20180285 박민준 컴퓨터비전(가) 과제2

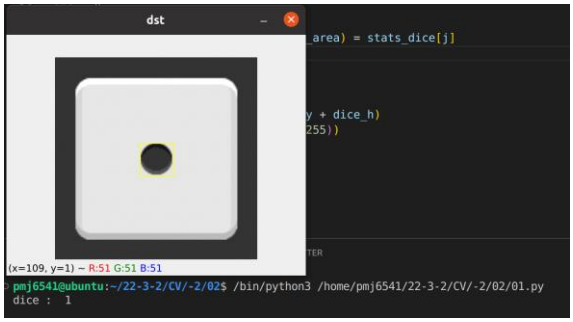
1. 사각형 내의 원의 수 세기:

한 이미지 내의 하나의 사각형이 있고, 그 안에 n개의 원이 있을 때, 원의 숫자를 console에 출력하기.

소스코드:

```python
import cv2

src = cv2.imread('case1/01.png', cv2.IMREAD_GRAYSCALE)
_, src_bin = cv2.threshold(src, 0, 80, cv2.THRESH_BINARY | cv2.THRESH_TRIANGLE)
cnt, labels, stats, centroids = cv2.connectedComponentsWithStats(src_bin)
dst = cv2.cvtColor(src, cv2.COLOR_GRAY2BGR)
dice = [0]

(x,y,w,h,area) = stats[1]
dice_src = ~src[y:y+h,x:x+w]
_, tmp = cv2.threshold(dice_src, 0, 80, cv2.THRESH_BINARY | cv2.THRESH_TRIANGLE)
dice.append(tmp)
cnt_dice, _, stats_dice, _ = cv2.connectedComponentsWithStats(dice[1])
ans = 0
for j in range(2,cnt_dice):
    (dice_x,dice_y,dice_w,dice_h,dice_area) = stats_dice[j]
    if dice_w < 30 or dice_h < 30:
        continue
    ans += 1
    pt1 = (dice_x+x, dice_y+y)
    pt2 = (dice_x+x + dice_w, dice_y+y + dice_h)
    cv2.rectangle(dst,pt1,pt2,(0,255,255))
print("dice : ",ans)

cv2.imshow('dst',dst)
cv2.waitKey()
cv2.destroyAllWindows()
```

수행결과:



설명:

ㄱ. 원의 수를 셀 사각형 사진을 읽고, 이진화를 진행한다.

ㄴ. 이진화 한 결과 내에서 connectedComponentsWithStats 함수를 통해 사각형(주사위의 면)을 인식한다.

ㄷ. 인식한 사각형 내에서 다시 connectedComponetsWithStats 함수를 통해 주사위 눈금을 인식한다.

ㄹ. 인식한 눈금이 (w <30 and h <30)에 해당하지 않는다면, 눈금의 숫자에 +1을 해준다.

ㅁ. 결과 출력

2. 사각형 내의 원의 수 세기

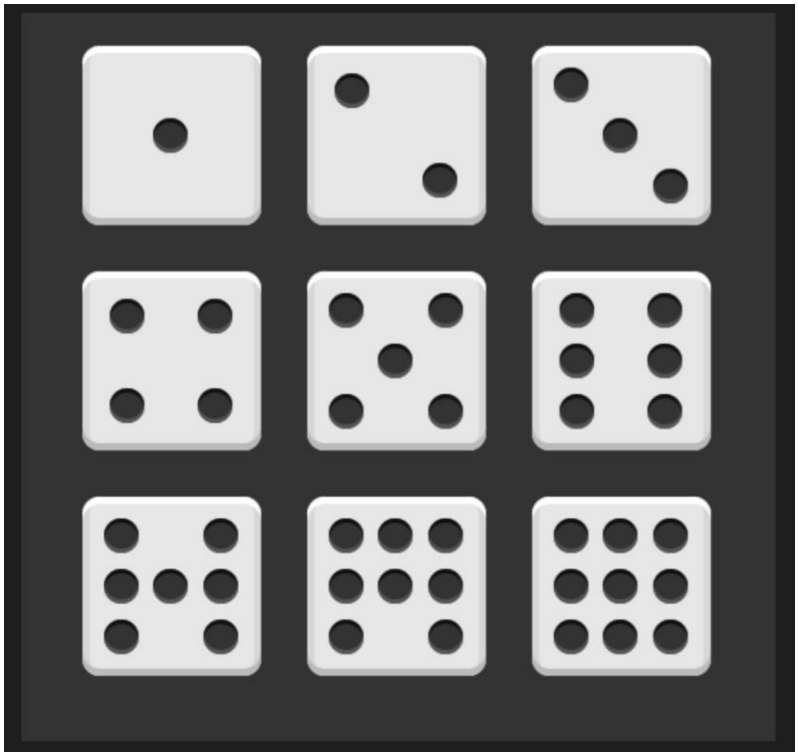
사각형 안에 각각 n개의 원이 있을 때, 각 내부에 있는 원의 숫자를 오름차순으로 출력하기

소스코드:

02.py > ...

```python
import cv2

src = cv2.imread('case2/02.png', cv2.IMREAD_GRAYSCALE)
_, src_bin = cv2.threshold(src, 0, 60, cv2.THRESH_BINARY | cv2.THRESH_TRIANGLE)
cnt, labels, stats, centroids = cv2.connectedComponentsWithStats(src_bin)
dst = cv2.cvtColor(src, cv2.COLOR_GRAY2BGR)
dice = [0]
ans = []


for i in range(1,cnt):
    (x,y,w,h,area) = stats[i]
    dice_src = ~src[y:y+h,x:x+w]
    _, tmp = cv2.threshold(dice_src, 0, 60, cv2.THRESH_BINARY | cv2.THRESH_TRIANGLE)
    dice.append(tmp)
    cnt_dice, _, stats_dice, _ = cv2.connectedComponentsWithStats(dice[i])

    ans_num = 0
    for j in range(2,cnt_dice):
        (dice_x,dice_y,dice_w,dice_h,dice_area) = stats_dice[j]
        if dice_area < 200:
            continue
        ans_num += 1
        pt1 = (dice_x+x, dice_y+y)
        pt2 = (dice_x+x + dice_w, dice_y+y + dice_h)
        cv2.rectangle(dst,pt1,pt2,(0,255,255))
    ans.append(ans_num)
    if area < 20:
        continue
ans.sort()
print('dice : ',ans)

cv2.imshow('dst',dst)
cv2.waitKey()
cv2.destroyAllWindows()
```

수행결과:



```
pmj6541@ubuntu:~/22-3-2/CV/-2/02$ /bin/python3 /home/pmj6541/22-3-2/CV/-2/02/02.py
dice :  [1, 2, 3, 4, 5, 6, 7, 8, 9]
```

설명:

ㄱ. #1과 마찬가지의 과정으로 진행.

ㄴ. connectedComponentsWithStats 함수의 반환값인 cnt를 이용해 사진 내에서 검출된 모든 사각형(주사위 면)에 대해 작업을 수행한 뒤 리스트로 반환.

ㄷ. 결과 리스트 출력.

#3, #4, #5 항목은 코드 재 사용성과 가독성을 높이기 위해
Binalization, Labeling, DiceNumber, ImageProcess 클래스들을 구현

- Binalization: 입력 받은 사진에서 주사위 면들을 검출할 수 있도록 이진화 된 사진을 반환하는 함수들을 구현한 클래스
- Labeling: 이진화 된 사진에서 주사위 면들을 검출해 리스트 형식으로 반환하는 함수들을 구현한 클래스
- DiceNumber: Labeling을 통해 검출된 주사위 면에서 눈금을 인식해 해당 주사위 면의 눈 숫자들을 리스트 형식으로 반환하는 함수들을 구현한 클래스

- ImageProcess: 더 나은 이진화를 위해 이미지를 전 처리하는 과정에서 필요한 함수들을 구현한 클래스

들어가기 앞서, #3 #4 #5의 전체적인 구조는 #1 #2와 동일하게 모두
이진화 -> 주사위 면 검출 -> 주사위 눈 검출이다.

3. 주사위 읽기(1) : 책상 위에 놓인 임의의 주사위의 눈을 읽기(단일 색)

소스코드:

<03.py>

```python
import cv2
import Binalization, Labeling, DiceNumber, ImageProcess

src = cv2.imread('case3/03_1.png', cv2.IMREAD_COLOR)
src_gray = cv2.cvtColor(src,cv2.COLOR_BGR2GRAY)

#binalization
src_bin = Binalization.binImg_03(src_gray)


#dice labeling
dice_list = Labeling.labelingImg_03_04(src_bin)


#getnumber from dice
dice_ans_list = DiceNumber.getDiceNumber_03_04(dice_list)


#sort & print
dice_ans_list.sort()
print("dice : ", dice_ans_list)
ImageProcess.showImg(src)

```

<binImg_03>

```python
def binImg_03(src) :
    alpha = 3.0
    dst = np.empty(src.shape, dtype=src.dtype)
    for y in range(src.shape[0]):
        for x in range(src.shape[1]):
            dst[y, x] = ImageProcess.saturated(src[y, x] + (src[y, x]-200) * alpha)


    _, src_bin = cv2.threshold(dst, 0, 255, cv2.THRESH_BINARY | cv2.THRESH_OTSU)

    src_bin = cv2.morphologyEx(src_bin, cv2.MORPH_OPEN, None)
    src_bin = cv2.morphologyEx(src_bin, cv2.MORPH_CLOSE, None)

    return src_bin
```

<labelingImg_03_04>

```
3   def labelingImg_03_04(src) :
4       cnt, _, stats, _ = cv2.connectedComponentsWithStats(src)
5       dice_list = []
6       for i in range(1,cnt):
7           (x, y, w, h, area) = stats[i]
8
9           if abs(w-h)>15 or area < 2500:
10              continue
11          if area < 100:
12              continue
13          tmp = src[y:y+h,x:x+w]
14          dice_list.append(tmp)
15      return dice_list
```

<labelingDice_03_04>

```
3   def labelingDice_03_04(src) :
4       cnt, _, stats, _ = cv2.connectedComponentsWithStats(src)
5       dice_list = []
6       for i in range(1,cnt):
7           (x, y, w, h, area) = stats[i]
8
9           if abs(w-h)>15 or area > 700:
10              continue
11          if area < 200:
12              continue
13          tmp = src[y:y+h,x:x+w]
14          cnt_test, _, _, _ = cv2.connectedComponentsWithStats(tmp)
15          if cnt_test == 2:
16              dice_list.append(tmp)
17      return dice_list
```

<getDiceNumber_03_04>

```
19  def getDiceNumber_03_04(dice_list) :
20      ans = []
21      tmp_dice = []
22      for i in range(len(dice_list)):
23          tmp_dice = labelingDice_03_04(~dice_list[i])
24          if len(tmp_dice)<7 and len(tmp_dice)>0:
25              ans.append(len(tmp_dice))
26      return ans
```

수행결과:

- 어두운 배경

```
pmj6541@ubuntu:~/22-3-2/CV/-2/02$ /bin/python3 /home/pmj6541/22-3-2/CV/-2/02/03.py
dice :  [1, 2, 3, 3, 4, 6]
```

- 밝은 배경

```
pmj6541@ubuntu:~/22-3-2/CV/-2/02$ /bin/python3 /home/pmj6541/22-3-2/CV/-2/02/03.py
dice :  [2, 3, 4, 6, 6]
```

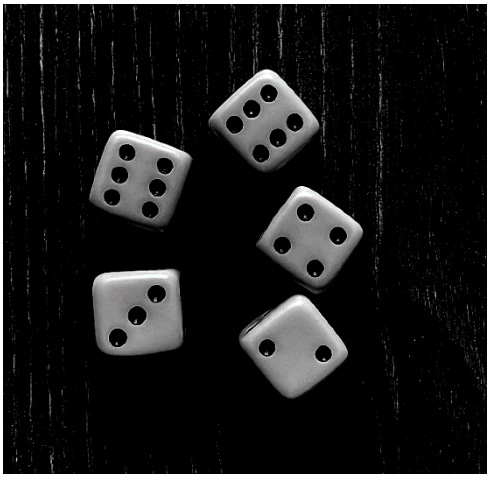
설명:

ㄱ. 이진화

밝은 배경과 어두운 배경 모두에서 흰 주사위 면이 검출이 잘 되어야 하였기 때문에 이를 고민해보았다.

어두운 배경에서는 단순이 GrayScale로 바꾼 이미지에서도 이진화가 잘 되었지만, 밝은 배경에서는 배경 또한 주사위의 흰 면이라고 인식하게 되어 이진화가 잘 되지 않았다.

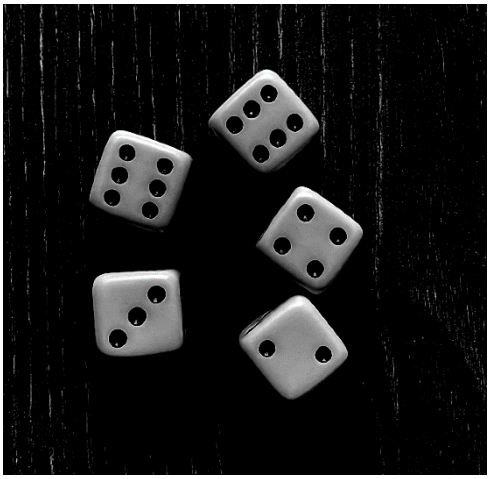
이를 해결하기 위해 지난 과제의 2번 문제였던 명암비 조절을 진행하는 과정을 추가해주었다.



```
def saturated(value):
    if value > 255:
        value = 255
    elif value < 0:
        value = 0
    return value
```

```
alpha = 3.0
dst = np.empty(src.shape, dtype=src.dtype)
for y in range(src.shape[0]):
    for x in range(src.shape[1]):
        dst[y, x] = ImageProcess.saturated(src[y, x] + (src[y, x]-200) * alpha)
```

좌측 사진은 밝은 배경의 사진을 GrayScale로 변환한 사진이고, 우측 사진은 명암 비 조절 과정을 진행한 사진이다. 이를 통해 주사위 면에 미치지 못하는 밝기를 가지는 배경은 어두워지게 만드는 과정을 추가하여 이진화를 잘 진행할 수 있었다.



ㄴ. 주사위 면 검출

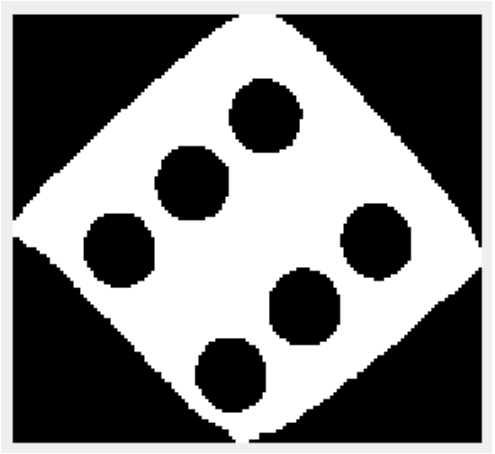
앞서 설명한 내용과 대부분 동일하다. connectedComponentsWithStats() 함수를 통해 labeling된 객체들을 검출하고, 이 객체들 중 주사위 면에 해당하는 객체들을 리스트에 넣어 반환해주었다.

검출 조건:

```
if abs(w-h)>15 or area < 2500:
    continue
if area < 100:
    continue
```

주사위의 흰 면은 2500 이상

너비 높이 차이가 15 이하



ㄷ. 주사위 눈금 검출

앞서 설명한 내용과 대부분 동일하다. connectedComponentsWithStats() 함수를 통해 labeling된 객체들을 검출하고, 이 객체들 중 주사위 눈금에 해당하는 객체들을 리스트에 넣어 반환해주었다.

또한 눈금으로 판단하는 조건 중, 해당 labeling된 객체 내에 또다른 요소가 없어야 눈금이 된다는 것도 추가하였다.

검출 조건:

```
if abs(w-h)>15 or area > 700:
    continue
if area < 200:
    continue
```

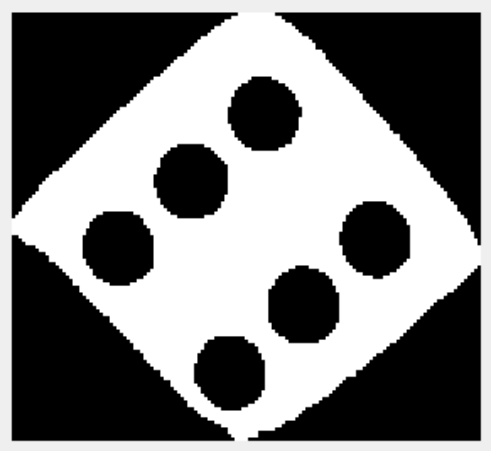
200 <= 눈금의 크기 <= 700

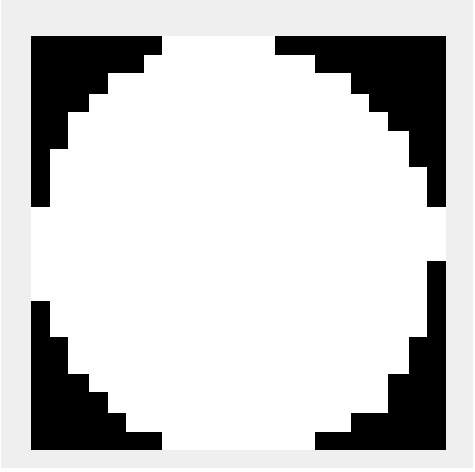
너비 높이 차이가 15 이하

```
cnt_test, _, _, _ = cv2.connectedComponentsWithStats(tmp)
if cnt_test == 2:
    dice_list.append(tmp)
```

눈금 내의 요소의 수인 cnt_test 값을 비교하여 눈금 판단.

눈금을 검출하기 위해 주사위 면을 반전시킨 뒤 함수 입력.





ㄹ. 취약점

검출 조건에 해당하는 주사위 면과 주사위 눈금의 경우를 검출하기 때문에 주사위가 확대된 사진의 경우 해당 조건에 검출이 되지 않는다는 단점이 될 수 있다.

4. 주사위 읽기(2): 책상 위에 놓인 임의의 주사위의 눈을 읽기(다수의 색)

소스코드:

<04.py>

```python
import cv2
import Binalization, Labeling, DiceNumber, ImageProcess

src = cv2.imread('case4/04_1.png', cv2.IMREAD_COLOR)


#binalization
src_bin = Binalization.binImg_04(src)
src_bin_INV = Binalization.binImg_04_INV(src)

#dice labeling
dice_list = Labeling.labelingImg_03_04(src_bin)
dice_list_INV = Labeling.labelingImg_03_04(src_bin_INV)

#getnumber from dice
dice_ans_list = DiceNumber.getDiceNumber_03_04(dice_list)
dice_ans_list_INV = DiceNumber.getDiceNumber_03_04(dice_list_INV)

#sort & print
dice_ans_list = dice_ans_list + dice_ans_list_INV
dice_ans_list.sort()
print("dice : ", dice_ans_list)
ImageProcess.showImg(src)

```

<binImg_04>

```python
def binImg_04(src) :
    src_blu = ImageProcess.blurring(src)
    bgr_planes = cv2.split(src_blu)
    _, src_bin_0 = cv2.threshold(bgr_planes[0], 155, 85, cv2.THRESH_BINARY )
    _, src_bin_1 = cv2.threshold(bgr_planes[1], 155, 85, cv2.THRESH_BINARY )
    _, src_bin_2 = cv2.threshold(bgr_planes[2], 155, 85, cv2.THRESH_BINARY )

    src_bin = src_bin_0 +  src_bin_1 + src_bin_2

    src_bin = cv2.dilate(src_bin, np.ones((3,3), np.uint8))
    _, src_bin = cv2.threshold(src_bin, 180, 255, cv2.THRESH_BINARY )

    return src_bin
```

<binImg_04_INV>

```
def binImg_04_INV(src) :
    src_blu = ImageProcess.blurring(src)
    bgr_planes = cv2.split(src_blu)
    _, src_bin_0 = cv2.threshold(bgr_planes[0], 155, 85, cv2.THRESH_BINARY )
    _, src_bin_1 = cv2.threshold(bgr_planes[1], 155, 85, cv2.THRESH_BINARY )
    _, src_bin_2 = cv2.threshold(bgr_planes[2], 155, 85, cv2.THRESH_BINARY )

    src_bin = src_bin_0 +  src_bin_1 + src_bin_2

    src_bin = cv2.dilate(src_bin, np.ones((3,3), np.uint8))
    _, src_bin = cv2.threshold(src_bin, 254, 0, cv2.THRESH_TOZERO_INV )
    _, src_bin = cv2.threshold(src_bin, 70, 255, cv2.THRESH_BINARY )

    return src_bin
```

<labelingImg_03_04>

```
def labelingImg_03_04(src) :
    cnt, _, stats, _ = cv2.connectedComponentsWithStats(src)
    dice_list = []
    for i in range(1,cnt):
        (x, y, w, h, area) = stats[i]

        if abs(w-h)>15 or area < 2500:
            continue
        tmp = src[y:y+h,x:x+w]
        dice_list.append(tmp)
    return dice_list
```

<labelingDice_03_04>

```
def labelingDice_03_04(src) :
    cnt, _, stats, _ = cv2.connectedComponentsWithStats(src)
    dice_list = []
    for i in range(1,cnt):
        (x, y, w, h, area) = stats[i]

        if abs(w-h)>15 or area > 700:
            continue
        if area < 200:
            continue
        tmp = src[y:y+h,x:x+w]
        cnt_test, _, _, _ = cv2.connectedComponentsWithStats(tmp)
        if cnt_test == 2:
            dice_list.append(tmp)
    return dice_list
```

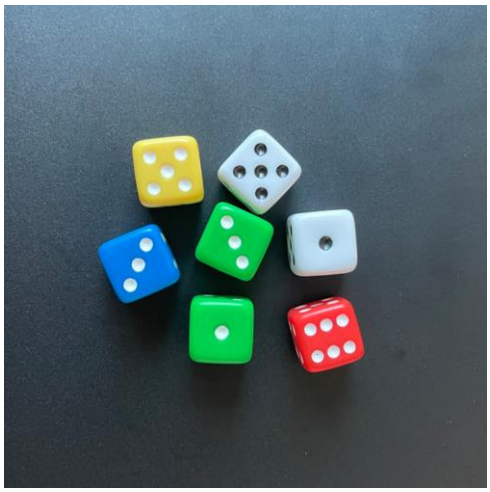
<getDiceNumber_03_04>

```
def getDiceNumber_03_04(dice_list) :
    ans = []
    tmp_dice = []
    for i in range(len(dice_list)):
        tmp_dice = labelingDice_03_04(~dice_list[i])
        if len(tmp_dice)<7 and len(tmp_dice)>0:
            ans.append(len(tmp_dice))
    return ans
```

수행결과:

```
pmj6541@ubuntu:~/22-3-2/CV/-2/02$ /bin/python3 /home/pmj6541/22-3-2/CV/-2/02/04.py
dice :  [1, 2, 2, 3, 3, 4, 4, 6, 6]
```

```
pmj6541@ubuntu:~/22-3-2/CV/-2/02$ /bin/python3 /home/pmj6541/22-3-2/CV/-2/02/04.py
dice :  [1, 1, 2, 3, 3, 4, 4, 6, 6]
```



```
pmj6541@ubuntu:~/22-3-2/CV/-2/02$ /bin/python3 /home/pmj6541/22-3-2/CV/-2/02/04.py
dice :  [1, 1, 3, 3, 5, 5, 6]
```

설명:

ㄱ. 이진화

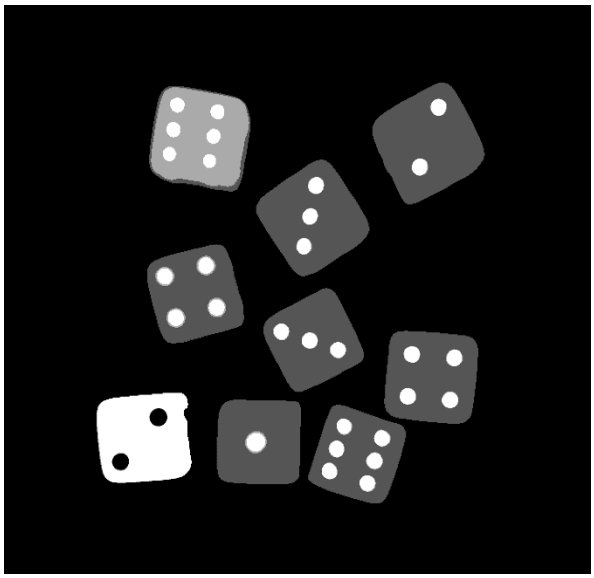
예시 데이터 내의 주사위 색은 흰색, BGR, 그리고 BGR중 두 요소를 결합한 색상ex)노란색이었다. 해당 색들의 주사위를 이진화 하기에는 흰색이 아닌 BGR요소로 면 색상이 이루어진 주사위들의 눈금의 색들이 흰색이어서 흰 주사위가 섞여 있는 사진 속에서는 이진화가 어렵다는 단점이 있었다. 이를 해결하기 위해 1)**흰색의 주사위를 검출하는 이진화**와, 2)**흰색을 검은색으로 변경하고, 흰색이 아닌 주사위 면들을 흰색으로 바꿔주는 이진화**를 진행하였다.

0) 공통

```
def blurring(src):
    for ksize in range(3, 9, 2):
        src = cv2.blur(src,(ksize,ksize))
    return src
```

```
src_blu = ImageProcess.blurring(src)
bgr_planes = cv2.split(src_blu)
_, src_bin_0 = cv2.threshold(bgr_planes[0], 155, 85, cv2.THRESH_BINARY )
_, src_bin_1 = cv2.threshold(bgr_planes[1], 155, 85, cv2.THRESH_BINARY )
_, src_bin_2 = cv2.threshold(bgr_planes[2], 155, 85, cv2.THRESH_BINARY )

src_bin = src_bin_0 +  src_bin_1 + src_bin_2
```

더 나은 이진화를 위해 사진에 평균값 필터를 이용해 블러링을 해주고,
각 BGR요소를 나누어 준 뒤 src_bin_0~2까지 이진화를 진행한다.
그 후 세 값들을 모두 합한다.

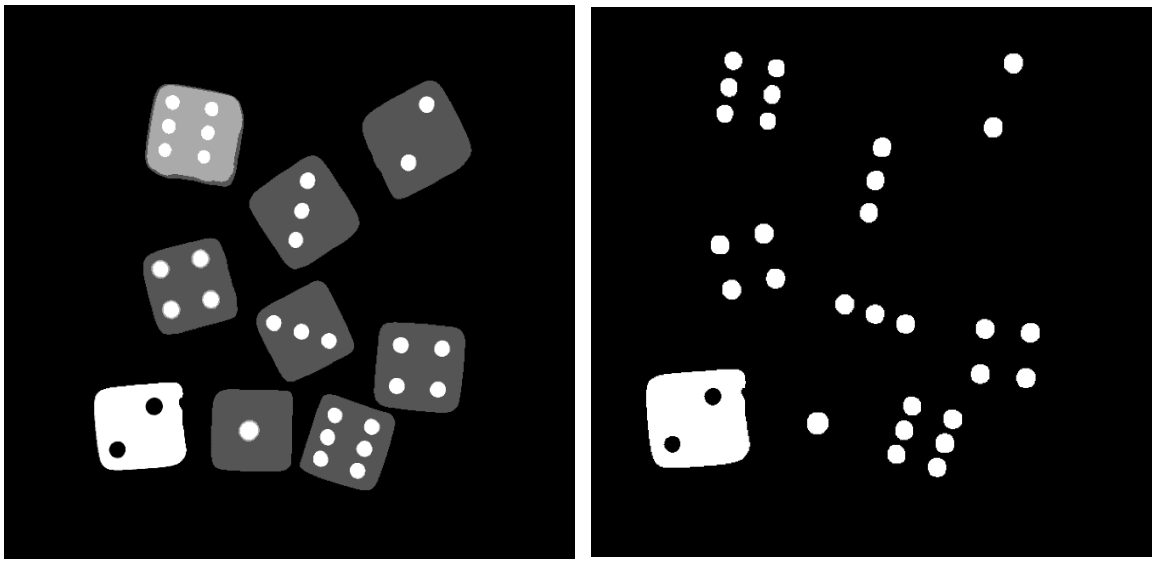


이진화 결과, 검은 계열은 0, BGR 각 색상은 85, BGR 중 두가지 혼합 색상은 170, 흰색은 255가 된다.

1) 흰색 주사위 이진화

```
src_bin = cv2.dilate(src_bin, np.ones((3,3), np.uint8))
_, src_bin = cv2.threshold(src_bin, 180, 255, cv2.THRESH_BINARY )
```

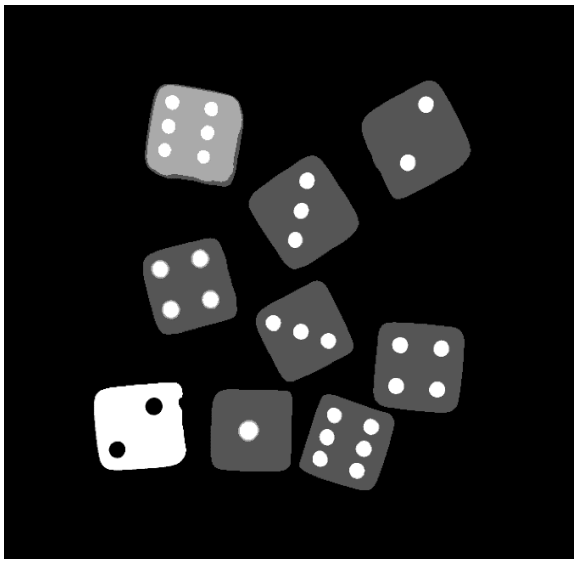
위의 이진화 결과를 더 나은 식별을 위해 dilate를 진행해 주고, 180(>170)을 Threshold값으로 한 이진화 작업을 진행하면 다음과 같은 결과가 나와 흰색 주사위 면에 대한 이진화 결과 값을 얻을 수 있다.



2) 색상 주사위 이진화

```
src_bin = cv2.dilate(src_bin, np.ones((3,3), np.uint8))
_, src_bin = cv2.threshold(src_bin, 254, 0, cv2.THRESH_TOZERO_INV )
_, src_bin = cv2.threshold(src_bin, 70, 255, cv2.THRESH_BINARY )
```

마찬가지로 더 나은 식별을 위해 dilate를 진행해주고, 255에 해당하는 픽셀들을 전부 0으로 바꾸고, 70이상(<85)의 픽셀들을 전부 0으로 바꾸어 주면 색상 주사위 면에 대한 이진화 결과 값을 얻을 수 있다.




ㄴ. 주사위 면 검출

#3과 동일한 함수 사용.

ㄷ. 주사위 눈금 검출

#3과 동일한 함수 사용

ㄹ. 취약점

#3과 마찬가지로 조건이 정해져 있어 주사위의 크기가 커지거나 작아지는 경우 검출이 어려울 수 있다.

#4에서 이진화를 할 때, 각 BGR요소의 Threshold 값을 155로 설정하고 진행하였다. 위 예시 사진의 주사위들 중에는 검정색이 없다. 검정색 주사위는 BGR요소가 전부 0에 해당하기 때문에 자동적으로 배경과 동일하게 처리되어 해당 주사위에 대한 labeling이 불가능 하다.

5. 주사위 읽기(3) : 여러 면이 보이는 주사위에 대해서 가장 넓은 면의 눈 출력

소스코드:

<05.py>

```python
import cv2
import Binalization, Labeling, DiceNumber, ImageProcess

src = cv2.imread('case5/05_1.png', cv2.IMREAD_COLOR)
src_gray = cv2.cvtColor(src,cv2.COLOR_BGR2GRAY)

#binalization
src_bin = Binalization.binImg_05(src_gray)


#dice labeling
dice_list = Labeling.labelingImg_05(src_bin)


#getnumber from dice
dice_ans_list = DiceNumber.getDiceNumber_05(dice_list)


#sort & print
dice_ans_list.sort()
print("dice : ", dice_ans_list)
ImageProcess.showImg(src)
```

<binImg_05>

```python
def binImg_05(src) :
    alpha = 3.0
    for y in range(src.shape[0]):
        for x in range(src.shape[1]):
            src[y, x] = ImageProcess.saturated(src[y, x] + (src[y, x]-190) * alpha)

    _, src_bin = cv2.threshold(src, 80, 255, cv2.THRESH_BINARY )

    src_bin = cv2.morphologyEx(src_bin, cv2.MORPH_OPEN, None)
    src_bin = cv2.morphologyEx(src_bin, cv2.MORPH_CLOSE, None)

    return src_bin
```

<labelingImg_05>

```
def labelingImg_05(src) :
    cnt, _, stats, _ = cv2.connectedComponentsWithStats(src)
    dice_list = []
    for i in range(1,cnt):
        (x, y, w, h, area) = stats[i]

        if area < 10000:
            continue
        if area < 100:
            continue
        tmp = src[y:y+h,x:x+w]
        dice_list.append(tmp)
    return dice_list
```

<labelingDice_05>

```
def labelingDice_05(src) :
    cnt, _, stats, _ = cv2.connectedComponentsWithStats(src)
    dice_list = []
    for i in range(1,cnt):
        (x, y, w, h, area) = stats[i]

        if abs(w-h)>15 or area > 2000:
            continue
        if area < 350:
            continue
        tmp = src[y:y+h,x:x+w]
        cnt_test, _, _, _ = cv2.connectedComponentsWithStats(tmp)
        if cnt_test == 2:
            dice_list.append(tmp)
    return dice_list
```

<getDiceNumber_05>

```
def getDiceNumber_05(dice_list) :
    ans = []
    tmp_dice = []
    for i in range(len(dice_list)):
        tmp_dice = labelingDice_05(~dice_list[i])
        if len(tmp_dice)<7 and len(tmp_dice)>0:
            ans.append(len(tmp_dice))
    return ans
```

수행결과:

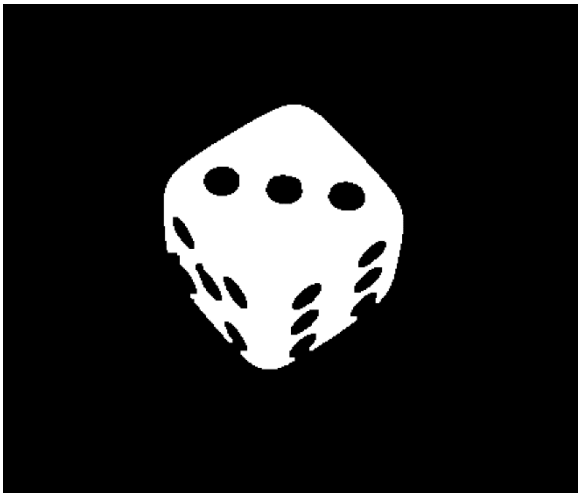
```
pmj6541@ubuntu:~/22-3-2/CV/-2/02$ /bin/python3 /home/pmj6541/22-3-2/CV/-2/02/05.py
dice :  [3]
```

```
pmj6541@ubuntu:~/22-3-2/CV/-2/02$ /bin/python3 /home/pmj6541/22-3-2/CV/-2/02/05.py
dice :  [3]
```

설명:

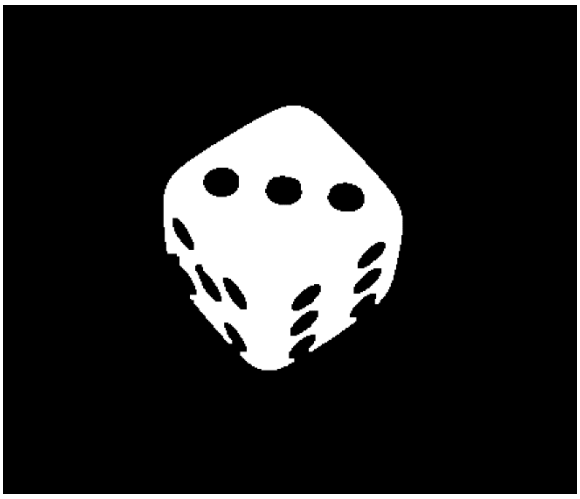
ㄱ. 이진화

예시 사진의 배경에 맞추어 #3과 비슷한 형태로 진행하였다.



ㄴ. 주사위 면 검출

#3과 비슷하지만 검출하는 주사위 면의 크기를 더 늘려 주사위 전체를 검출할 수 있도록 수정해주었다.



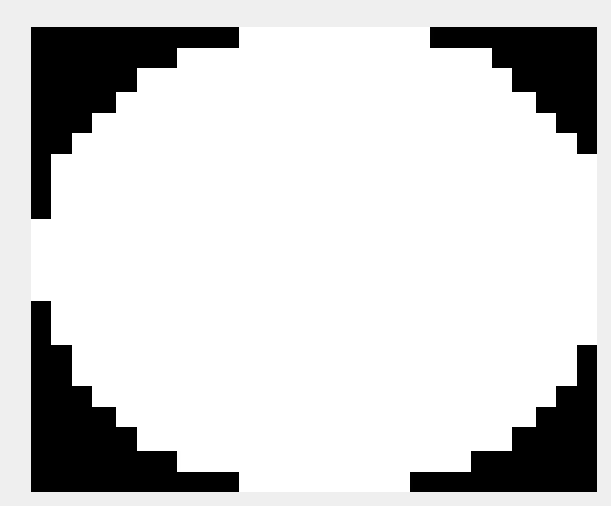
검출 조건:

```
if area < 10000:
    continue
```

주사위 전체를 검출해야 하기 때문에 10000이상의 면적 검출

ㄷ. 주사위 눈금 검출

#3과 비슷하지만 주사위 전체에서 보이는 눈금들 중 찾고자 하는 면의 눈금을 검출해야 하기 때문에 조건을 수정해주었다.



검출 조건:

```
if abs(w-h)>15 or area > 2000:
    continue
if area < 350:
    continue
```

주사위 눈금의 너비, 높이 차가 15이하

350<주사위 눈금의 면적<2000

ㄹ. 취약점

#3와 비슷하지만, 찾고자 하는 주사위 면의 눈금은 다른 눈금들과 비교해서 더 넓다는 직관을 이용하여 해당 문제를 해결하였다. 위의 #3, #4와 마찬가지로 조건이 사진에 나와있는 주사위의 크기에 따라 달라지기 때문에 일정 규격에서 벗어난 주사위 사진에 원하는 결과를 얻어내지 못할 확률이 높다.