AI모빌리티

송경환 교수

tell : 010-3054-9340

email : 2020021@itc.ac.kr

시험은 객관식 다수, 코드 문제 나옴. 영상보고 다시 한 번 정리.

범위는 사실상 5주차 까지 4분짜리 영상들 위주로 나옴

https://github.com/ndb796/Python-Data-Analysis-and-Image-Processing-Tutorial

설치 : https://developer.nvidia.com/embedded/downloads

▼ 2주차

OpenCV : 컴퓨터 비전을 위해 사용

- 영상 처리와 컴퓨터 비전을 위한 오픈소스 라이브러리
- C, C++, Python 등에서 사용

OpenCV 사용법

```
cv2.imread(file_name,flag)
```

이미지를 읽어 Numpy 객체로 만드는 함수

- file_name : 읽고자 하는 이미지 파일
- flag : 이미지를 읽는 방법 설정

IMREAD_COLOR : 이미지를 CoLOR로 읽고, 투명한 부분은 무시

IMREAD_GRAYSCALE : 이미지를 Grayscale로 읽기

IMAREAD_UNCHANGED : 이미지를 Color로 읽고, 투명한 부분도 읽기(Alpha)

반환 값 : Numpy객체 (행, 열, 색상 :기본 BGR)

```
cv2.namedWindow(winname, flags=None)
```

새 창 띄우기

- winname : 창 고유 이름
- flag : 창 속성 지정 플래그

cv2.WINDOW_NORMAL : 영상 크기가 창 크기에 맞게 지정됨

cv2.WINDOW_AUTOSIZE : 창 크기가 영상 크기에 맞게 자동으로 변경됨

```
cv2.imshow(title, image)
```

특정한 이미지를 화면에 출력

- title : 윈도우 창의 제목

- image : 출력한 이미지 객체

```
cv2.waitKey(time)
```

키보드 입력을 처리하는 함수

- time : 입력 대기 시간 (무한대기 : 0)

반환 값 : 사용자가 입력한 Ascii Code (Esc : 27)

```
cv2.destroyAllWindows()
```

화면의 모든 윈도우 닫는 함수

```
cv2.cvtColor(file_name,flag)
```

이미지 변환

- file_name : 변환할 대상 이미지
- flag : 변환할 색 공간

```
cv2.imwrite(title, file_name)
```

이미지 파일 저장

- title : 저장할 파일 이름 지정
- file_name : 저장할 이미지 변수

```
cv2.moveWindow(winname, x, y)
```

창 위치 지정

- winname : 창 이름
- x, y : 이동할 위치 좌표

```
cv2.resizeWindow(winname, width, height)
```

크기 지정

- winname : 창 이름
- width, height : 변경할 창 크기

```
import cv2
img_basic = cv2.imread('cat.jpg', cv2.IMREAD_COLOR) #CAT 이미지를 컬러를 포함해서 읽음
cv2.imshow('Image Basic',img_basic) #보여줄 제목, 파일 변수
cv2.waitKey(0)
cv2.imwrite('result.png',img_basic) #저장할 제목, 파일 변수

cv2.destroyAllWindows() #화면의 윈도우 닫기.
```

```
img_gray= cv2.cvtColor(img_basic,cv2.COLOR_BGR2GRAY) #bgr형태로 이루어진 넘파이 객체를 GRAY형태로 바꾸겠다.
cv2.imshow('img_gray',img_gray)
cv2.waitKey(0)
cv2.imwrite('result2.png',img_gray)
```

▼ 3주차

컴퓨터 비전

- 사람의 시각시스템을 컴퓨터가 모사하는 시스템. 높은 수준의 이해

컴퓨터 비전 관련분야

- 수학
- 신호처리
- 컴퓨터 과학 등

컴퓨터 비전 연구 분야

- 객체 검출
- 분할
- 인식
- 움직임 정보 및 추적

컴퓨터 비전 응용 분야

- 머신 비전(machine vision)

  공장 자동화 : 제품의 불량 검사, 위치확인, 측정 등

  높은 정확도와 빠른 처리 시간 요구

  조명, 렌즈, 필터, 실시간 처리
- 인공지능 서비스

  인공지능 로봇과 자율 주행 자동차

  입력 영상을 객체와 배경으로 분할 → 객체와 배경 인식 → 상황인식 → 로봇과 자동차의 행동 지시

  Computer Vision + Sensor Fusion + Deep Learning

  Amazon Go / 구글, 테슬라의 자율 주행 자동차

영상의 표현 방법

- 영상(image) : 픽셀이 바둑판 모양의 격자에 나열되어 있는 형태(2차원 행렬)
- 픽셀(pixel) : 영상의 기본 단위, 화소

- 영상에서 사용되는 좌표계

  w x h 영상 (w-by-h image)

  M x N 행렬 (m-by-n matrix)

- 그레이스케일 영상

  색상 정보가 없이 오직 밝기 정보만으로 구성된 영상

  밝기 정보를 256 단계로 표현

  python에서는 numpy.uint8을 사용

  영상크기 : 가로크기 x 세로크기 Bytes

- 트루컬러 영상

  컬러 사진처럼 색상 저보를 가지고 있어서 다양한 색상을 표현할 수 있는 영상

  Red, Green, Blue 색 성분을 256 단계로 표현 → 256^3 = 16,777,216 색상 표현 가능

  영상크기 : 가로크기 x 세로크기 x 3 Bytes

영상 파일 형식 특징

- BMP : 픽셀 데이터를 압축하지 않고 그대로 저장 - >용량 큼
- JPG : 주로 사진과 같은 컬러 영상을 저장하기 위해 사용 → 압출률이 좋음
- GIF : 256 색상 이하의 영상을 저장 → 움직이는 GIF 지원
- PNG : 무손실 압축, 알파 채널을 지원

OpenCV 설치

- pip install opencv-python

▼ 4주차

OpenCV 버전확인

```
cv2.__version__ #OpenCV 버전확인
```

이미지의 자료형 확인

```
print(type(img)) #자료형 확인
print(img.shape) #이미지의 가로, 세로, 구성된 색(bgr)
print(img.dtype) #이미지의 데이터 타입(uint8)
```

비디오캡쳐

```
cv2.VideoCapture(filename, apipreference=None)
cv2.VideoCapture(index, apipreference=None)
```

동영상, 정지 영상 시퀀스, 비디오 스트림 열기

- filename : 비디오 파일 이름, 정지영상 시퀀스, 비디오 스트림 URL
- apipreference : cv2.VideoCapture 객체

카메라 열기

- index : 0이면 시스템 기본 카메라
- apipreference : cv2.VideoCapture 객체

외곽선 검출

```
cv2.Canny(file_name, 50, 150) #외곽선 검출
```

비디오 저장

```
a=cv2.VideoWriter(filename, fourcc, fps, frameSize, isColor=None)
a.write(저장할 이미지 변수)
```

동영상 파일 저장하기

- filename : 저장할 파일 이름
- fourcc : fourcc <코덱

- fps : 초당 프레임 수
- frameSize : 프레임 크기
- isColor : 컬려영상이면 True, 아니면 False

이진화

```
cv2.threshold(src, thresh, maxval, type[, dst])
```

- src : 이진화할 대상 이미지 그레이 스케일이여야 함.
- thresh : 쓰레쉬홀드 값
- maxval : 픽셀값
- type[, dst] : cv2.THRESH_BINARY

이진화된 이미지는 임계값 기준으로 그레이스케일이미지를 2개의 영영으로 나눔 픽셀값이 높으면 흰색, 낮으면 검은색

트랙바

```
import cv2
def nothing(x):
    pass

cv2.namedWindow('Binary') #트랙바 설정을 위한 윈도우 설정
cv2.createTrackbar('threshold', 'Binary', 0, 255, nothing)
cv2.setTrackbarPos('threshold', 'Binary', 127) #트랙바의 기본값을 127로 설정

img_color = cv2.imread('red_ball.jpg', cv2.IMREAD_COLOR)
cv2.imshow('Color', img_color)

img_gray = cv2.cvtColor(img_color, cv2.COLOR_BGR2GRAY)
cv2.imshow('Gray', img_gray)

while(True):
    low = cv2.getTrackbarPos('threshold', 'Binary') #트랙바의 값을 받아옴
    #트랙바의 값에 따라 트레쉬홀드를 변화시킴.
    ret,img_binary = cv2.threshold(img_gray, low, 255, cv2.THRESH_BINARY_INV)

    cv2.imshow('Binary', img_binary)

    #원본 이미지와 트레쉬홀드를 적용한 이미지의 교집합을 찾음.
    img_result = cv2.bitwise_and(img_color, img_color, mask = img_binary)
    cv2.imshow('Result', img_result)

    if cv2.waitKey(1)&0xFF == 27:
        break

cv2.destroyAllWindows()
```

RGB와 HSV

- RGB는 3개의 색공간을 혼합하여 색을 표현하기 때문에 원하는 색의 범위를 구하기 힘듬
- HSV는 Hue 성분이 일정한 영역을 가져 원하는 색의 범위를 구하기 쉬움.

```
import cv2

img_color = cv2.imread('1.jpg')
height,width = img_color.shape[:2] #이미지의 가로 세로 크기를 가져옴.

img_hsv = cv2.cvtColor(img_color, cv2.COLOR_BGR2HSV)#HSV이미지로 변환함.


lower_blue = (120-10, 30, 30) #최소영역을 지정함.
upper_blue = (120+10, 255, 255) #최대영역을 지정함.
img_mask = cv2.inRange(img_hsv, lower_blue, upper_blue) #범위내에 있는 영역을 추출함. 흰색


#원본 이미지와 추출한 파란색의 교집함을 찾음.
```

```
img_result = cv2.bitwise_and(img_color, img_color, mask = img_mask)


cv2.imshow('img_color', img_color)
cv2.imshow('img_mask', img_mask)
cv2.imshow('img_result', img_result)


cv2.waitKey(0)
cv2.destroyAllWindows()
```

HSV 색 공간에서 특정색 검출하기

```
import cv2 as cv
import numpy as np


hsv = 0
lower_blue1 = 0
upper_blue1 = 0
lower_blue2 = 0
upper_blue2 = 0
lower_blue3 = 0
upper_blue3 = 0


def nothing(x):
    pass

def mouse_callback(event, x, y, flags, param):
    global hsv, lower_blue1, upper_blue1, lower_blue2, upper_blue2, lower_blue3, upper_blue3, threshold

    # 마우스 왼쪽 버튼 누를시 위치에 있는 픽셀값을 읽어와서 HSV로 변환합니다.
    if event == cv.EVENT_LBUTTONDOWN:
        print(img_color[y, x])
        color = img_color[y, x]

        one_pixel = np.uint8([[color]])
        hsv = cv.cvtColor(one_pixel, cv.COLOR_BGR2HSV)
        hsv = hsv[0][0]

        threshold = cv.getTrackbarPos('threshold', 'img_result')

        # HSV 색공간에서 마우스 클릭으로 얻은 픽셀값과 유사한 필셀값의 범위를 정합니다.
        if hsv[0] < 10:
            print("case1")
            lower_blue1 = np.array([hsv[0]-10+180, threshold, threshold])
            upper_blue1 = np.array([180, 255, 255])
            lower_blue2 = np.array([0, threshold, threshold])
            upper_blue2 = np.array([hsv[0], 255, 255])
            lower_blue3 = np.array([hsv[0], threshold, threshold])
            upper_blue3 = np.array([hsv[0]+10, 255, 255])
            #     print(i-10+180, 180, 0, i)
            #     print(i, i+10)

        elif hsv[0] > 170:
            print("case2")
            lower_blue1 = np.array([hsv[0], threshold, threshold])
            upper_blue1 = np.array([180, 255, 255])
            lower_blue2 = np.array([0, threshold, threshold])
            upper_blue2 = np.array([hsv[0]+10-180, 255, 255])
            lower_blue3 = np.array([hsv[0]-10, threshold, threshold])
            upper_blue3 = np.array([hsv[0], 255, 255])
            #     print(i, 180, 0, i+10-180)
            #     print(i-10, i)
        else:
            print("case3")
            lower_blue1 = np.array([hsv[0], threshold, threshold])
            upper_blue1 = np.array([hsv[0]+10, 255, 255])
            lower_blue2 = np.array([hsv[0]-10, threshold, threshold])
            upper_blue2 = np.array([hsv[0], 255, 255])
            lower_blue3 = np.array([hsv[0]-10, threshold, threshold])
            upper_blue3 = np.array([hsv[0], 255, 255])
            #     print(i, i+10)
            #     print(i-10, i)

        print(hsv[0])
        print("@1", lower_blue1, "~", upper_blue1)
        print("@2", lower_blue2, "~", upper_blue2)
        print("@3", lower_blue3, "~", upper_blue3)


cv.namedWindow('img_color')
```

```
cv.setMouseCallback('img_color', mouse_callback)

cv.namedWindow('img_result')
cv.createTrackbar('threshold', 'img_result', 0, 255, nothing)
cv.setTrackbarPos('threshold', 'img_result', 30)

cap = cv.VideoCapture(0)

while(True):
    #img_color = cv.imread('2.jpg')
    ret,img_color = cap.read()
    height, width = img_color.shape[:2]
    img_color = cv.resize(img_color, (width, height), interpolation=cv.INTER_AREA)

    # 원본 영상을 HSV 영상으로 변환합니다.
    img_hsv = cv.cvtColor(img_color, cv.COLOR_BGR2HSV)

    # 범위 값으로 HSV 이미지에서 마스크를 생성합니다.
    img_mask1 = cv.inRange(img_hsv, lower_blue1, upper_blue1)
    img_mask2 = cv.inRange(img_hsv, lower_blue2, upper_blue2)
    img_mask3 = cv.inRange(img_hsv, lower_blue3, upper_blue3)
    img_mask = img_mask1 | img_mask2 | img_mask3

    kernel = np.ones((11,11), np.uint8)
    img_mask = cv.morphologyEx(img_mask, cv.MORPH_OPEN, kernel)
    img_mask = cv.morphologyEx(img_mask, cv.MORPH_CLOSE, kernel)

    # 마스크 이미지로 원본 이미지에서 범위값에 해당되는 영상 부분을 획득합니다.
    img_result = cv.bitwise_and(img_color, img_color, mask=img_mask)


    numOfLabels, img_label, stats, centroids = cv.connectedComponentsWithStats(img_mask)

    for idx, centroid in enumerate(centroids):
        if stats[idx][0] == 0 and stats[idx][1] == 0:
            continue

        if np.any(np.isnan(centroid)):
            continue

        x,y,width,height,area = stats[idx]
        centerX,centerY = int(centroid[0]), int(centroid[1])
        print(centerX, centerY)

        if area > 50:
            cv.circle(img_color, (centerX, centerY), 10, (0,0,255), 10)
            cv.rectangle(img_color, (x,y), (x+width,y+height), (0,0,255))

    cv.imshow('img_color', img_color)
    cv.imshow('img_mask', img_mask)
    cv.imshow('img_result', img_result)


    # ESC 키누르면 종료
    if cv.waitKey(1) & 0xFF == 27:
        break


cv.destroyAllWindows()
```

컨투어 : 특정영역의 경계를 따라 같은 픽셀 강도를 갖는 지점을 연결하는 선

- 모양분석, 오브젝트 검출에 사용

컨투어 찾기

```
contours, hierarchy = cv2.findContours(image, mode, method[,contours[,hierarchy[,offset]]])
```

- image : 입력 이미지, 흰색과 검은색으로만 이루어진 이미지여야함.(이진화)
- contours : 검출된 컨투어, 오브젝트의 외곽선을 구상하는 x,y좌표를 리스트로 저장됨.
- hierarchy : 검출된 컨투어의 정보를 구조적으로 저장함. 리스트로 저장됨.
- mode : 검출된 엣지 정보를 리스트로 저장하는 방식을 지정함

  데이터는 [Next, Previous, First_Child, Parent]로 구성됨.

  RETR_TREE : 컨투어 내부에 다른 컨투어가 있으면 계층구조로 만듬.

RETR_LIST : 모든 컨투어가 같은 계층 레벨을 가짐. [Next, Previous만 값이 있음 나머지 -1]

RETR_EXTERNAL : 가장 외곽에 있는 컨투어만 그림.

RETR_CCOMP : 모든 컨투어를 2개의 레벨 계층으로 재구성함.

- method : 컨투어를 구성하는 포인트 검출하는 방법을 지정함

  cv.CHAIN_APPROX_NONE : 모든 경계점 저장

  cv.CHAIN_APPROX_SIMPLE : 시작점과 끝점만 저장.

- offset : 지정한 크기만큼 컨투어를 구성하는 포인터의 좌표를 이동하여 저장

컨투어 그리기

```
image = cv.drawContours(image, contours, contourIdx, color[, thickness[, lineType
[, hierarchy[, maxLevel[, offset]]]]])
```

- image : 컨투어를 그릴 이미지(컬러이미지)
- contours : 이미지 위에 그릴 컨투어가 저장된 리스트 혹은 벡터
- contourIdx : 이미지에 그릴 특정 컨투어의 인덱스 음수로 지정하면 모든 컨투어를 그림
- color : 컨투어의 색상 bgr순서로 적어야함.
- thickness : 컨투어 선의 굵기

영역크기

```
for cnt in contours:
  area = cv.contourArea(cnt) #영역크기를 구함.
  print(area)
```

근사화

```
for cnt in contours:
  epsilon = 0.02 * cv.arcLength(cnt, True) # contours 둘레의 길이를 계산하는 함수
  approx = cv.approxPolyDP(cnt, epsilon, True) #
```

- arcLength()함수는 contours 둘레의 길이를 계산하는 함수이다.

  parameter1: 둘레를 계산할 contour이다.

  parameter2: contour가 폐곡선인지 계곡선인지 여부를 타나낸다. True는 폐곡선을 의미한다.

  0.1: 둘레의 길이에 10%를 나타낸다.

  epsilon: 근사 정확도이다.

- approxPolyDP()함수는 다각형을 대상으로 꼭지점을 점점 줄여나가는 함수이다. epsilon(오차)만큼을 최대한으로 해서 꼭지점을 줄여나간다. 그래서 epsilon값이 작을수록 원본과 비슷한 결과가 도출되고 epsilon(오차)값이 크면 클수록 꼭지점의 개수가 점점 더 줄어든다. epsilon이 너무 크면 꼭지점이 계속해서 줄어나가게 되니까 결국 꼭지점의 개수가 0인 점으로 결과가 나올 수 있다. 그래서 parameter1에는 꼭지점의 개수를 줄일 contour를 넣고 parameter2로 줄여나갈 오차 epsilon을 넣고 parameter3는 폐곡선인지 계곡선인지 여부를 넣는다.

무게중심

```
for cnt in contours:
  M = cv.moments(cnt) #공간 모멘트를 구함.

  cx = int(M['m10']/M['m00'])
  cy = int(M['m01']/M['m00'])

  cv.circle(img_color, (cx,cy), 10, (0,0,255), -1)
```

경계사각형

```
for cnt in contours:
  x,y,w,h = cv.boundingRect(cnt)
  cv.rectangle(img_color, (x,y), (x+w, y+h), (0, 255, 0), 3)

for cnt in contours:
  rect = cv.minAreaRect(cnt) # 방향을 고려해서 그려짐.
  box = cv.boxPoints(rect)
  box = np.int0(box)
  cv.drawContours(img_color, [box], 0, (0,0,255),3)
```

Convex Hull

```
for cnt in contours:
  hull = cv.convexHull(cnt) #컨투어를 모두 포함하는 블록 다각형을 그리는 것
  cv.drawContours(img_color, [hull], 0, (255,0,255),5)
```

Convexity Defects

```
for cnt in contours:
    hull = cv.convexHull(cnt, returnPoints = False)#컨투어를 모두 포함하는 블록 다각형을 그리는 것
    defects = cv.convexityDefects(cnt, hull) #블록다각형 내부에 들어가부분 체크
    #[시작점, 끝점, 가장 먼 지점, 가장 가까운 지점까지의 대략적인 거리]반환

    for i in range(defects.shape[0]):
        s,e,f,d = defects[i,0]
        start = tuple(cnt[s][0])
        end = tuple(cnt[e][0])
        far = tuple(cnt[f][0])

        print(d)


        if d > 500:
            cv.line(img_color, start, end, [0, 255, 0], 5)
            cv.circle(img_color, far, 5, [0,0,255], -1)
```

Hough Line Transform : 이미지에서 직선 검출시 사용

- sin곡선이 겹치는 부분을 찾아 직선으로 연결함.

```
import cv2 as cv
import numpy as np
import math
import time


img_original = cv.imread('1.jpg', cv.IMREAD_COLOR)

img_gray = cv.cvtColor(img_original, cv.COLOR_BGR2GRAY)
img_edge = cv.GaussianBlur(img_gray, (5, 5), 0, 0)
img_edge = cv.Canny(img_edge, 50, 150, 3)

height = img_edge.shape[0]
width = img_edge.shape[1]
tmp = min(height, width)
hough_height = int(1.5 * tmp)


accumulator_width = 180
accumulator_height = hough_height * 2
accumulator_size = accumulator_height * accumulator_width


accumulator = np.zeros((accumulator_height, accumulator_width))
table_sin = np.zeros(180)
table_cos = np.zeros(180)
DEG2RAD = np.pi / 180


for angle in range(0,180):
    table_sin[angle] = math.sin(np.radians(angle))
    table_cos[angle] = math.cos(np.radians(angle))


start=time.process_time()
```

```
for y in range(0,height):
    for x in range(0,width):
        if img_edge.item(y, x) > 0:
            for angle in range(0,180):
                r = int(x * table_cos[angle] + y * table_sin[angle])
                r = r + hough_height # r이 음수인 경우 때문 -r ~ r 범위를 0 ~ 2r 범위로 변경
                accumulator[r, angle] +=1

end=time.process_time()
print(end - start)



# accumulator를 이미지화
img_accumulator = np.ones((accumulator_height, accumulator_width, 3), np.uint8)
img_accumulator = img_accumulator * 255

accumulator2 = cv.convertScaleAbs(accumulator,3,5)

start=time.process_time()
for r in range(0,accumulator_height):
    for angle in range(0,accumulator_width):

        value = accumulator2[r, angle]

        if value > 0:
            img_accumulator.itemset(r, angle, 0, 255 - value)
            img_accumulator.itemset(r, angle, 1, 255 - value)
            img_accumulator.itemset(r, angle, 2, 255 - value)


end=time.process_time()
print(end - start)


start=time.process_time()
count = 0
for r in range(0, accumulator_height):
    for angle in range(0,180):

        if accumulator.item(r,angle) > 80: # Hough Line Transform Threshold

            #현재 위치가 local maxima인지 검사
            max = accumulator[r, angle]
            for y in range(-5,6):
                for x in range(-5,6):

                    new_r = r + y
                    new_angle = angle + x

                    if new_angle < 0:
                        new_angle = 180 + new_angle
                    elif new_angle >= 180:
                        new_angle = new_angle - 180

                    if new_r >= 0 and new_r < accumulator_height:
                        if accumulator[new_r, new_angle] > max:
                            max = accumulator[new_r, new_angle]
                            x = y = 6 #local maxima 아님. loop 종료

            if max > accumulator.item(r, angle):
                continue #현재 위치는 local maxima가 아님


            # r = x * cos(theta) + y * sin(theta)
            # x = (r - y * sin(theta)) / cos(theta) # 수직선인 경우
            # y = (r - x * cos(theta)) / sin(theta) # 수평선인 경우


            if angle >= 45 and angle <= 135: # 수직선
                x1 = 0
                x2 = width
                y1 = ((r - hough_height) - x1 * table_cos[angle]) / table_sin[angle]
                y2 = ((r - hough_height) - x2 * table_cos[angle]) / table_sin[angle]

            else: #수평선
                y1 = 0
                y2 = height
                x1 = ((r - hough_height) - y1 * table_sin[angle]) / table_cos[angle]
                x2 = ((r - hough_height) - y2 * table_sin[angle]) / table_cos[angle]


            x1 = int(x1)
            y1 = int(y1)
            x2 = int(x2)
            y2 = int(y2)
```

```
            cv.circle(img_accumulator, (angle, r), 5, (255, 0, 0),-1)
            cv.line(img_original, (x1, y1), (x2, y2), (255, 0, 0), 1)
            count += 1

            print("(%d,%d)-(%d,%d), angle=%d, r=%d, accmulator=%d" % (x1,y1,x2,y2,angle,r,accumulator.item(r, angle)))
end=time.process_time()
print(end - start)

cv.imshow("img_result", img_original)
cv.imshow("img_gray", img_gray)
cv.imshow("img_edge", img_edge)
cv.imshow("img_accumulator", img_accumulator)
cv.imwrite("img_accumulator.jpg", img_accumulator)


cv.waitKey(0)
```

두가지 색 검출하기

```
import cv2 as cv
import numpy as np


color1 = 0
color2 = 0

ranges = 20
set_color = False
step = 0


def nothing(x):
    global color1, color2
    global lower_blueA1, lower_blueA2, lower_blueA3
    global upper_blueA1, upper_blueA2, upper_blueA3
    global lower_blueB1, lower_blueB2, lower_blueB3
    global upper_blueB1, upper_blueB2, upper_blueB3

    #새츄레이션과 벨류 설정을 위한  트랙바 값을 가져옵니다.
    saturation_th1 = cv.getTrackbarPos('saturation_th1', 'img_result')
    value_th1 = cv.getTrackbarPos('value_th1', 'img_result')

    saturation_th2 = cv.getTrackbarPos('saturation_th2', 'img_result')
    value_th2 = cv.getTrackbarPos('value_th2', 'img_result')

    color1 = int(color1)
    color2 = int(color2)

    # HSV 색공간에서 마우스 클릭으로 얻은 픽셀값과 유사한 필셀값의 범위를 정합니다.
    if color1 < ranges:
        lower_blueA1 = np.array([color1 - ranges + 180, saturation_th1, value_th1])
        upper_blueA1 = np.array([180, 255, 255])
        lower_blueA2 = np.array([0, saturation_th1, value_th1])
        upper_blueA2 = np.array([color1, 255, 255])
        lower_blueA3 = np.array([color1, saturation_th1, value_th1])
        upper_blueA3 = np.array([color1 + ranges, 255, 255])
        #     print(i-range+180, 180, 0, i)
        #     print(i, i+range)

    elif color1 > 180 - ranges:
        lower_blueA1 = np.array([color1, saturation_th1, value_th1])
        upper_blueA1 = np.array([180, 255, 255])
        lower_blueA2 = np.array([0, saturation_th1, value_th1])
        upper_blueA2 = np.array([color1 + ranges - 180, 255, 255])
        lower_blueA3 = np.array([color1 - ranges, saturation_th1, value_th1])
        upper_blueA3 = np.array([color1, 255, 255])
        #     print(i, 180, 0, i+range-180)
        #     print(i-range, i)
    else:
        lower_blueA1 = np.array([color1, saturation_th1, value_th1])
        upper_blueA1 = np.array([color1 + ranges, 255, 255])
        lower_blueA2 = np.array([color1 - ranges, saturation_th1, value_th1])
        upper_blueA2 = np.array([color1, 255, 255])
        lower_blueA3 = np.array([color1 - ranges, saturation_th1, value_th1])
        upper_blueA3 = np.array([color1, 255, 255])
        #     print(i, i+range)
        #     print(i-range, i)

    #두번째 색을 추출하기 위한 범위를 계산합니다.
    if color2 < ranges:
        lower_blueB1 = np.array([color2 - ranges + 180, saturation_th2, value_th2])
        upper_blueB1 = np.array([180, 255, 255])
        lower_blueB2 = np.array([0, saturation_th2, value_th2])
```

```
        upper_blueB2 = np.array([color2, 255, 255])
        lower_blueB3 = np.array([color2, saturation_th2, value_th2])
        upper_blueB3 = np.array([color2 + ranges, 255, 255])
        #     print(i-range+180, 180, 0, i)
        #     print(i, i+range)

    elif color2 > 180 - ranges:
        lower_blueB1 = np.array([color2, saturation_th2, value_th2])
        upper_blueB1 = np.array([180, 255, 255])
        lower_blueB2 = np.array([0, saturation_th2, value_th2])
        upper_blueB2 = np.array([color2 + ranges - 180, 255, 255])
        lower_blueB3 = np.array([color2 - ranges, saturation_th2, value_th2])
        upper_blueB3 = np.array([color2, 255, 255])
        #     print(i, 180, 0, i+range-180)
        #     print(i-range, i)
    else:
        lower_blueB1 = np.array([color2, saturation_th2, value_th2])
        upper_blueB1 = np.array([color2 + ranges, 255, 255])
        lower_blueB2 = np.array([color2 - ranges, saturation_th2, value_th2])
        upper_blueB2 = np.array([color2, 255, 255])
        lower_blueB3 = np.array([color2 - ranges, saturation_th2, value_th2])
        upper_blueB3 = np.array([color2, 255, 255])
        #     print(i, i+range)
        #     print(i-range, i)


cv.namedWindow('img_color')
cv.namedWindow('img_result')

cv.createTrackbar('saturation_th1', 'img_result', 0, 255, nothing)
cv.setTrackbarPos('saturation_th1', 'img_result', 30)
cv.createTrackbar('value_th1', 'img_result', 0, 255, nothing)
cv.setTrackbarPos('value_th1', 'img_result', 30)
cv.createTrackbar('saturation_th2', 'img_result', 0, 255, nothing)
cv.setTrackbarPos('saturation_th2', 'img_result', 30)
cv.createTrackbar('value_th2', 'img_result', 0, 255, nothing)
cv.setTrackbarPos('value_th2', 'img_result', 30)

cap = cv.VideoCapture(0)


while(True):

    ret,img_color = cap.read() #웹캠으로부터 영상추출
    img_color = cv.flip(img_color, 1) #거울처럼 좌우 전환

    if ret == False:
        continue;


    img_color2 = img_color.copy()
    img_hsv = cv.cvtColor(img_color2, cv.COLOR_BGR2HSV)

    height, width = img_color.shape[:2]
    cx = int(width / 2)
    cy = int(height / 2)


    if set_color == False: #색을 입력받기 전이라면

        rectangle_color = (0, 255, 0) #녹색 사각형을 화면에 그림.

        if step == 1:
            rectangle_color = (0, 0, 255) #두번째 색을 입력 받기 위한 빨간색 사각형을 그려줍니다.

        cv.rectangle(img_color, (cx - 20, cy - 20), (cx + 20, cy + 20), rectangle_color, 5)


    else:

        # 범위 값으로 HSV 이미지에서 마스크를 생성합니다.
        img_maskA1 = cv.inRange(img_hsv, lower_blueA1, upper_blueA1)
        img_maskA2 = cv.inRange(img_hsv, lower_blueA2, upper_blueA2)
        img_maskA3 = cv.inRange(img_hsv, lower_blueA3, upper_blueA3)
        temp = cv.bitwise_or(img_maskA1, img_maskA2)
        img_maskA = cv.bitwise_or(img_maskA3, temp)

        img_maskB1 = cv.inRange(img_hsv, lower_blueB1, upper_blueB1)
        img_maskB2 = cv.inRange(img_hsv, lower_blueB2, upper_blueB2)
        img_maskB3 = cv.inRange(img_hsv, lower_blueB3, upper_blueB3)
        temp = cv.bitwise_or(img_maskB1, img_maskB2)
        img_maskB = cv.bitwise_or(temp, img_maskB3)


        # 모폴로지 연산
        kernel = np.ones((11,11), np.uint8)
        img_maskA = cv.morphologyEx(img_maskA, cv.MORPH_OPEN, kernel)
```

```
        img_maskA = cv.morphologyEx(img_maskA, cv.MORPH_CLOSE, kernel)


        kernel = np.ones((11,11), np.uint8)
        img_maskB = cv.morphologyEx(img_maskB, cv.MORPH_OPEN, kernel)
        img_maskB = cv.morphologyEx(img_maskB, cv.MORPH_CLOSE, kernel)


        # 마스크 이미지로 원본 이미지에서 범위값에 해당되는 영상 부분을 획득합니다.
        img_maskC = cv.bitwise_or(img_maskA, img_maskB) #두 장의 마스크이미지를 or연산하여 하나로 합친 후
        img_result = cv.bitwise_and(img_color, img_color, mask=img_maskC) #원본 영상과 마스크 이미지를 and연산합니다.


        # 라벨링
        numOfLabelsA, img_labelA, statsA, centroidsA = cv.connectedComponentsWithStats(img_maskA) #첫번째 색 영역을 라벨링하여 영역을 추출

        for idx, centroid in enumerate(centroidsA):
            if statsA[idx][0] == 0 and statsA[idx][1] == 0:
                continue

            if np.any(np.isnan(centroid)):
                continue

            x,y,width,height,area = statsA[idx]
            centerX,centerY = int(centroid[0]), int(centroid[1])

            if area > 1500:
                cv.circle(img_color, (centerX, centerY), 10, (0,0,255), 10)
                cv.rectangle(img_color, (x,y), (x+width,y+height), (0,0,255))


        numOfLabelsB, img_labelB, statsB, centroidsB = cv.connectedComponentsWithStats(img_maskB) #두번째 색 영역을 라벨링하여 영역을 추출
        for idx, centroid in enumerate(centroidsB):
            if statsB[idx][0] == 0 and statsB[idx][1] == 0:
                continue

            if np.any(np.isnan(centroid)):
                continue

            x,y,width,height,area = statsB[idx]
            centerX,centerY = int(centroid[0]), int(centroid[1])

            if area > 1500:
                cv.circle(img_color, (centerX, centerY), 10, (255,0,0), 10)
                cv.rectangle(img_color, (x,y), (x+width,y+height), (255,0,0))


        cv.imshow('img_result', img_result)


    cv.imshow('img_color', img_color)


    key = cv.waitKey(1) & 0xFF

    if key == 27: # esc
        break

    elif key == 32: # space 스페이스바를 누르면
        if step == 0:
            roi = img_color2[cy-20:cy+20, cx-20:cx+20] #입력영역의 픽셀을 가져옵니다.
            roi = cv.medianBlur(roi, 3)
            cv.imshow("roi1", roi)
            hsv = cv.cvtColor(roi, cv.COLOR_BGR2HSV)#hsv 색공간으로 변환하 #
            h,s,v = cv.split(hsv)#Hue 성분의 편균을 구합니다.
            color1 = h.mean()
            print(color1) #첫번째 색을 받는데 사용함.
            step += 1

        elif step == 1:
            roi = img_color2[cy-20:cy+20, cx-20:cx+20]
            roi = cv.medianBlur(roi, 3)
            cv.imshow("roi2", roi)
            hsv = cv.cvtColor(roi, cv.COLOR_BGR2HSV)
            h,s,v = cv.split(hsv)
            color2 = h.mean()
            set_color = True
            nothing(0)
            print(color2)
            step += 1

cap.release()
cv.destroyAllWindows()
```

▼ 5주차

해리스 코너 검출 : 이미지에서 코너점 검출시 사용

- 코너점은 에지와 에지가 만나서 만들어짐
- 평면은 픽셀강도가 없음
- 에지 위에서는 방향으로 픽셀강도가 있음
- 코너에서는 모든방향으로 픽셀 강도가 있음

```
dst = cv.cornerHarris(src, blockSize, ksize, k[, dst[, borderType]])
```

- src : 입력이미지 (float32 타입의 그레이스케일 이미지여야함)
- blockSize : 코너 검출시 고려할 이웃점을 결정하는 윈도우 크기
- ksize : 소벨 연산시 사용되는 커널의 크기입니다.
- k : 경험적 상수0.04~0.06 사이의 값 사용

```
import numpy as np
import cv2 as cv


filename = 'shape.jpg'
img = cv.imread(filename)

gray = cv.cvtColor(img,cv.COLOR_BGR2GRAY)
gray = np.float32(gray)

dst = cv.cornerHarris(gray, 5, 3, 0.04)

#result is dilated for marking the corners, not important
dst = cv.dilate(dst,None)

# Threshold for an optimal value, it may vary depending on the image.
img[dst>0.01*dst.max()]=[0,0,255]

cv.imshow('dst',img)

if cv.waitKey(0) & 0xff == 27:
    cv.destroyAllWindows()
```

도형검출하기

- 이진화된 이미지에서 도형에 대한 컨투어를 검출하고 근사화시켜 도형의 모양을 판정함
- 컨투어를 검출할 도형이 흰색이 되어야 함.

```
import cv2 as cv

#컨투어 영역 내에 텍스트를 출력하는 함수
def setLabel(image, str, contour):
    #주어진 문자열 외곽을 둘러쌀 박스의 너비와 높이
    (text_width, text_height), baseline = cv.getTextSize(str, cv.FONT_HERSHEY_SIMPLEX, 0.7, 1)
    #주어진 컨투어 외곽을 둘러쌀 박스의 위치, 너비, 높이를 계산
    x,y,width,height = cv.boundingRect(contour)
    #컨투어 외곽을 둘러싸는 박스의 정중앙에 텍스트와 텍스트 박스 표시
    pt_x = x+int((width-text_width)/2)
    pt_y = y+int((height + text_height)/2)
    cv.rectangle(image, (pt_x, pt_y+baseline), (pt_x+text_width, pt_y-text_height), (200,200,200), cv.FILLED)
    cv.putText(image, str, (pt_x, pt_y), cv.FONT_HERSHEY_SIMPLEX, 0.7, (0,0,0), 1, 8)


img_color = cv.imread('test.png', cv.IMREAD_COLOR)
cv.imshow('result', img_color)
cv.waitKey(0)

img_gray = cv.cvtColor(img_color, cv.COLOR_BGR2GRAY)
cv.imshow('result', img_gray)
cv.waitKey(0)

ret,img_binary = cv.threshold(img_gray, 127, 255, cv.THRESH_BINARY_INV|cv.THRESH_OTSU)
cv.imshow('result', img_binary)
```

```
cv.waitKey(0)

#컨투어를 검출함 , 외곽에 있는 컨투어 검출, 컨투어의 양 끝점만 저장함.
contours, hierarchy = cv.findContours(img_binary, cv.RETR_EXTERNAL, cv.CHAIN_APPROX_SIMPLE)

for cnt in contours:
    size = len(cnt)
    print(size) #컨투어의 개수 확인

    epsilon = 0.005 * cv.arcLength(cnt, True)
    approx = cv.approxPolyDP(cnt, epsilon, True) #컨투어를 근사화시킴.

    size = len(approx)
    print(size) #근사화한 컨투어의 개수 확인 < 이걸로 도형을 판별함.

    #
    cv.line(img_color, tuple(approx[0][0]), tuple(approx[size-1][0]), (0, 255, 0), 3)
    for k in range(size-1):
        cv.line(img_color, tuple(approx[k][0]), tuple(approx[k+1][0]), (0, 255, 0), 3)

    if cv.isContourConvex(approx): #오목하게 들어간 경우를 제외시킴.
        if size == 3:
            setLabel(img_color, "triangle", cnt)
        elif size == 4:
            setLabel(img_color, "rectangle", cnt)
        elif size == 5:
            setLabel(img_color, "pentagon", cnt)
        elif size == 6:
            setLabel(img_color, "hexagon", cnt)
        elif size == 8:
            setLabel(img_color, "octagon", cnt)
        elif size == 10:
            setLabel(img_color, "decagon", cnt)
        else:
            setLabel(img_color, str(size), cnt)
    else:
        setLabel(img_color, str(size), cnt)

cv.imshow('result', img_color)
cv.waitKey(0)
```

관심영역(ROI)

```
import cv2


mouse_is_pressing = False #마우스 버튼을 누른 상태인지 체크
start_x, starty = -1, -1 #roi 시작점


def mouse_callback(event, x, y, flags, param):
    global start_x, start_y,mouse_is_pressing #전역변수 사용 선언

    img_result = img_color.copy() #원본영상 카피

    if event == cv2.EVENT_LBUTTONDOWN: #roi 시작점 저장

        mouse_is_pressing = True #마우스 버튼 누름.
        start_x, start_y = x, y

        cv2.circle(img_result, (x,y), 10, (0, 255, 0), -1)

        cv2.imshow("img_color", img_result)

    elif event == cv2.EVENT_MOUSEMOVE: #시작점부터 움직인 커서 위치까지 사각형 그림

        if mouse_is_pressing:
            cv2.rectangle(img_result, (start_x, start_y), (x, y), (0, 255, 0), 3)

            cv2.imshow("img_color", img_result)

    elif event == cv2.EVENT_LBUTTONUP: #버튼에서 손을 때면 roi 끝지점 설정

        mouse_is_pressing = False #마우스 버튼 땜.

        img_cat = img_color[start_y:y, start_x:x]
        img_cat = cv2.cvtColor(img_cat, cv2.COLOR_BGR2GRAY); #그레이스케일
        img_cat = cv2.cvtColor(img_cat, cv2.COLOR_GRAY2BGR); #그레이스케일


        img_result[start_y:y, start_x:x] = img_cat #새로운 창을 만듬.
        cv2.imshow("img_color", img_result) #원본 창에 선택한 지점 회색
        cv2.imshow("img_cat", img_cat) #선택한 부분을 새로운 창에 띄움.
```

```
img_color = cv2.imread('cat.jpg', cv2.IMREAD_COLOR )

cv2.imshow("img_color", img_color)
cv2.setMouseCallback('img_color', mouse_callback) #마우스를 통해 이벤트를 발생시키는 부분


cv2.waitKey(0)#키보드에 입력이 있으면 종료

cv2.destroyAllWindows()
```

Canny Edge Detector

- 좋은 에지란?
  1. 낮은 에러율
  2. 적황한 에지 위치
  3. 응답 최소화

- 알고리즘 순서
  1. 노이즈제거

     가우시안 블러를 적용해서 이미지 상의 노이지를 제거하고 상세부분은 단순화
  2. 에지 그레디언트 크기, 방향 계산

     angle = atan2(-gy[x], gx[x] * 180 / pi 엣지 방향검출 식

     gx는 수평방향 수직선 검출

     gy는 수직방향 수평선 검출
  3. Non-maximum Suppression

     이미지 전체를 스캔 엣지가 될 수 없는 픽셀 제거

     높은 픽셀과, 낮은 픽셀 사이에서 엣지가 검출됨
  4. Hysteresis Thresholding

     2개의 쓰레쉬홀드값을 사용하여 이전 단계의 엣지의 수를 줄임.

     그레디언트가 하이스레쉬홀드와 로우 스레쉬 홀드 사이에 있을때 엣지로 적용.

```
import cv2


def nothing():
    pass



img_gray = cv2.imread('house.jpg', cv2.IMREAD_GRAYSCALE)


cv2.namedWindow("Canny Edge")
cv2.createTrackbar('low threshold', 'Canny Edge', 0, 1000, nothing)
cv2.createTrackbar('high threshold', 'Canny Edge', 0, 1000, nothing)

cv2.setTrackbarPos('low threshold', 'Canny Edge', 50)
cv2.setTrackbarPos('high threshold', 'Canny Edge', 150)

cv2.imshow("Original", img_gray)

while True:

    low = cv2.getTrackbarPos('low threshold', 'Canny Edge')
    high = cv2.getTrackbarPos('high threshold', 'Canny Edge')

    img_canny = cv2.Canny(img_gray, low, high)
    cv2.imshow("Canny Edge", img_canny)

    if cv2.waitKey(1)&0xFF == 27:
        break
```

```
cv2.destroyAllWindows()
```

라벨링

- 영상을 이진화영상으로 만든 뒤 흰색의 영역마다 정수를 부여하는 것
- 영상 → 그레이스케일 → 케니 연산자 이용 엣지 검출 → 이미지 반전(이진화) → 컨투어를 사용하여 이미지 보강 → 라벨링

```
# 라벨링 함수 : 입력이미지만 넣으면 됨.
retval, labels, stats, centroids = cv2.connectedComponentsWithStats(image)
```

```
import cv2


# 0
image = cv2.imread("test.png", cv2.IMREAD_COLOR) #이미지를 컬러로 가져옴.
cv2.imshow("result", image)
cv2.waitKey(0)

blurred = cv2.GaussianBlur(image, (5, 5), 0) #이미지에 가우시안 블러 적용

gray = cv2.cvtColor(blurred, cv2.COLOR_BGR2GRAY) #그레이스케일 이미지로 바꿈
cv2.imshow("result", gray)
cv2.waitKey(0)

edge = cv2.Canny(gray, 50, 150) #그레이스케일 이미지로부터 엣지 검출
cv2.imshow("result", edge )
cv2.waitKey(0)


# 1
edge = cv2.bitwise_not(edge) #영상을 반전함
cv2.imshow("result", edge )
cv2.waitKey(0)


#컨투어 추출
contours = cv2.findContours(edge.copy(), cv2.RETR_LIST, cv2.CHAIN_APPROX_SIMPLE)
cv2.drawContours(edge, contours[0], -1, (0, 0, 0), 1)
cv2.imshow("result", edge )
cv2.waitKey(0)


# 2 라벨링하여 영역에 번호를 부여함
nlabels, labels, stats, centroids = cv2.connectedComponentsWithStats(edge)


for i in range(nlabels):

  if i < 2:
    continue
  #컨투어별로 넓이, 중심좌표, 영영박스 좌표등을 가져옴.
  area = stats[i, cv2.CC_STAT_AREA]
  center_x = int(centroids[i, 0])
  center_y = int(centroids[i, 1])
  left = stats[i, cv2.CC_STAT_LEFT]
  top = stats[i, cv2.CC_STAT_TOP]
  width = stats[i, cv2.CC_STAT_WIDTH]
  height = stats[i, cv2.CC_STAT_HEIGHT]

  #영역넓이가 50보다 크면 사각형을 그리고 중심에 원을 그린후 라벨 값 출력
  if area > 50:
    cv2.rectangle(image, (left, top), (left + width, top + height),
        (0, 0, 255), 1)
    cv2.circle(image, (center_x, center_y), 5, (255, 0, 0), 1)
    cv2.putText(image, str(i), (left + 20, top + 20),
        cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 0, 0), 2);


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

템플릿 매칭

- 템플릿 이미지와 일치하는 부분을 이미지에서 찾는 기법

- 이미지의 모든 좌표에서 템플릿 이미지 크기의 영역에 있는 픽셀들과 템플릿 이미지의 픽셀과의 차이를 구하여 절대값을 취한 후 모두 더 함.(SAD)
- 이미지의 모든 위치에서 계산한 SAD 중 가장 낮은 값을 찾으면 일치하는 부분임.
- 2개 이상 찾으려면 threshold를 이용해야함. sad값이 threshold 작은 곳

```
import cv2
import numpy as np


img_color = cv2.imread('image.jpg', cv2.IMREAD_COLOR)
img_original = cv2.cvtColor(img_color, cv2.COLOR_BGR2GRAY) #템플릿 찾을 이미지

img_template = cv2.imread('template.jpg', cv2.IMREAD_GRAYSCALE) #템플릿 이미지

#이미지의 너비와 높이를 가져옴
original_h, original_w = img_original.shape[:2]
template_h, template_w = img_template.shape[:2]

#최소 SAD 지점을 저장하기 변수
best_position_y = 0
best_position_x = 0
best_position_sad = 100000


#모든 좌표에서의 SAD를 저장할 리스트
point = []

# 원본 이미지 스캔
for original_x in range(original_w - template_w):
    for original_y in range(original_h - template_h):

        SAD = 0
        # #템플릿 이미지 스캔
        # for template_x in range(template_w):
        #     for template_y in range(template_h):

        #         original_pixel = img_original[template_y + original_y, template_x + original_x]
        #         template_pixel = img_template[template_y, template_x]

        #         SAD += abs( original_pixel - template_pixel)

        #원본 이미지에서 템플릿크기만큼 ROI를 구함.
        original_pixel = img_original[original_y:original_y + template_h , original_x:original_x + template_w].ravel()
        template_pixel = img_template.ravel()

        #넘파이의 벡터라이제이션을 사용
        SAD = np.abs(np.subtract(original_pixel, template_pixel, dtype=np.float))
        SAD = SAD.sum()

        # 최소 SAD 지점 찾기
        if best_position_sad > SAD:

            best_position_y = original_y
            best_position_x = original_x
            best_position_sad = SAD
        point.append((original_x, original_y, SAD))


#리스트에서 최소 SAD와 차이가 100이하인 좌표에만 사각형을 그림
for p in point:
    if np.abs(p[2] - best_position_sad) < 100:
        print(p[2])
        cv2.rectangle(img_color, (p[0], p[1]), (p[0]+template_w, p[1]+template_h), (255, 0, 0), 3 )

cv2.imshow('result', img_color)
cv2.waitKey(0)
```

OpenCV에서 픽셀에 접근하는 방법

- 픽셀 : 화면을 구성하는 기본 단위
- 컬러이미지 : RGB로 구성, 3가지 채널로 구성, 0~255로 구성됨.
- 그레이스케일 : RGB/3을 하여 하나의 채널로 구성

```
import cv2
import numpy as np
```

```
img_color = cv2.imread("image.jpg", cv2.IMREAD_COLOR)

#원본이미지와 같은 크기의 넘파이 배열을 생성함
height, width, channel = img_color.shape
img_gray = np.zeros((height, width), np.uint8)

for y in range(0, height):
    for x in range(0, width):
        # 컬러 영상의 경우 픽셀값 읽어오기
        b = img_color.item(y, x, 0)
        g = img_color.item(y, x, 1)
        r = img_color.item(y, x, 2)


        gray = (int(b) + int(g) + int(r)) / 3.0

        # 그레이스케일의 경우 픽셀값 저장하기
        img_gray.itemset(y, x, gray)


cv2.imshow('bgr', img_color)
cv2.imshow('gray', img_gray)

cv2.waitKey(0)

cv2.destroyAllWindows()
```

손 검출하기(색기반)

```
import cv2 as cv
import numpy as np
import os



#OpenCV cascasde를 이용하여 사용자의 얼굴 부분을 검정색으로 만듬.
def detect(img, cascade):
    rects = cascade.detectMultiScale(img, scaleFactor=1.1, minNeighbors=5, minSize=(30, 30),
                                     flags=cv.CASCADE_SCALE_IMAGE)
    if len(rects) == 0:
        return []
    rects[:,2:] += rects[:,:2]
    return rects


def removeFaceAra(img, cascade):
  gray = cv.cvtColor(img, cv.COLOR_BGR2GRAY)
  gray = cv.equalizeHist(gray)
  rects = detect(gray, cascade)

  height,width = img.shape[:2]

  for x1, y1, x2, y2 in rects:
      cv.rectangle(img, (x1-10, 0), (x2+10, height), (0,0,0), -1)

  return img


#HSV 색공간에서 H와 S 성분을 조절하여 살색영역을 찾음.
def make_mask_image(img_bgr):

  img_hsv = cv.cvtColor(img_bgr, cv.COLOR_BGR2HSV)

  #img_h,img_s,img_v = cv.split(img_hsv)

  low = (0, 30, 0)
  high = (15, 255, 255)

  img_mask = cv.inRange(img_hsv, low, high)
  return img_mask


def distanceBetweenTwoPoints(start, end):

  x1,y1 = start
  x2,y2 = end

  return int(np.sqrt(pow(x1 - x2, 2) + pow(y1 - y2, 2)))


def calculateAngle(A, B):

  A_norm = np.linalg.norm(A)
```

```
  B_norm = np.linalg.norm(B)
  C = np.dot(A,B)

  angle = np.arccos(C/(A_norm*B_norm))*180/np.pi
  return angle

#검출된 컨투어 영역에서 가장 큰영역을 찾음.
def findMaxArea(contours):

  max_contour = None
  max_area = -1


  for contour in contours:
    area = cv.contourArea(contour)

    x,y,w,h = cv.boundingRect(contour)

    if (w*h)*0.4 > area:
        continue

    if w > h:
        continue

    if area > max_area:
      max_area = area
      max_contour = contour

  if max_area < 10000:
    max_area = -1

  return max_area, max_contour

#컨투어에 대한 컨벡스홀을 계산하여 선의 방향이 바뀌는 부분을 손가락 후보로 함.
def getFingerPosition(max_contour, img_result, debug):
  points1 = []


  # STEP 6-1
  M = cv.moments(max_contour)

  cx = int(M['m10']/M['m00'])
  cy = int(M['m01']/M['m00'])


  max_contour = cv.approxPolyDP(max_contour,0.02*cv.arcLength(max_contour,True),True)
  hull = cv.convexHull(max_contour)

  for point in hull:
    if cy > point[0][1]:
      points1.append(tuple(point[0]))

  if debug:
    cv.drawContours(img_result, [hull], 0, (0,255,0), 2)
    for point in points1:
      cv.circle(img_result, tuple(point), 15, [ 0, 0, 0], -1)


  # STEP 6-2 컨투어를 근사화하고 디펙트하여 손가락을 검출함.
  hull = cv.convexHull(max_contour, returnPoints=False)
  defects = cv.convexityDefects(max_contour, hull)

  if defects is None:
    return -1,None

  points2=[]
  for i in range(defects.shape[0]):
    s,e,f,d = defects[i, 0]
    start = tuple(max_contour[s][0])
    end = tuple(max_contour[e][0])
    far = tuple(max_contour[f][0])

    angle = calculateAngle( np.array(start) - np.array(far), np.array(end) - np.array(far))

    if angle < 90:
      if start[1] < cy:
        points2.append(start)

      if end[1] < cy:
        points2.append(end)

  if debug:
    cv.drawContours(img_result, [max_contour], 0, (255, 0, 255), 2)
    for point in points2:
      cv.circle(img_result, tuple(point), 20, [ 0, 255, 0], 5)
```

```
  # STEP 6-3 컨벡스홀을 통해 찾은 손가락 후보와 디펙트를 통해 찾은 손가락 후보를 합침.
  points = points1 + points2
  points = list(set(points))


  # STEP 6-4 좌우에 있는 엣지가 90도인 경우에만 손가락으로 인식하게 만듬.
  new_points = []
  for p0 in points:

    i = -1
    for index,c0 in enumerate(max_contour):
      c0 = tuple(c0[0])

      if p0 == c0 or distanceBetweenTwoPoints(p0,c0)<20:
        i = index
        break

    if i >= 0:
      pre = i - 1
      if pre < 0:
        pre = max_contour[len(max_contour)-1][0]
      else:
        pre = max_contour[i-1][0]

      next = i + 1
      if next > len(max_contour)-1:
        next = max_contour[0][0]
      else:
        next = max_contour[i+1][0]


      if isinstance(pre, np.ndarray):
            pre = tuple(pre.tolist())
      if isinstance(next, np.ndarray):
        next = tuple(next.tolist())


      angle = calculateAngle( np.array(pre) - np.array(p0), np.array(next) - np.array(p0))

      if angle < 90:
        new_points.append(p0)

  return 1,new_points


def process(img_bgr, debug):

  img_result = img_bgr.copy()

  # STEP 1
  img_bgr = removeFaceAra(img_bgr, cascade)


  # STEP 2
  img_binary = make_mask_image(img_bgr)


  # STEP 3 검출된 살색영역에 검은 구멍이 있는 경우를 없애기 위한 코드
  kernel = cv.getStructuringElement(cv.MORPH_ELLIPSE, (5, 5))
  img_binary = cv.morphologyEx(img_binary, cv.MORPH_CLOSE, kernel, 1)
  cv.imshow("Binary", img_binary)


  # STEP 4 바이너리 이미지의 흰색 외곽을 따라 컨투어를 표시함.
  contours, hierarchy = cv.findContours(img_binary, cv.RETR_EXTERNAL, cv.CHAIN_APPROX_SIMPLE)

  if debug:
    for cnt in contours:
      cv.drawContours(img_result, [cnt], 0, (255, 0, 0), 3)


  # STEP 5
  max_area, max_contour = findMaxArea(contours)

  if max_area == -1:
    return img_result

  if debug:
    cv.drawContours(img_result, [max_contour], 0, (0, 0, 255), 3)


  # STEP 6
  ret,points = getFingerPosition(max_contour, img_result, debug)


  # STEP 7
  if ret > 0 and len(points) > 0:
```

```
    for point in points:
      cv.circle(img_result, point, 20, [ 255, 0, 255], 5)

  return img_result



current_file_path = os.path.dirname(os.path.realpath(__file__))
cascade = cv.CascadeClassifier(cv.samples.findFile(current_file_path + "\haarcascade_frontalface_alt.xml"))


# cap = cv.VideoCapture('test.avi')

cap = cv.VideoCapture(0)

while True:

  ret,img_bgr = cap.read()

  if ret == False:
    break

  img_result = process(img_bgr, debug=False)

  key = cv.waitKey(1)
  if key== 27:
      break


  cv.imshow("Result", img_result)


cap.release()
cv.destroyAllWindows()
```

명함검출 40분 넘어서 순서만 정리

1. 이진화
2. 외곽선 검출 & 다각형 근사화
3. 투영 변환
4. OCR

▼ 6주차

책 검출

```
mport numpy as np
import cv2


step = 0
mouse_is_pressing = False


def distanceBetweenTwoPoints(point1, point2):

  x1,y1 = point1
  x2,y2 = point2

  return int(np.sqrt(pow(x1 - x2, 2) + pow(y1 - y2, 2)))


def mouse_callback(event,x,y,flags,param):

    global mouse_is_pressing,points

    if step != 1:
      return

    if event == cv2.EVENT_MOUSEMOVE:
        if mouse_is_pressing == True:

          for i,point in enumerate(points):
            if distanceBetweenTwoPoints((x,y), point) < 15:
              points[i][0] = x
              points[i][1] = y
              break

    elif event == cv2.EVENT_LBUTTONDOWN:
```

```
      for point in points:
        if distanceBetweenTwoPoints((x,y), point) < 10:
          mouse_is_pressing = True
          break

    elif event == cv2.EVENT_LBUTTONUP:

      mouse_is_pressing = False


def angle_between(v0, v1):

  angle = np.math.atan2(np.linalg.det([v0,v1]),np.dot(v0,v1))

  return np.degrees(angle)


def sort_points(points):

    points = points.astype(np.float32)

    rect = np.zeros((4, 2), dtype = "float32")

    # sort : top left, top right, bottom right, bottom left
    s = points.sum(axis = 1)
    min_index = np.argmin(s)
    rect[0] = points[min_index]
    points = np.delete(points, min_index, axis = 0)

    s = points.sum(axis = 1)
    max_index = np.argmax(s)
    rect[2] = points[max_index]
    points = np.delete(points, max_index, axis = 0)

    v0 = points[0] - rect[0]
    v1 = points[1] - rect[0]

    angle = angle_between(v0, v1)

    if angle < 0:
        rect[1] = points[1]
        rect[3] = points[0]
    else:
        rect[1] = points[0]
        rect[3] = points[1]

    return rect


def transform(img_input, points):

    points = sort_points(points)
    topLeft, topRight, bottomRight, bottomLeft = points

    topWidth = distanceBetweenTwoPoints(bottomLeft, bottomRight)
    bottomWidth = distanceBetweenTwoPoints(topLeft, topRight)
    maxWidth = max(int(topWidth), int(bottomWidth))

    leftHeight = distanceBetweenTwoPoints(topLeft, bottomLeft)
    rightHeight = distanceBetweenTwoPoints(topRight, bottomRight)
    maxHeight = max(int(leftHeight), int(rightHeight))

    # top left, top right, bottom right, bottom left
    dst = np.array([[0, 0],[maxWidth - 1, 0],
        [maxWidth - 1, maxHeight - 1],[0, maxHeight - 1]], dtype = "float32")


    H = cv2.getPerspectiveTransform(points, dst)
    img_warped = cv2.warpPerspective(img_input, H, (maxWidth, maxHeight))

    return img_warped


def findMaxArea(contours):

  max_contour = None
  max_area = -1


  for contour in contours:
    area = cv2.contourArea(contour)

    x,y,w,h = cv2.boundingRect(contour)

    if area > max_area:
      max_area = area
```

```
      max_contour = contour


  return max_area, max_contour


def process(img_input, debug):

  points = []
  height,width =img_input.shape[:2]

  # Step 1 Grab Cut을 사용하여 책 주변만 남기고 배경을 제거함.
  img_mask = np.zeros(img_input.shape[:2],np.uint8)

  bgdModel = np.zeros((1,65),np.float64)
  fgdModel = np.zeros((1,65),np.float64)

  rect = (10,10,width-30,height-30)
  cv2.grabCut(img_input, img_mask, rect, bgdModel,fgdModel,3,cv2.GC_INIT_WITH_RECT)
  img_mask = np.where((img_mask==2)|(img_mask==0), 0, 1).astype('uint8')
  img = img_input*img_mask[:,:,np.newaxis]

  background = img_input - img

  background[np.where((background >= [0,0,0]).all(axis = 2))] = [0,0,0]

  img_grabcut = background + img

  if debug:
    cv2.imshow('grabCut', img_grabcut)


  # Step 2 그레이스케일로 변환한 후 캐니 에지 티텍션을 사용하여 엣지를 검출함
  img_gray = cv2.cvtColor(img_grabcut, cv2.COLOR_BGR2GRAY);
  img_canny = cv2.Canny(img_gray, 30, 90);

  if debug:
    cv2.imshow('Canny', img_canny)


  # Step 3 끊어진 에지를 이어주기 위해 모폴로지 close 연산을 수행함
  kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (5, 5))
  img_canny = cv2.morphologyEx(img_canny, cv2.MORPH_CLOSE, kernel, 1)

  if debug:
    cv2.imshow('morphology', img_canny)


  # Step 4 캐니 에지 결과에서 컨투어를 찾아 가장 큰 영역을 차지하는 컨투어를 찾음
  contours, hierarchy = cv2.findContours(img_canny, cv2.RETR_LIST, cv2.CHAIN_APPROX_SIMPLE)
  max_area, max_contour = findMaxArea(contours)
  if max_area < 0:
    return points

  if debug:
    img_contour = img_input.copy()
    cv2.drawContours(img_contour, [max_contour], 0, (0, 0, 255), 3)
    cv2.imshow('Contour', img_contour)


  # Step 5 컨투어를 근사화하고 컨벡스 홀을 구함.
  max_contour = cv2.approxPolyDP(max_contour,0.02*cv2.arcLength(max_contour,True),True)
  hull = cv2.convexHull(max_contour)

  if debug:
    img_convexhull = img_input.copy()
    cv2.drawContours(img_convexhull, [hull], 0, (255,255,0), 5)
    cv2.imshow('convexHull', img_convexhull)


  # Step 6 컨벡스 홀로 부터 4개의 코너점을 계산함.
  size = len(max_contour)

  if size == 4:
    for c in hull:
      points.append(tuple(c[0].tolist()))
    points=np.array(points)
  else:
    rect = cv2.minAreaRect(hull)
    box = cv2.boxPoints(rect)
    points = np.int0(box.tolist())

  found = False
  for p in points:
    if p[0] < 0 or p[0] > width-1 or p[1] < 0 or p[1] > height -1:
      found = True
      break
```

```
  if found:
    points = np.array([[10,10], [width-11, 10], [width-11, height-11], [10, height-11]])

  return points



img_input = cv2.imread('9.jpg')
height, width = img_input.shape[:2]

points = process(img_input, debug=False)

size = len(points)

if size > 0:
  cv2.namedWindow('input')
  cv2.setMouseCallback("input", mouse_callback, 0);

  step = 1


  while True:

    img_result = img_input.copy()
    for point in points:
        cv2.circle(img_result, tuple(point), 10, (255,0,0), 3 )
    cv2.imshow('input', img_result)

    key = cv2.waitKey(1)
    if key == 32:
      break


  img_final = transform(img_input, points )


  cv2.imshow('input', img_result)
  cv2.imshow('result', img_final )

else:
  cv2.imshow('input', img_input)

cv2.waitKey(0)

cv2.destroyAllWindows()
```

손 검출하기(Background Subtraction)

- 일정시간 배경을 촬영한 후, 이후 들어오는 물체들을 흰색 영역으로 검출해줌.

```
import cv2 as cv
import numpy as np
import os


def detect(img, cascade):
    rects = cascade.detectMultiScale(img, scaleFactor=1.1, minNeighbors=5, minSize=(30, 30),
                                     flags=cv.CASCADE_SCALE_IMAGE)
    if len(rects) == 0:
        return []
    rects[:,2:] += rects[:,:2]
    return rects

def removeFaceAra(img, cascade):
  gray = cv.cvtColor(img, cv.COLOR_BGR2GRAY)
  gray = cv.equalizeHist(gray)
  rect = detect(gray, cascade)

  return rect

def findMaxArea(contours):

  max_contour = None
  max_area = -1


  for contour in contours:
    area = cv.contourArea(contour)

    x,y,w,h = cv.boundingRect(contour)

    if (w*h)*0.4 > area:
```

```
        continue

    if w > h:
        continue

    if area > max_area:
      max_area = area
      max_contour = contour

  if max_area < 10000:
    max_area = -1

  return max_area, max_contour


def distanceBetweenTwoPoints(start, end):

  x1,y1 = start
  x2,y2 = end

  return int(np.sqrt(pow(x1 - x2, 2) + pow(y1 - y2, 2)))


def calculateAngle(A, B):

  A_norm = np.linalg.norm(A)
  B_norm = np.linalg.norm(B)
  C = np.dot(A,B)

  angle = np.arccos(C/(A_norm*B_norm))*180/np.pi
  return angle


def getFingerPosition(max_contour, img_result, debug):
  points1 = []


  # STEP 6-1
  M = cv.moments(max_contour)

  cx = int(M['m10']/M['m00'])
  cy = int(M['m01']/M['m00'])


  max_contour = cv.approxPolyDP(max_contour,0.02*cv.arcLength(max_contour,True),True)
  hull = cv.convexHull(max_contour)

  for point in hull:
    if cy > point[0][1]:
      points1.append(tuple(point[0]))

  if debug:
    cv.drawContours(img_result, [hull], 0, (0,255,0), 2)
    for point in points1:
      cv.circle(img_result, tuple(point), 15, [ 0, 0, 0], -1)


  # STEP 6-2
  hull = cv.convexHull(max_contour, returnPoints=False)
  defects = cv.convexityDefects(max_contour, hull)

  if defects is None:
    return -1,None

  points2=[]
  for i in range(defects.shape[0]):
    s,e,f,d = defects[i, 0]
    start = tuple(max_contour[s][0])
    end = tuple(max_contour[e][0])
    far = tuple(max_contour[f][0])

    angle = calculateAngle( np.array(start) - np.array(far), np.array(end) - np.array(far))

    if angle < 90:
      if start[1] < cy:
        points2.append(start)

      if end[1] < cy:
        points2.append(end)

  if debug:
    cv.drawContours(img_result, [max_contour], 0, (255, 0, 255), 2)
    for point in points2:
      cv.circle(img_result, tuple(point), 20, [ 0, 255, 0], 5)


  # STEP 6-3
```

```
  points = points1 + points2
  points = list(set(points))


  # STEP 6-4
  new_points = []
  for p0 in points:

    i = -1
    for index,c0 in enumerate(max_contour):
      c0 = tuple(c0[0])

      if p0 == c0 or distanceBetweenTwoPoints(p0,c0)<20:
        i = index
        break

    if i >= 0:
      pre = i - 1
      if pre < 0:
        pre = max_contour[len(max_contour)-1][0]
      else:
        pre = max_contour[i-1][0]

      next = i + 1
      if next > len(max_contour)-1:
        next = max_contour[0][0]
      else:
        next = max_contour[i+1][0]


      if isinstance(pre, np.ndarray):
        pre = tuple(pre.tolist())
      if isinstance(next, np.ndarray):
        next = tuple(next.tolist())


      angle = calculateAngle( np.array(pre) - np.array(p0), np.array(next) - np.array(p0))

      if angle < 90:
        new_points.append(p0)

  return 1,new_points

def process(img_bgr, img_binary, debug):

  img_result = img_bgr.copy()

  # # STEP 1
  # img_bgr = removeFaceAra(img_bgr, cascade)


  # # STEP 2
  # img_binary = make_mask_image(img_bgr)


  # # STEP 3
  # kernel = cv.getStructuringElement(cv.MORPH_ELLIPSE, (5, 5))
  # img_binary = cv.morphologyEx(img_binary, cv.MORPH_CLOSE, kernel, 1)
  # if debug:
  #   cv.imshow("Binary", img_binary)


  # STEP 4
  contours, hierarchy = cv.findContours(img_binary, cv.RETR_EXTERNAL, cv.CHAIN_APPROX_SIMPLE)

  if debug:
    for cnt in contours:
      cv.drawContours(img_result, [cnt], 0, (255, 0, 0), 3)


  # STEP 5
  max_area, max_contour = findMaxArea(contours)

  if max_area == -1:
    return img_result

  if debug:
    cv.drawContours(img_result, [max_contour], 0, (0, 0, 255), 3)


  # STEP 6
  ret,points = getFingerPosition(max_contour, img_result, debug)


  # STEP 7
  if ret > 0 and len(points) > 0:
    for point in points:
```

```
      cv.circle(img_result, point, 20, [ 255, 0, 255], 5)

  return img_result



current_file_path = os.path.dirname(os.path.realpath(__file__))
cascade = cv.CascadeClassifier(cv.samples.findFile(current_file_path + "\haarcascade_frontalface_alt.xml"))
cap = cv.VideoCapture('hand.avi')


#  http://layer0.authentise.com/segment-background-using-computer-vision.html
fgbg = cv.createBackgroundSubtractorMOG2(varThreshold=200, detectShadows=0)

index = 0

while(1):
    index = index + 1

    ret, frame = cap.read()
    if ret == False:
        break;

    frame = cv.flip(frame, 1)

    blur = cv.GaussianBlur(frame, (5,5), 0)
    rect = removeFaceAra(frame, cascade)

    fgmask = fgbg.apply(blur, learningRate=0)

    kernel = cv.getStructuringElement(cv.MORPH_ELLIPSE, (5, 5))
    fgmask = cv.morphologyEx(fgmask, cv.MORPH_CLOSE, kernel, 2)


    height,width = frame.shape[:2]
    for x1, y1, x2, y2 in rect:
        cv.rectangle(fgmask, (x1-10, 0), (x2+10, height), (0,0,0), -1)


    img_result = process(frame, fgmask, debug=False)


    cv.imshow('mask', fgmask)
    cv.imshow('result', img_result)


    key = cv.waitKey(30) & 0xff
    if key == 27:
        break

cap.release()
cv.destroyAllWindows()
```

트럼프 카드 인식

```
import cv2 as cv
import numpy as np



# top left, top right, bottom right, bottom left
def sort_points(points):

    points = points.astype(np.float32)


    new_points = np.zeros((4, 2), dtype = "float32")


    s = points.sum(axis = 1)
    min_index = np.argmin(s)
    new_points[0] = points[min_index]
    points = np.delete(points, min_index, axis = 0)


    s = points.sum(axis = 1)
    max_index = np.argmax(s)
    new_points[2] = points[max_index]
    points = np.delete(points, max_index, axis = 0)


    if points[0][1] > points[1][1]:
        new_points[1] = points[1]
```

```
        new_points[3] = points[0]
    else:
        new_points[1] = points[0]
        new_points[3] = points[1]

    return new_points



def transform(img_input, points):

    points = sort_points(points)
    topLeft, topRight, bottomRight, bottomLeft = points

    maxWidth = 400
    maxHeight = 600


    dst = np.array([[0, 0],[maxWidth - 1, 0],
        [maxWidth - 1, maxHeight - 1],[0, maxHeight - 1]],
        dtype = "float32")


    H = cv.getPerspectiveTransform(points, dst)
    img_warped = cv.warpPerspective(img_input, H, (maxWidth, maxHeight))

    return img_warped



#1 사진을 불러옴
img_color = cv.imread('card.jpg', cv.IMREAD_COLOR)


height,width =img_color.shape[:2]
cv.imshow('@', img_color)
cv.waitKey(0)

#2 그레이스케일로 변환
img_gray = cv.cvtColor(img_color, cv.COLOR_BGR2GRAY)
cv.imshow('@', img_gray)
cv.waitKey(0)

#3 이진화를 통해 트럼프와 바닥을 분리
ret, img_binary = cv.threshold(img_gray, 0, 255,
    cv.THRESH_BINARY|cv.THRESH_OTSU)
cv.imshow('@', img_binary)

#4 모폴로지 연산을 통해 노이즈 제거
kernel = cv.getStructuringElement( cv.MORPH_RECT, ( 3, 3 ) )
img_binary = cv.morphologyEx(img_binary, cv.MORPH_OPEN, kernel)
cv.imshow('@', img_binary)
cv.waitKey(0)


#5 컨투어를 검출함
contours, hierarchy = cv.findContours(img_binary, cv.RETR_EXTERNAL,
        cv.CHAIN_APPROX_SIMPLE)

#6 컨투어의 영역을 계산 후 100이하면 제외
for contour in contours:

    area = cv.contourArea(contour)

    if area < 100:
        continue

    #7 근사화
    epsilon = 0.02 * cv.arcLength(contour, True)
    approx = cv.approxPolyDP(contour, epsilon, True)


    size = len(approx)


    #8 컨투어를 그림
    img_result = img_color.copy()
    cv.drawContours(img_result, [approx], -1, (0, 255, 0), 2);
    cv.imshow('@', img_result)
    cv.waitKey(0)

    #9 컨투어를 근사화한 결과가 4개의 엣지라면 컨벡스홀을 구한후 리스트에 저장
    if cv.isContourConvex(approx):
        if size == 4:
            hull = cv.convexHull(approx)
```

```
            points = []

            for p in hull:
                points.append(p[0])
            points = np.array(points)

            #10 정명에서 바라본 트럼프카드로 변환
            img_card = transform(img_color, points )
            img_gray = cv.cvtColor(img_card, cv.COLOR_BGR2GRAY) #템플릿 매칭을 위해 그레이스케일로변환

            #11 저장해둔 템플릿 이미지를 가져옴
            max = -1
            max_idx = -1
            for i in range(1,5):
                img_template = cv.imread( str(i) + '.jpg', cv.IMREAD_GRAYSCALE)

                #12 템플릿 매칭을 통하여 가장 유사한 이미지를 찾음.
                res = cv.matchTemplate(img_gray, img_template, cv.TM_CCOEFF)
                min_val, max_val, min_loc, max_loc = cv.minMaxLoc(res)

                if max < max_val:
                    max = max_val
                    max_idx = i

            #13 가장 유사한 이미지를 보여줌.
            img_template = cv.imread( str(max_idx) + '.jpg', cv.IMREAD_GRAYSCALE)
            img_card = cv.hconcat([img_gray, img_template])

            cv.imshow('Card', img_card)
            cv.waitKey(0)
```

OpenCV 기반으로 파란공 트래킹하여 그림 그리기

- 이건 영상을 보는게 훨씬 빠름.

▼ 7주차

화면 캡쳐한 결과를 템플릿 매칭

```
# pip install pyautogui
# pip install opencv-python
import cv2 as cv
import numpy as np
import pyautogui



cv.namedWindow("result");
cv.moveWindow("result", 0, 500);

img_piece = cv.imread('dino.png', cv.IMREAD_COLOR)
h,w = img_piece.shape[:2]


while 1:
    #캡쳐한 이미지를 가져옴.
    pic = pyautogui.screenshot(region=(0, 0, 700, 500)) #x좌표, y좌표, 너비, 높이
    img_frame = np.array(pic) #넘파이 배열로 변환
    img_frame  = cv.cvtColor(img_frame, cv.COLOR_RGB2BGR) #RGB를 BGR로 변경

    #템플릿 메소드 지정
    meth = 'cv.TM_CCOEFF'
    method = eval(meth)

    #캡처 영상에서 공룡 이미지를 찾음
    res = cv.matchTemplate(img_piece, img_frame, method)
    min_val, max_val, min_loc, max_loc = cv.minMaxLoc(res)#템플릿결과로부터 경계사각형 계산
    top_left = max_loc
    bottom_right = (top_left[0] + w, top_left[1] + h)

    #찾은 공룡 위치에 초록색 사각형을 그려줌
    cv.rectangle(img_frame, top_left, bottom_right, (0, 255, 0), 2)
    print(max_val, top_left)

    cv.imshow('result', img_frame)

    key = cv.waitKey(1)
    if key == 27:
        break
```

DLIB 사용 얼굴 랜드마크 검출

스티칭 구현하기

흑백 사진을 컬러 사진으로 변환

이미지에서 텍스트 영역을 찾아주는 MSER예제

OpenPose를 사용하여 손가락 인식하는 예제