히스토그램 평활화

```
src = cv.imread(IMGPATH, cv.IMREAD_COLOR)

src_ycrcb = cv.cvtColor(src, cv.COLOR_BGR2YCrCb)

# 컬러 영상을 YCrCb로 변환

ycrcb_planes = cv.split(src_ycrcb)

# Y, Cr, Cb성분으로 추출

ycrcb_planes[0] = cv.equalizeHist(ycrcb_planes[0])

# Y성분만 히스토그램 평활화 수행

dst_ycrcb = cv.merge(ycrcb_planes)

# 다시 각 성분을 합치기

dst = cv.cvtColor(dst_ycrcb, cv.COLOR_YCrCb2BGR)

# 이후 다시 BGR로 변환
```

- 색감은 유지한채 명암비만 높일 수 있음.

색상 범위 지정에 의한 영상 분할

- 컬러영상을 다루는 응용에서 자주 요구되는 기법은 특정 색상 영역을 추출하는 작업.
- 컬러 영상에서 대표 색상 영역을 구분할 때에는 RGB색공간 보다 HSV등 색상 정보가 따로 설정되어 있는 색 공간을 사용하는 것이 유리
- inrange()함수를 통해 원소 값이 특정 범위안에 있는지 확인 할 수 있음 -> inrage()는 픽셀 값이 지정한 밝기 또는 색상 범위에 포함되어 있으면 흰색이다. 즉 True이면 1 -> 그렇지 않으면 검은색으로 채워진 마스크 영상을 반환 -> 만약 그레이스케일 영상을 입력으로 사용할 경우, 특정 밝기 값 범위에 있는 픽셀 영역을 추출할 수 있음.

색상 범위 지정에 의한 영역 분할 예제

```
def on_hue_changed(_=None):
    lower_hue = cv.getTrackbarPos("Lower Hue", 'mask')
    upper_hue = cv.getTrackbarPos("Upper Hue", 'mask')

    lowerb = (lower_hue, 100, 0)
    upperrb = (upper_hue, 255, 255)

    mask = cv.inRange(src_hsv, lowerb, upperb)

    cv.imshow('mask',mask)

def main():
    global src_hsv

    src = cv.imread(IMGPATH, cv.IMREAD_COLOR)

    src_hsv = cv.cvtColor(src, cv.COLOR_BGR2HSV)

    cv.imshow(src)

    cv.nameWindow('mask')
    cv.createTrackbar('Lower Hue', 'mask', 40, 179, on_hue_changed)
    cv.createTrackbar('Upper Hue', 'mask', 80, 179, on_hue_changed)
    on_hue_changed(0)
```

히스토그램 역투영

- HSV 색 공간에서 H값을 이용하면 간단하게 특정 색상을 골라낼 수 있어서 편리함. -> 그러나 빨간색, 노란색, 녹색, 파란색 등 원색을 찾을때 효과적.
- 만약 입력에서 찾고자 하는 객체의 기준영상을 미리 갖고 있다면 컬러 히스토그램 정보를 이용하여 비슷한 색상 영역을 찾을 수 있음.
- 기준 영상으로부터 찾고자 하는 객체의 컬러 히스토그램을 미리 구하고 주어진 입력 영상에서 해당 히스토그램에 부합하는 영역을 찾는 방식
- 이처럼 주어진 히스토그램 모델과 일치하는 픽셀을 찾아내는 기법을 히스토그램 역투영이라고 함.

히스토그램 역투영 예제

- calcBackProject()로 역투영 가능.

```
ref = cv.imread(cv.IMREAD_COLOR)
mask = cv.imread(cv.IMREAD_GRAYSCALE)
ref_ycrcb = cv.imread(cv.COLOR_BGR2YCrCb)
```

```
channels = [1,2] # cr, cb만 사용

cr_bins = 128
cb_bins = 128

histSize = [cr_bins, cb_bins]

cr_range = [0,256]
cb_range = [0,256]

ranges = cr_range + cb_range

hist = cv.calcHist([ref_ycrcb], channels, mask, histSize, ranges)
# 배열로 넣는것을 주의하자.

src = cv.imread()
src_ycrcb = cv.cvtColor(src, cv.COLOR_BGR2YCrCb)

backproj = cv.calcBackProject([src_ycrcb], channels, hist, ranges, 1)
```

- mask영상은 미리 확보해두어야 함.
- 해당 마스크 영역에 CrCb의 히스토그램 값을 구하고 역투영 함.
- 즉 원본이 있다면, 찾고자 하는 객체의 색상 히스토그램을 구하고, 해당 위치를 찾아서 값을 써준다.