과목명 : 컴퓨터 비전

과제명 : 화소처리 실습예제

학 과 : 소프트웨어학과

제 출 일 자 : 2022.04.30

학 번 : 1726088

이 름 : 최민수

[문제] 6.2.2 영상의 화소값 확인

[코드]

```
import cv2
import numpy as np
image=cv2.imread("C:/Users/chlasltn/Desktop/IMAGE/Imagess/pixel.jpg",cv2.IMREAD_GRAYSCALE)

(x,y),(w,h) =(180,37),(15,10) #좌표설정
roi_img=image[y:y+h,x:x+w]      #행렬접근
print("roi_img =")

for row in roi_img:  #원소 순회 방식 출력
    for p in row:
        print("%4d"% p, end="")

print()
cv2.rectangle(image,(x,y,w,h),255,1) #관심 영역 사각형 표시
cv2.imshow("image",image)
cv2.waitKey(0)
```

[실행결과]


```
roi_img =
  56  51  59  66  84 104 154 206 220 208 203 207 205 204 204  75  57  53  53
 72  71 100 152 195 214 212 201 209 207 205  88  76  65  53  51  60  73  96 143
200 219 200 206 204 202  91  92  80  63  53  59  59  61  89 144 195 222 205 20
0 205  89  94  90  82  63  54  51  56  65  92 149 203 223 209 196  89  91  90
 89  84  64  54  55  51  56  94 140 208 223 203  91  86  84  85  97  86  72  59
 50  53  66  81 148 211 216  92  86  85  88  92  95  88  70  55  53  59  64  89
155 211  88  85  86  90  87  87  89  86  72  56  50  53  59  88 175  87  85  8
6  88  87  84  86  90  86  70  53  44  51  56 111
```

[문제] 6.2.3 행렬 가감 연산 통한 영상 밝기 변경

[소스코드]

```
import cv2
import numpy as np

image=cv2.imread("C:/Users/chlasltn/Desktop/imagesss/bright.jpg",cv2.IMREAD_GRAYSCALE)

dst1=cv2.add(image,100) #OpenCV Saturation 방식 사용
dst2=cv2.subtract(image,100)
dst3=image+100  #Numpy.ndarray Modulo 방식 사용
dst4=image-100

cv2.imshow("Original Image",image)
cv2.imshow("dst1 bright :OpenCV",dst1)
cv2.imshow("dst2 dark :OpenCV",dst2)
cv2.imshow("dst3 bright : numpy",dst3)
cv2.imshow("dst3 dark : numpy",dst3)
cv2.waitKey(0)
```

[실행결과]

[문제] 6.1.1 행렬 원소 접근 방법

[소스코드]

```
import cv2
import numpy as np

def mat_access1(mat):  #원소 직접 접근 방식
    for i in range(mat.shape[0]):
        for j in range(mat.shape[1]):
            k=mat[i,j]
            mat[i,j]=k*2

def mat_access2(mat):  #item(), itemset() 함수 방식 사용
    for i in range(mat.shape[0]):
        for j in range(mat.shape[1]):
            k=mat.item(i,j)
            mat.itemset((i,j),k*2)

mat1=np.arange(10).reshape(2,5)   # 0~10 사이 정수형 난수 생성
mat2=np.arange(10).reshape(2,5)

print("원소 처리 전: \n%s\n" % mat1)
mat_access1(mat1)
print("원소 처리 후: \n%s\n" % mat1)

print("원소 처리 전: \n%s\n" % mat2)
mat_access1(mat1)
print("원소 처리 후: \n%s\n" % mat2)
```

[실행결과]

```
원소 처리 전:
[[0 1 2 3 4]
 [5 6 7 8 9]]

원소 처리 후:
[[ 0  2  4  6  8]
 [10 12 14 16 18]]

원소 처리 전:
[[0 1 2 3 4]
 [5 6 7 8 9]]

원소 처리 후:
[[0 1 2 3 4]
 [5 6 7 8 9]]
```

[문제] mat:ptr()을 통한 행렬 원소 접근

[소스코드]

```
import cv2
import numpy as np
import time

def pixel_access1(image): # 화소 직접 접근
    image1=np.zeros(image.shape[:2],image.dtype)
    for i in range(image.shape[0]):
        for j in range(image.shape[1]):
            pixel=image[i,j]
            image1[i,j]=255-pixel
    return image1

def pixel_access2(image):  # item(), itemset() 함수 방식 사용
    image2=np.zeros(image.shape[:2],image.dtype)
    for i in range(image.shape[0]):
        for j in range(image.shape[1]):
            pixel=image.item(i,j)
            image2.itemset((i,j),255-pixel)
    return image2

def pixel_access3(image):  #  LookUp Table() 사용
    lut=[255- i for i in range(256)]
    lut=np.array(lut,np.uint8)
    image3=lut[image]
    return image3

def pixel_access4(image):  # OpenCV 함수 사용
    image4=cv2.subtract(255,image)
    return image4

def pixel_access5(image):  # Numpy.ndarray 함수 사용
    image5=255-image
    return image5
image=cv2.imread("C:/Users/chlasltn/Desktop/imagesss/bright.jpg",cv2.IMREAD_GRAYSCALE)

def time_check(func,msg):  # 수행시간 체크 함수
    start_time=time.perf_counter()
    ret_img=func(image)
    elapsed=(time.perf_counter()-start_time)*1000
    print(msg,"수행시간 : %0.2f ms"%elapsed)
    return ret_img
image1=time_check(pixel_access1, "[방법 1] 직접 접근 방식")
image2=time_check(pixel_access2, "[방법 2] item()함수 방식")
image3=time_check(pixel_access3, "[방법 3] LUT 방식")
image4=time_check(pixel_access4, "[방법 4] OpenCV 방식")
image5=time_check(pixel_access5, "[방법 5] ndarray 방식")
```

[실행결과]

```
[방법1] 직접 접근 방식 수행시간 : 508.84 ms
[방법2] item()함수 방식 수행시간 : 49.10 ms
[방법3] LUT 방식 수행시간 : 0.68 ms
[방법4] OpenCV 방식 수행시간 : 0.27 ms
[방법5] ndarray 방식 수행시간 : 0.17 ms
```

[문제] 6.2.4 행렬 합과 곱 연산을 통한 영상 합성

[소스코드]

```
import cv2
import numpy as np
import time

image1=cv2.imread("C:/Users/chlasltn/Desktop/imagesss/add1.jpg",cv2.IMREAD_GRAYSCALE)
image2=cv2.imread("C:/Users/chlasltn/Desktop/imagesss/add2.jpg",cv2.IMREAD_GRAYSCALE)

alpha,beta=0.6,0.7  # 곱셈 비율

add_img1=cv2.add(image1,image2) # 두 영상 더하기
add_img2=cv2.add(image1*alpha,image2*beta) # 두 영상 비율에 따른 더하기
add_img2=np.clip(add_img2,0,255).astype('uint8') # Saturation 처리
add_img3=cv2.addWeighted(image1,alpha,image2,beta,0) # 두 영상 비율에 따른 더하기

titles=['image1','image2','add_img1','add_img2','add_img3']

for t in titles: cv2.imshow(t,eval(t))

cv2.waitKey(0)
```

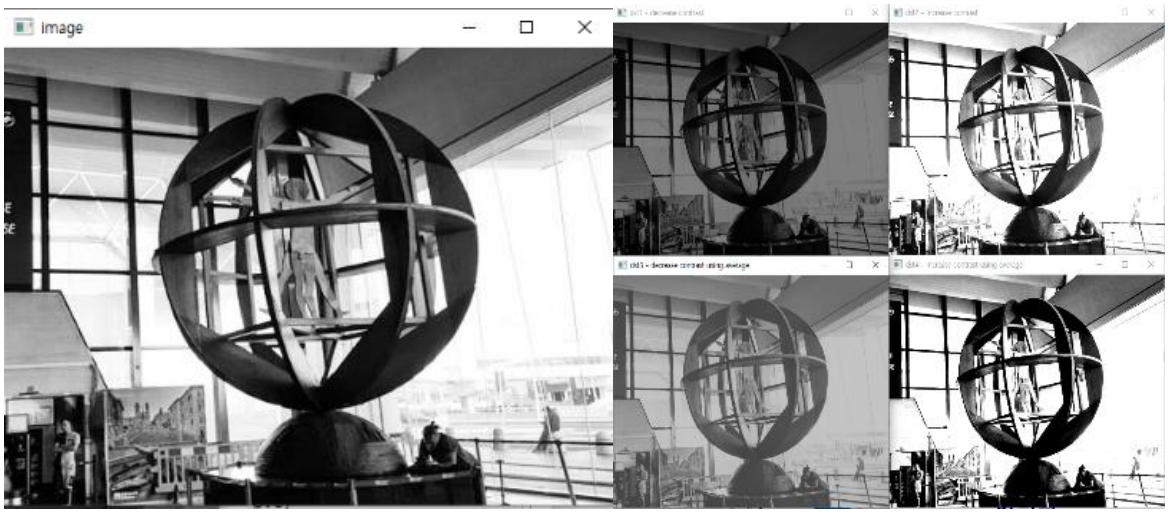
[실행결과]

[문제] 6.2.5 영상 대비 변경

[소스코드]

```
import cv2
import numpy as np
import time

image=cv2.imread("C:/Users/chlasltn/Desktop/imagesss/contrast.jpg",cv2.IMREAD_GRAYSCALE)

noimage=np.zeros(image.shape[:2],image.dtype) # 더미 영상 생성

avg=cv2.mean(image)[0]/2.0 # 화소를 평균으로 맞춤
dst1=cv2.scaleAdd(image,0.5,noimage) # 명암 대비 감소 scaleAdd 함수 사용
dst2=cv2.scaleAdd(image,2.0,noimage) # 명암 대비 증가
dst3=cv2.addWeighted(image,0.5,noimage,0,avg) # 명암 대비 감소 addWeighted 함수 사용
dst4=cv2.addWeighted(image,2.0,noimage,0,-avg) # 명암 대비 증가

cv2.imshow("image",image)
cv2.imshow("dst1 - decrease contrast",dst1)
cv2.imshow("dst2 - increase contrast",dst2)
cv2.imshow("dst3 - decrease contrast using average",dst3)
cv2.imshow("dst4 - increase contrast using average",dst4)
cv2.waitKey(0)
```

[실행결과]



[문제] 6.3.1 영상 히스토그램 계산

[소스코드]

```
import numpy as np, cv2

def calc_histo(image, hsize, ranges=[0, 256]): # 사용자 정의 함수
    hist = np.zeros((hsize, 1), np.float32)  # 히스토그램 행렬
    gap = ranges[1] / hsize  # 간격

    for i in range(image.shape[0]):  # 2 차원 행렬
        for j in range(image.shape[1]):
            idx = int(image.item(i,j) / gap)
            hist[idx] += 1

    return hist
image = cv2.imread("C:/Users/chlasltn/Desktop/IMAGE/Imagess/pixel.jpg", cv2.IMREAD_GRAYSCAL
E)
# 영상 읽기

hsize, ranges = [32], [0, 256] # 히스토그램 간격수
gap = ranges[1]/hsize[0] # 계급 간격
ranges_gap  = np.arange(0, ranges[1]+1, gap) # 넘파이 계급간격

hist1 = calc_histo(image, hsize[0], ranges)   #user 함수 사용
hist2 = cv2.calcHist([image], [0], None, hsize, ranges)  # OpenCV 함수 사용
hist3, bins = np.histogram(image, ranges_gap ) # numpy 모듈 함수

print("User 함수: \n", hist1.flatten())
print("OpenCV 함수: \n", hist2.flatten())
print("numpy 함수: \n", hist3)

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

[실행결과]

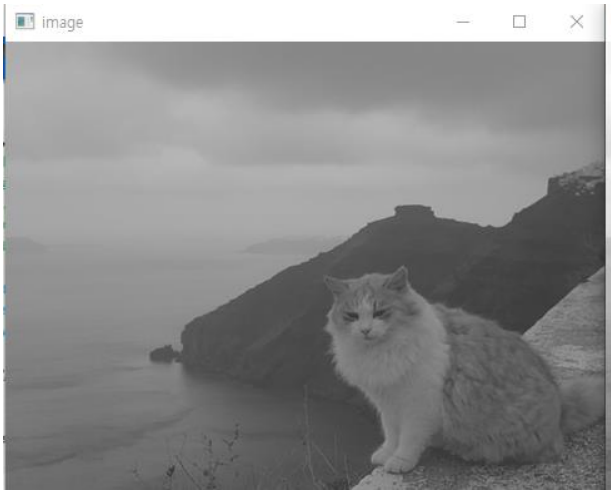

```
User 함수:
 [  97.  247.  563. 1001. 1401. 1575. 1724. 1951. 2853. 3939. 3250. 2549.
 2467. 2507. 2402. 2418. 2727. 3203. 3410. 3161. 2985. 2590. 3384. 4312.
 4764. 3489. 2802. 2238. 1127.  628.  199.   37.]
OpenCV 함수:
 [  97.  247.  563. 1001. 1401. 1575. 1724. 1951. 2853. 3939. 3250. 2549.
 2467. 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]
```

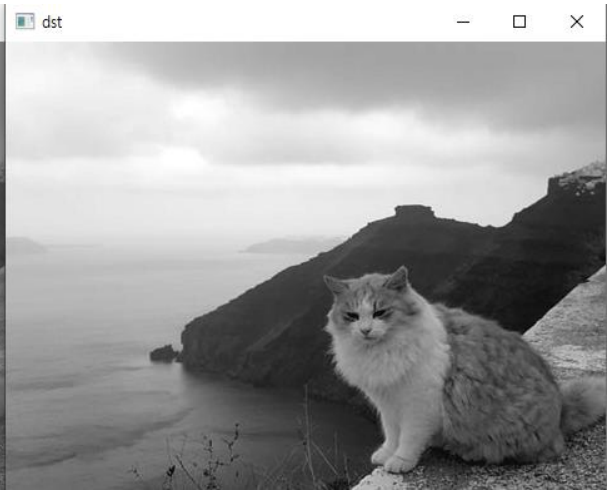
[문제] 6.3.5 히스토그램 스트레칭

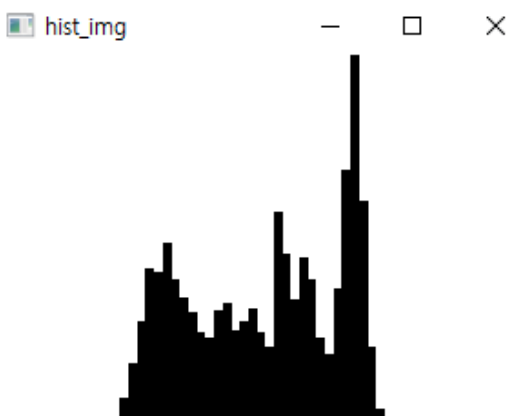
[소스코드]

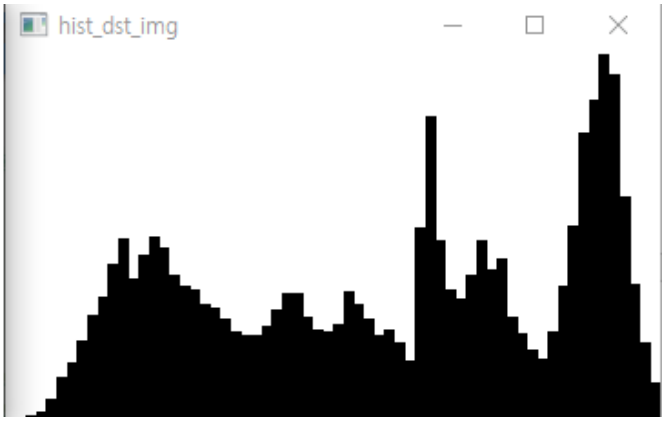
```
import numpy as np, cv2


def draw_histo(hist, shape=(200,256)): # Common.histogram 오류로 함수 생성
    hist_img = np.full(shape, 255, np.uint8) # 255로 채우기
    cv2.normalize(hist, hist, 0, shape[0], cv2.NORM_MINMAX) # 정규화
    gap = hist_img.shape[1]/hist.shape[0]


    for i,h in enumerate(hist):
        x= int(round(i*gap))    # x 좌표
        w= int(round(gap)) # y 좌표
        cv2.rectangle(hist_img, (x, 0, w, int(h)), 0, cv2.FILLED)

    return cv2.flip(hist_img, 0)

def search_value_idx(hist, bias=0):    # 값 있는 첫 계급 검색 함수

    for i in range(hist.shape[0]):
        idx = np.abs(bias - i)     # 검색 위치
        if hist[idx] > 0: return idx   #위치 반환

    return -1
image = cv2.imread("C:/Users/chlasltn/Desktop/imagesss/hist_stretch.jpg", cv2.IMREAD_GRAYSC
ALE)
# 영상 읽기

bsize, ranges = [64], [0,256] #계급 개수 , 화소 범위
hist = cv2.calcHist([image],[0],None,bsize,ranges)
bin_width = ranges[1]/bsize[0]  # 너비 설정

low = search_value_idx(hist, 0) * bin_width    # 최저 화소값
high = search_value_idx(hist, bsize[0]-1) * bin_width # 최고 화소값

idx = np.arange(0,256)     # 룩업 인덱스 생성
idx = (idx - low)/(high- low) * 255
idx[0:int(low)] = 0
idx[int(high+1):] = 255

dst = cv2.LUT(image,idx.astype('uint8'))     #룩업 테이블 사용

hist_dst = cv2.calcHist([dst],[0],None, bsize, ranges) # 히스토그램 계산
hist_img = draw_histo(hist, (200,300))           # 원본 영상 히스토그램
hist_dst_img = draw_histo(hist_dst, (200,360))  # 결과 영상 히스토그램

print("high value = ",high)
print("low value = ",low)



cv2.imshow("image",image); cv2.imshow("hist_img",hist_img)
cv2.imshow("dst",dst);  cv2.imshow("hist_dst_img",hist_dst_img)
cv2.waitKey(0)
cv2.destroyAllWindows();
```

[결과]







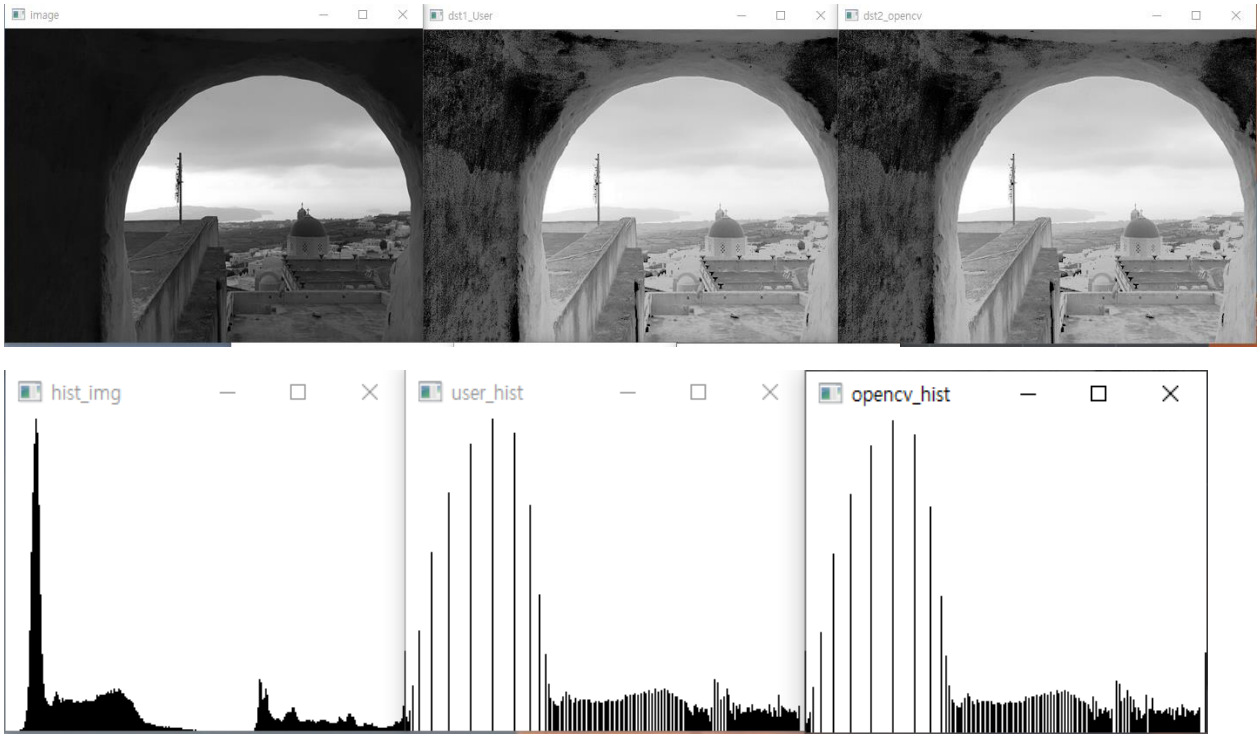


[문제] 6.3.6 히스토그램 평활화

[소스코드]

```
import numpy as np, cv2

def draw_histo(hist, shape=(200,256)): # Common.histogram 오류로 함수 생성
    hist_img = np.full(shape, 255, np.uint8) # 255로 채우기
    cv2.normalize(hist, hist, 0, shape[0], cv2.NORM_MINMAX) # 정규화
    gap = hist_img.shape[1]/hist.shape[0]

    for i,h in enumerate(hist):
        x= int(round(i*gap))    # x 좌표
        w= int(round(gap)) # y 좌표
        cv2.rectangle(hist_img, (x, 0, w, int(h)), 0, cv2.FILLED)

    return cv2.flip(hist_img, 0)

image = cv2.imread("C:/Users/chlasltn/Desktop/imagesss/equalize.jpg", cv2.IMREAD_GRAYSCALE)
# 영상 읽기

bins,ranges=[256],[0,256]
hist=cv2.calcHist([image],[0],None,bins,ranges)

accum_hist=np.zeros(hist.shape[:2],np.float32) # 히스토그램 누적합 계산
accum_hist[0]=hist[0]

for i in range(1,hist.shape[0]):
    accum_hist[i]=accum_hist[i-1]+hist[i]


accum_hist=(accum_hist/sum(hist))*255 # 누적 합 계산

dst1=[[accum_hist[val] for val in row] for row in image] # 화소 값 계산
dst1=np.array(dst1,np.uint8)
dst2=cv2.equalizeHist(image) # OpenCV 히스토그램 평활화

hist1=cv2.calcHist([dst1],[0],None,bins,ranges) # 히스토그램 계산
hist2=cv2.calcHist([dst2],[0],None,bins,ranges)

hist_img=draw_histo(hist)
hist_img1=draw_histo(hist1)
hist_img2=draw_histo(hist2)
cv2.imshow("image",image); cv2.imshow("hist_img",hist_img)
cv2.imshow("dst1_User",dst1); cv2.imshow("user_hist",hist_img1)
cv2.imshow("dst2_opencv",dst2); cv2.imshow("opencv_hist",hist_img2)
cv2.waitKey(0)
```

[결과]

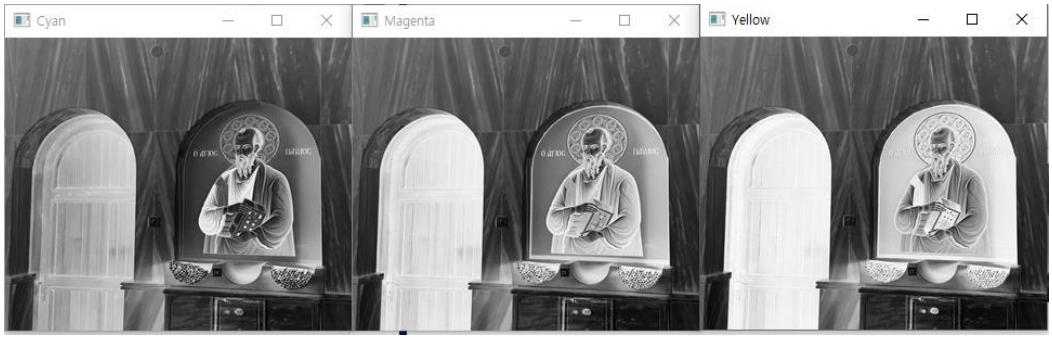


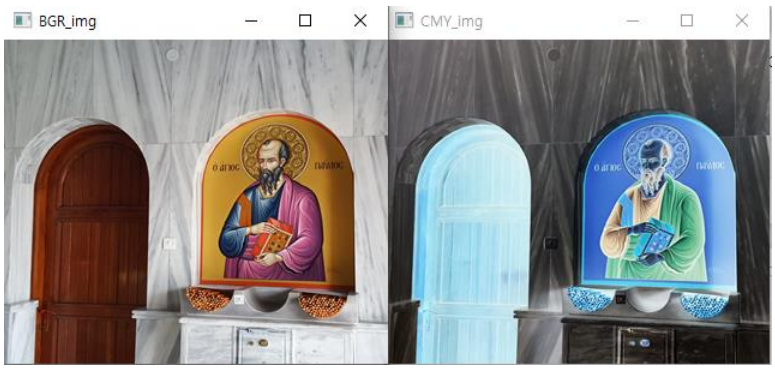
[문제] 6.4.1 컬러 공간 변환

[소스코드]

```
import numpy as np,cv2

BGR_img=cv2.imread("C:/Users/chlasltn/Desktop/imagesss/color_model.jpg",cv2.IMREAD_COLOR)
# IMREAD_COLOR 을 사용해 컬러 이미지 읽기

white=np.array([255,255,255],np.uint8)
CMY_img=white - BGR_img

Yellow,Magenta,Cyan =cv2.split(CMY_img) # 채널분리

titles=['BGR_IMG','CMY_img','Yellow','Magenta','Cyan']
for t in titles:cv2.imshow(t,eval(t
cv2.waitKey(0)
```

[실행결과]



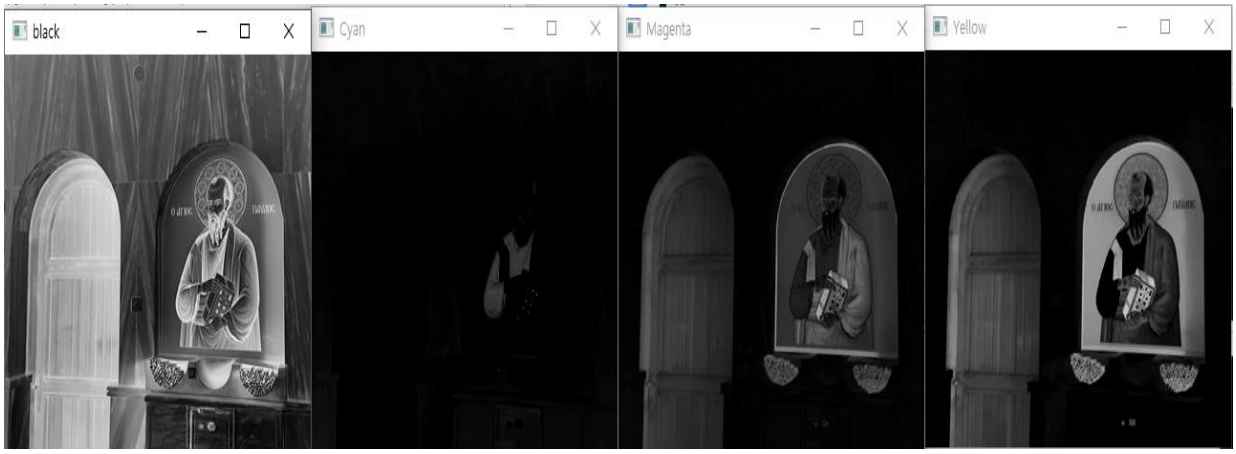


[문제] 6.4.2 컬러 공간 변환

[소스코드]

```
import numpy as np,cv2

BGR_img=cv2.imread("C:/Users/chlasltn/Desktop/imagesss/color_model.jpg",cv2.IMREAD_COLOR)
# 컬러 영상 읽기

white=np.array([255,255,255],np.uint8)

CMY_img=white - BGR_img
CMY=cv2.split(CMY_img) # 채널 분리

black=cv2.min(CMY[0],cv2.min(CMY[1],CMY[2])) # 원소 간의 최솟값 저장

Yellow,Magenta,Cyan=CMY – black # 3 개 행렬 화소값 차분

titles=['black','Yellow','Magenta','Cyan']
[cv2.imshow(t,eval(t)) for t in titles]
cv2.waitKey(0)
```

[실행결과]

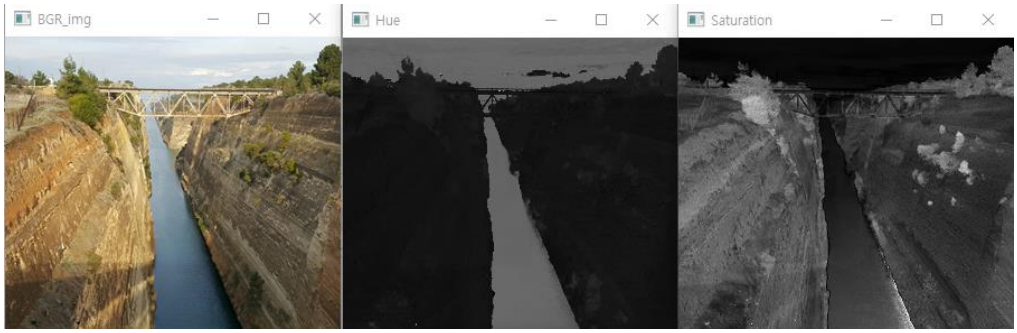


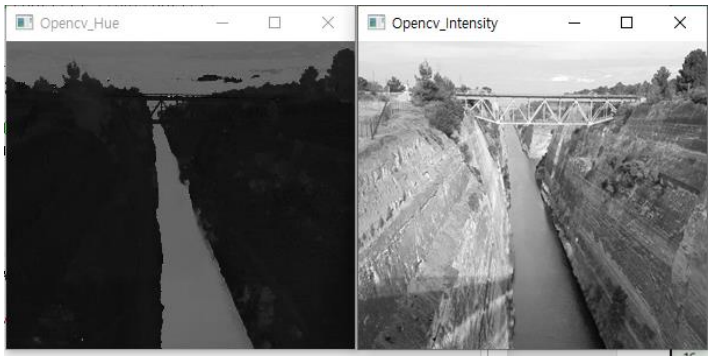
[문제] 6.4.3 컬러 공간 변환

[소스코드]

```
import numpy as np,time,cv2,math;


def calc_hsi(bgr): # HIS 를 계산하는 함수 정의
    B,G,R=float(bgr[0]),float(bgr[1]),float(bgr[2]) # float 형으로 형 변환
    bgr_sum=(R+G+B)
    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

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


def bgr2hsi(image): # BGR 컬러에서 HSI 함수로 변환
    hsv=[[calc_hsi(pixel) for pixel in row] for row in image]
    return cv2.convertScaleAbs(np.array(hsv))


BGR_img=cv2.imread("C:/Users/chlasltn/Desktop/imagesss/color_space.jpg",cv2.IMREAD_COLOR)

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

titles=['BGR_img','Hue','Saturation','Intensity']
[cv2.imshow(t,eval(t)) for t in titles]
[cv2.imshow('Opencv_'+t,eval(t+'2')) for t in titles[1:]]
cv2.waitKey(0)
```

[실행결과]





[문제] 6.4.4 다양한 컬러 공간 변환

[소스코드]

```
import cv2

BGR_img=cv2.imread("C:/Users/chlasltn/Desktop/imagesss/color_space.jpg",cv2.IMREAD_COLOR)

Gray_img=cv2.cvtColor(BGR_img,cv2.COLOR_BGR2GRAY) # 명암도 영상 변환
YCC_img=cv2.cvtColor(BGR_img,cv2.COLOR_BGR2YCrCb)
YUV_img=cv2.cvtColor(BGR_img,cv2.COLOR_BGR2YUV)
LAB_img=cv2.cvtColor(BGR_img,cv2.COLOR_BGR2LAB)

YCC_ch=cv2.split(YCC_img)
YUV_ch=cv2.split(YUV_img)
LAB_ch=cv2.split(LAB_img)

cv2.imshow("BGR_img",BGR_img)
cv2.imshow("Gray_img",Gray_img)

sp1,sp2,sp3=['Y','Cr','Cb'],['Y','U','V'],['L','A','B']

for i in range(len(sp1)):
    cv2.imshow("ycc_img[%d]-%s" %(i,sp1[i]),YCC_ch[i])
    cv2.imshow("yuv_img[%d]-%s" %(i,sp2[i]),YUV_ch[i])
    cv2.imshow("lab_img[%d]-%s" %(i,sp3[i]),LAB_ch[i])

cv2.waitKey(0)
```

[실행결과]

[문제] 6.4.5 Hue채널을 이용한 객체 검출

[소스코드]

```
import numpy as np,cv2

def onThreshold(value):
    th[0]=cv2.getTrackbarPos("hue_th1","result")
    th[1]=cv2.getTrackbarPos("hue_th2","result")

    _,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("C:/Users/chlasltn/Desktop/imagesss/color_space.jpg",cv2.IMREAD_COLOR)

HSV_img=cv2.cvtColor(BGR_img,cv2.COLOR_BGR2HSV)
hue=np.copy(HSV_img[:,:,0]) # 색상 채널 복사
th=[50,100] # 트랙바 범위

cv2.namedWindow("result")
cv2.createTrackbar("hue_th1","result",th[0],255,onThreshold)
cv2.createTrackbar("hue_th2","result",th[0],255,onThreshold)
cv2.waitKey(0)
```

[실행결과]