시험 대비 - 5장 연습문제

1번 문제

💡 OpenCV의 채널 처리 함수에 대해서 아는 대로 기술하시오

```
1. 채널 분리
cv2.split(): 다채널 배열을 여러 개의 단일 채널 배열로 분리한다.

2. 채널 병합
cv2.merget(): 여러 개의 단일 채널 배열을 다채널 배열로 합성한다.
```

2번 문제

💡 OpenCV의 사칙 연산을 수행하는 함수와 연산의 수행 방법에 대해서 기술하시오.

1. 덧셈 (Addition):
   - `cv2.add()`: 두 이미지나 이미지와 스칼라 사이의 픽셀별 덧셈을 수행합니다. 두 이미지의 크기가 같아야 합니다. 픽셀별로 더한 결과가 반환됩니다.
2. 뺄셈 (Subtraction):
   - `cv2.subtract()`: 두 이미지나 이미지와 스칼라 사이의 픽셀별 뺄셈을 수행합니다. 두 이미지의 크기가 같아야 합니다. 픽셀별로 뺀 결과가 반환됩니다.
3. 곱셈 (Multiplication):
   - `cv2.multiply()`: 두 이미지나 이미지와 스칼라 사이의 픽셀별 곱셈을 수행합니다. 두 이미지의 크기가 같아야 합니다. 픽셀별로 곱한 결과가 반환됩니다.
4. 나눗셈 (Division):
   - `cv2.divide()`: 두 이미지나 이미지와 스칼라 사이의 픽셀별 나눗셈을 수행합니다. 두 이미지의 크기가 같아야 합니다. 픽셀별로 나눈 결과가 반환됩니다.

3번 문제

💡 행렬(ndarray)을 초기화하는 방법들에 대해서 기술하고, 각 방법으로 선언하시오.

```
1. np.zeros로 초기화하기
# 3 x 3 크기에 값은 0인 넘파이 행렬 생성
zeros = np.zeros((3,3), np.uint8)

2. np.ones로 초기화기
# 3 x 3 크기에 값은 1인 넘파이 행렬 생성
ones = np.ones((3,3), np.uint8)

3. np.full로 초기화하기
# 3 x 3 크기에 모든 값이 100인 넘파이 행렬 생성
full = np.full((3,3),100,np.uint8)

4. np.random.rand로 초기화하기
# 3x3 크기의 랜덤 값으로 초기화된 행렬
random = np.random.rand(3,3)
```

4번 문제

💡 사칙 연산이나 논리 비트 연산에서 마스킹을 사용할 수 있다. 마스크 행령에 대한 의미와 사용법에 대해서 설명하시오.

mask는 연산 마스크라고 하는 8비트의 단일 채널 배열로서, 입력 배열의 원소 중에서 mask 배열의 원소가 0이 아닌 위치만 연산대상으로한다. 따라서 원하는 위치에 대해서 연산을 수행하고자 할 때, mask 배열의 원하는 위치에 0이 아닌 값을 지정하면 된다.

5번 문제

💡 cv2.reduce()함수에 대해서 설명하고, 특히 축소 시에 사용하는 연산 옵션에 대해서 상세히 설명하시오.

cv2.reduce()함수는 행렬을 열방향/행방향으로 옵션에 따라서 축소한다.

옵션에는 cv2.REDUCE_SUM, cv2.REDUCE_AVG, cv2.REDUCE_MAX, cv2.REDUCE_MIN가 있고 각각 모든 행(열)의 합, 평균, 최댓값, 최솟값을 구한다.

6번 문제

💡 다음 예시 코드의 컴파일 에러와 수정하기

```
import numpy as np, cv2

m1 = [1,2,3,1,2,3]
m2 = [3,3,4,2,2,3]
m3 = m1 + m2
m4 = m1 - m2

print("[m1] %s" %m1)
print("[m2] %s" %m2)
print("[m3] %s" %m3)
print("[m4] %s" %m4)
```

list형식인 m1과 m2 는 +, - 연산자로 계산할 수 없다.
m3와 m4를 아래와 같이 수정한다.

```
m3 = np.add(m1,m2)
m4 = np.subtract(m1,m2)
```

7번 문제

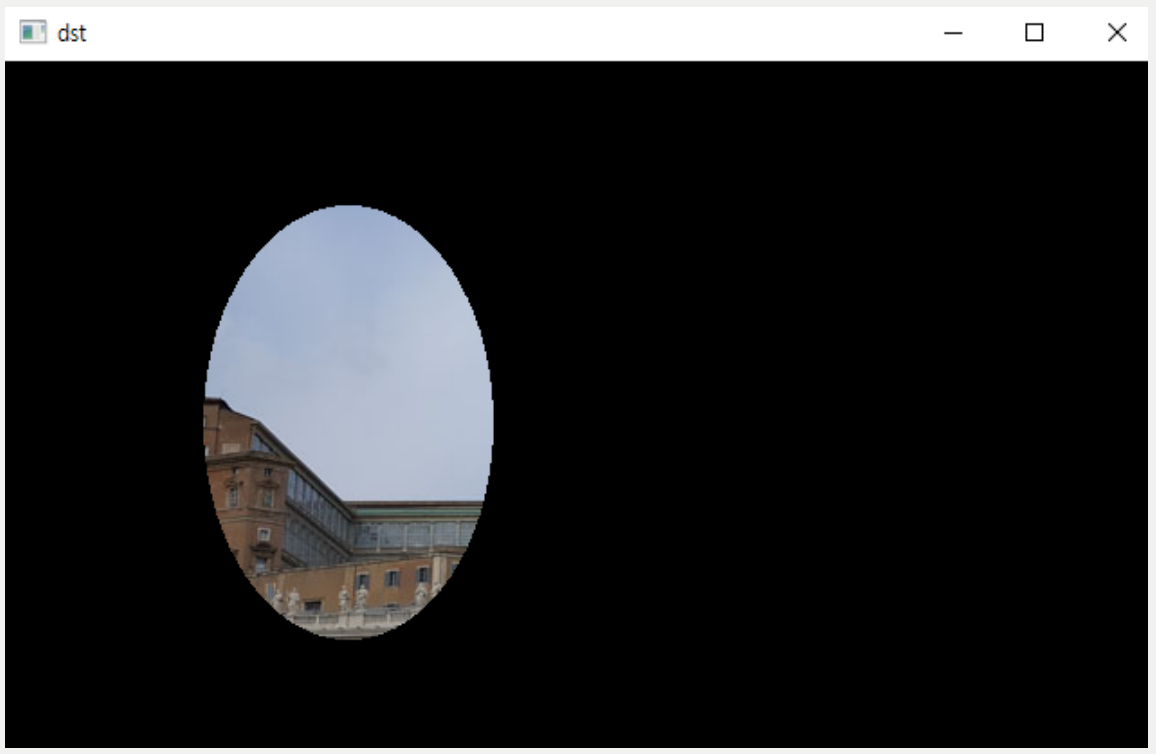
💡 컬러 영상 파일을 입력받아서 RGB 3개의 채널로 분리하고 각 채널을 컬러영상으로 윈도우에 표시하기

```
import numpy as np, cv2

logo = cv2.imread("logo.jpg", cv2.IMREAD_COLOR)
if logo is None: raise Exception("영상 파일 읽기 오류")

blue, green, red = cv2.split(logo)

zeros = np.zeros((blue.shape), np.uint8)
blueimg = cv2.merge([blue, zeros, zeros])
greenimg = cv2.merge([zeros, green, zeros])
redimg = cv2.merge([zeros, zeros, red])
```

```
cv2.imshow("blue", blueimg)
cv2.imshow("green", greenimg)
cv2.imshow("red", redimg)
cv2.waitKey(0)
```

8번 문제

💡 이미지의 타원 영역만 보이게하기




```
import numpy as np, cv2

image = cv2.imread("bit_test.jpg", cv2.IMREAD_COLOR)

if image is None : raise Exception("영상 파일 읽기 오류")

mask = np.zeros(image.shape[:2], np.uint8)
center = (190,200)

cv2.ellipse(mask,center, (80,120),0, 0, 360, 255,-1)
# mask = mask
dst = cv2.bitwise_and(image, image, mask = mask)

# cv2.imshow("image", image)
cv2.imshow("mask" ,mask)
cv2.imshow("dst",dst)
cv2.waitKey(0)
```

9번 연습문제

💡 3행, 6열의 행렬을 생성하고, 행렬의 원소를 초기화한 후에 cv2.reduce() 함수를 이용해서 가로 방향과 세로방향으로 감축하여 평균을 구한 결과를 출력하시오.

```
import numpy as np, cv2

data = [3,4,5,6,7,8,
```

```
        9,10,11,12,13,14,
        15,16,17,18,19,20]

m = np.array(data).reshape(3,6).astype(float)
# m = np.random.randint(0,10,(3,6)).astype(float)

result1 = cv2.reduce(m, dim=0, rtype=cv2.REDUCE_AVG)
result2 = cv2.reduce(m, dim=1, rtype=cv2.REDUCE_AVG)

print("세로(열) 방향 감축: \n", result1)
print("가로(행) 방향 감축: \n", result2)
```

10번 연습문제

💡 PC카메라로 영상을 읽어서 특정 부분의 합과 평균을 구하는 프로그램을 작성하시오
1) 관심 영역은 200, 100 좌표에서 200 x 100 크기로한다.
2) cv2.mean() 함수를 사용하여 평균을 구하시오.
3) cv2.mean() 함수를 사용하지 않고 영상의 원소 순회 방법으로 평균을 구하시오.

```
import numpy as np, cv2
from Common.utils import put_string

capture = cv2.VideoCapture(0)
if not capture.isOpened(): raise Exception("카메라 연결 안됨")

while True:
    ret, frame = capture.read()
    if not ret : break

    x,y,w,h = (200,100, 200,100)
    cv2.rectangle(frame, (x,y,w,h), (0,0,255),2)
    tmp = frame[y:y+h, x:x+w]

    # x2,y2 = 400,200
    # cv2.rectangle(frame, (x,y),(x2,y2), (0,0,255),2)
    # tmp = frame[y:y+100, x:x+200]

    # 함수이용
    average1 = tuple(map(int, cv2.mean(tmp)))

    # 행렬 순회
    value = np.array([0,0,0],np.uint8)
    for row in tmp:
        for pixel in row:
            value += pixel
    average2 = (value / (w*h)).astype(int)

    put_string(frame, "average1 : " , (10,30), average1[:-1])
    put_string(frame, "average2 : " , (50,70), average2)

    if cv2.waitKey(30) >= 0 : break
    cv2.imshow("ex10", frame)
capture.release()
```

11번 문제

💡 PC 카메라로 영상을 받아들여서 다음과 같이 윈도우의 특정 영역에서 재생하시오.
1) 메인 윈도우는 400 x 300 크기로 한다.
2) 관심 영역 30, 30 좌표에서 320 x 240 크기로한다.
3) 관심 영역에 빨간색 테두리를 두른다.

왜 frame이 360, 640 이 나오는 걸까

```
import numpy as np, cv2

capture = cv2.VideoCapture(0)
if not capture.isOpened() : raise Exception("카메라 연결 안됨")

capture.set(cv2.CAP_PROP_FRAME_WIDTH, 400)
capture.set(cv2.CAP_PROP_FRAME_HEIGHT, 300)

x,y,w,h = (30,30,320,240)
# blue = np.full((360,640,3), (255,0,0),np.uint8)
mask = np.full((360,640), 0,np.uint8)
cv2.rectangle(mask, (x,y,w,h),255,-1)


while True:
    ret, frame = capture.read()
    if not ret : break
    # cv2.rectangle(frame, (x,y,w,h), (0,0,255),2) # 빨간색 그려주기
    print(frame.shape)
    inner = cv2.bitwise_and(frame,frame,mask=mask)
    # outter = cv2.bitwise_and(blue,blue, mask =mask) 핑크색 테두리생김

    if cv2.waitKey(30) >= 0:break
    cv2.imshow("ex11", inner)
    # cv2.resizeWindow("ex11",400,300)

capture.release()
```

12번 문제

💡 영상파일을 읽어서 메인 윈도우에 다음과 같이 출력하시오.
1) 메인 윈도우의 특정 부분 2곳을 관심 영역으로 지정한다.
2) 관심 영역1는 영상의 밝기를 50만큼 밝게 한다.
3) 관심 영역2는 영상의 화소 대비를 증가시킨다.

```
import numpy as np, cv2

image = cv2.imread("color.jpg", cv2.IMREAD_COLOR)
if image is None : raise Exception("영상 파일 읽기 오류")

# 밝기 증가
x,y,w,h = (50,50,100,100)
cv2.rectangle(image, (x,y,w,h), (255,0,0),2) # 파란 영역

tmp = image[y:y+h, x:x+w]
img50 = np.full(tmp.shape, (50,50,50), np.uint8)
cv2.add(tmp, img50,tmp)
# cv2.add(tmp, img50,image[y:y+h, x:x+w])

# 화소대비 증가
x,y,w,h = (300,200,100,100)
cv2.rectangle(image, (x,y,w,h),(255,0,0),2)
```

```
tmp = image[y:y+h, x:x+w]
noimage = np.full(tmp.shape, (50,50,50), np.uint8)
cv2.scaleAdd(tmp, 2.0, noimage, tmp)

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

13번 문제

💡 13 cv2.sortIdx()함수를 활용해서 다음의 조건에 부합하도록 벡터의 원소를 정렬하시오.

1) 벡터의 원소는 Rect 객체이다.
2) 벡터의 원소는 임의로 지정한다.
3) 정렬의 기준은 Rect 객체의 크기이다.
4) 오름차순으로 정렬하여 콘솔 창에 출력한다.

```
import numpy as np, cv2

def print_rects(rects):
    print("-" * 46)
    print("사각형 원소 \t\t 랜덤 사각형 정보 \t\t 크기")
    print("-" * 46)
    for i,(x,y,w,h,a) in enumerate(rects):
        print("rects[%i] = [(%3d, %3d) from (%3d, %3d)] %5d" %(i,x,y,w,h,a))
    print()

rands = np.zeros((5,5), np.uint16) # 5행 5열 행렬 생성
starts = cv2.randn(rands[:,:2], 100, 50) # 0 ~ 4행까지 시작 좌표 랜덤 생성
ends = cv2.randn(rands[:, 2:-1], 300,50) # 5 ~ 9행까지 종료 좌표 랜덤 생성

sizes = cv2.absdiff(starts, ends) # 시작좌표와 종료좌표간 차분 절댓값
areas = sizes[:,0] * sizes[:,1]
rects = rands.copy()
rects[: , 2:-1] = sizes
rects[:,-1] = areas

idx = cv2.sortIdx(areas, cv2.SORT_EVERY_COLUMN).flatten()

print_rects(rects)
print_rects(rects[idx.astype('int')])
```

14번 문제

💡 연립방정식 풀이
3a + 6b + 3c = 2
-5a + 6b + c = 10
2a -3b +5c = 28

```
import numpy as np, cv2

data = [3,6,3, -5,6,1 ,2, -3, 5]
m1 = np.array(data, np.float32).reshape(3,3)
m2 = np.array([2,10,28], np.float32)
```

```
ret, inv = cv2.invert(m1, cv2.DECOMP_LU) # 역행렬 계산
if ret:
    dst1 = inv.dot(m2) # 행렬 곱 함수
    dst2 = cv2.gemm(inv, m2,1,None ,1) # 행렬곱 함수
    ret, dst3 = cv2.solve(m1,m2,cv2.DECOMP_LU) # 연립방정식 풀이

    print("[inv] = \n%s\n" % inv)
    print("[dst1] = ", dst1.flatten())
    print("[dst2] = ", dst2.flatten())
    print("[dst3] = ", dst3.flatten())
else:
    print("역행렬이 존재하지 않습니다.")

# ret는 메서드 성공여부
```

15번 문제

💡 사각형 회전하기.
전체 변환 행렬 = 이동 변환 행렬 * 회전 변환 행렬 * 이동 변환 행렬

```
import numpy as np,cv2

pts1 = np.array([(200,50,1) , (400,50,1), (400,250,1),(200,250,1)], np.float32)

theta = 45 * np.pi / 180

m = np.array([
            [np.cos(theta), -np.sin(theta), 0],
            [np.sin(theta), np.cos(theta), 0],
            [      0,      0,      1        ] ], np.float32)

delta = (pts1[2] - pts1[0]) // 2
center = pts1[0] + delta

t1 = np.eye(3, dtype = np.float32)
t2 = np.eye(3, dtype = np.float32)


t1[:2, 2] = center[:2] # 원점으로 옮기기
t2[:2, 2] = -center[:2] # 되돌아오기


# m - 회전 행렬
m2 = t1.dot(m).dot(t2) # 이동 * 회전 * 이동
m3  = t1.dot(m)

pts2 = pts1.dot(np.transpose(m2)) # 회전할 도형 * 전체 변환 행렬
# pts2 =pts1.dot(np.transpose(t1.dot(m).dot(t2)))
# pts2 = cv2.gemm(pts1, m2, 1, None,1,flags=cv2.GEMM_2_T)

image = np.full((400,500,3), 255, np.uint8)
cv2.polylines(image, [np.int32(pts1[: , :2])], True, (0,255,0),2)
cv2.polylines(image, [np.int32(pts2[: , :2])], True, (255,0,0),3)
cv2.imshow("image", image)
cv2.waitKey(0)
```