OpenCV

1. OpenCV를 활용한 영상 속 색 분석

코드는 아래와 같다.

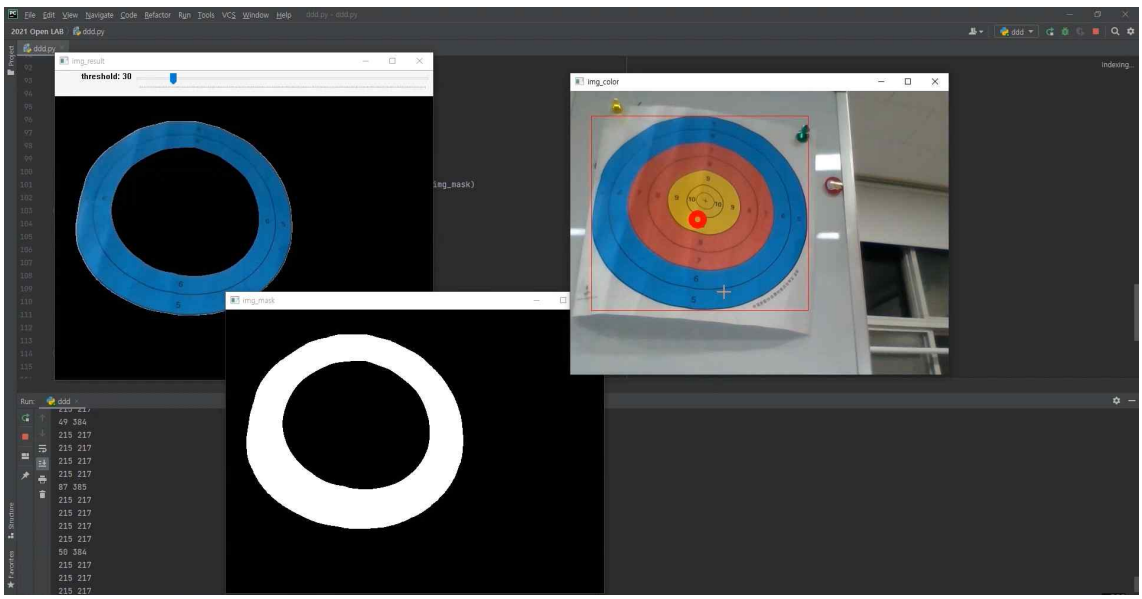
```
import cv2 as cv
import numpy as np

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 변환
    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 픽셀값의 범
위
        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 변환
    img_hsv = cv.cvtColor(img_color, cv.COLOR_BGR2HSV)

    # 범위 값으로 HSV 이미지에서 마스크 생성
    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 부분 획득
    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()
```

- 실제 작동 화면



Sympy

1) sympy 실습

sympy는 CAS(Computer Algebra System, 컴퓨터 대수 시스템)를 파이썬에서 구현할 수 있는 모듈이다. sympy를 이용하면 고차방정식, 연립방정식 등을 대수 표현을 가지고 연산할 수 있다. 본 코드는 Pycham community edition 2020.1에서 실행한 코드이다.

1. 변수 정의 - symbols()를 이용하여 문자를 변수로 정의

```
x, y, z = symbols('x y z')
```

2. 다항식 계산 - 기본적으로 식을 간단히 후 출력한다.

```
from sympy import *
x= symbols('x')
print(x**2+1-2*x**2)
결과: 1 - x**2
```

3. 식 간단히 하기 - 일반적인 다항식이 아닌 경우, simplify()를 이용하여 식을 정리한다.

```
from sympy import *
x= symbols('x')
print(simplify( cos(x)**2 + sin(x)**2 ))
결과: 1
```

4. 방정식 계산 - solve()이용

```
from sympy import *
x= symbols('x')
print(solve(2*x**2+3*x+4,x,dict=True))
pprint(solve(2*x**2+3*x+4,x,dict=True))
결과: [{x: -3/4 - sqrt(23)*I/4}, {x: -3/4 + sqrt(23)*I/4}]
⎡⎛      3   √23⋅ⅈ⎞  ⎛      3   √23⋅ⅈ⎞⎤
⎢⎨ x: - ─ - ─────⎬, ⎨ x: - ─ + ─────⎬⎥
⎣⎝      4     4  ⎠  ⎝      4     4  ⎠⎦
```

print()는 그냥 결과를 출력하지만 pprint()는 교과서식 표현(LaTeX)로 나타나게 된다.

5. 연립방정식 계산 - solve()이용

```
from sympy import *
x,y= symbols('x y')
eq1 = x+2*y-3
eq2 = 3*x-y+2
print(solve((eq1,eq2),dict=True))
결과: [{x: -1/7, y: 11/7}]
```

Math&Turtle

주제:태양,행성,지구간 거리 자료와 일출시간과 행성이 뜨는시간 간 비교 자료를 이용하여 지구에 대한 내행성의 회합 공전 궤도와 회합 주기, 이각을 알아내기

코드

```
import math
import turtle

Sun=[0,100]
Earth=[0,0]


def get_earthelo(sunrise,planrise):##태양 뜨는 시간과 행성 뜨는 시간 차이=>이각 구하기
  sun_time=list(sunrise)
  if sun_time.index(":") == 2:
    sun_hour=int(sun_time[0] + sun_time[1])
    sun_min=int(sun_time[3] + sun_time[4])
  else:
    sun_hour=int(sun_time[0])
    sun_min=int(sun_time[2]+sun_time[3])

  plan_time=list(planrise)
  if plan_time.index(":") == 2:
    plan_hour=int(plan_time[0] + plan_time[1])
    plan_min=int(plan_time[3] + plan_time[4])
  else:
    plan_hour=int(plan_time[0])
    plan_min=int(plan_time[2]+plan_time[3])

  return (plan_hour - sun_hour) * 15 + (plan_min - sun_min) / 4

def draw_rot():##엑셀 자료 개수만큼 점 그리는 과정
  turtle.penup()
  turtle.color('black')
  turtle.goto(0,0)
  turtle.pendown()
  turtle.dot(10, "blue")##지구
  turtle.penup()
  turtle.goto(0,100)
  turtle.pendown()
  turtle.dot(30, "red")##태양
  turtle.penup()
  turtle.goto(0,0)
  turtle.pendown()
  turtle.circle(100)##지구 궤도
  turtle.penup()
  f = open("C:\\Users\\user\\Desktop\\MyOnlyLord.txt", 'r')
  turtle.color('gray')
```

```
  while True:
    light = f.readline()
    if not light:break
    line=list(light)
    ilist = list(filter(lambda x: line[x] == '\t', range(len(line))))
    planetrise = line[ ilist[1] + 1 : ilist[2] ]
    sunrise = line[ ilist[7] + 1 : ilist[8] ]
    e_dist = float(''.join(line[ ilist[4] + 1 : ilist[5] ]))
    s_dist = float(''.join(line[ ilist[5] + 1 : ilist[6] ]))
    elo=get_earthelo(sunrise,planetrise)
    y = (e_dist**2 - s_dist**2 + 1) / 2
    if e_dist<y:
        print('모순점 발생')
        continue
    x = math.sqrt(e_dist**2 - y**2)
    if elo<=0:
      x*=-1
    turtle.goto(x*100, y*100)
    turtle.pendown()
    print(sunrise, planetrise, elo)
  f.close()
  f = open("C:\\Users\\user\\Desktop\\MyOnlyLight.txt", 'r')
  turtle.color('orange')
  turtle.penup()
  while True:
    light = f.readline()
    if not light:break
    line=list(light)
    ilist = list(filter(lambda x: line[x] == '\t', range(len(line))))
    planetrise = line[ ilist[2] + 1 : ilist[3] ]
    sunrise = line[ ilist[8] + 1 : ilist[9] ]
    e_dist = float(''.join(line[ ilist[5] + 1 : ilist[6] ]))
    s_dist = float(''.join(line[ ilist[6] + 1 : ilist[7] ]))
    elo=get_earthelo(sunrise,planetrise)
    y = (e_dist**2 - s_dist**2 + 1) / 2
    if e_dist<y:
        print('모순점 발생')
        continue
    x = math.sqrt(e_dist**2 - y**2)
    if elo<=0:
      x*=-1
    turtle.goto(x*100, y*100)
    turtle.pendown()
    print(sunrise, planetrise, elo)
  f.close()
```

코드 설명
1.지구와 태양의 기본 좌표 지정(지구=(0,0) 태양=(0,100) 100=1AU)

2.이각 구하기, 거리 구하기 함수 정의하기

3.두 거리 자료(텍스트 문서)를 이용한 행성의 좌표 구하는 함수 정의(두 원의 교점 이용)

4.지구와 태양, 완벽한 원이라고 가정한 지구의 공전궤도를 turtle을 이용해 그린다

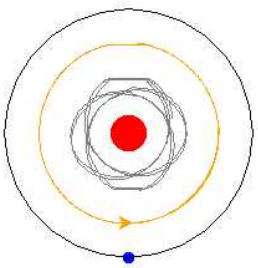
5.turtle을 이용해 행성의 좌표를 따라 그림 그리는 함수 프로그래밍(수성 궤도는 회색, 금성 궤도는 금색으로 색칠)

MyOnlyLight.txt

1	2020	Jan 1	9:40	14:49	19:57	1.278	0.7263	7:33	12:26	17:21
2	2020	Jan 2	9:41	14:51	20:00	1.2724	0.7262	7:30	12:27	17:22
3	2020	Jan 3	9:41	14:52	20:03	1.2667	0.7261	7:31	12:27	17:23
4	2020	Jan 4	9:42	14:54	20:06	1.261	0.7259	7:32	12:27	17:24
5	2020	Jan 5	9:42	14:56	20:09	1.2553	0.7258	7:32	12:28	17:24
6	2020	Jan 6	9:39	14:53	20:08	1.2495	0.7257	7:33	12:28	17:25
7	2020	Jan 7	9:39	14:55	20:11	1.2437	0.7256	7:34	12:29	17:26
8	2020	Jan 8	9:39	14:57	20:14	1.2379	0.7255	7:31	12:29	17:31
9	2020	Jan 9	9:40	14:58	20:17	1.232	0.7254	7:31	12:29	17:31
10	2020	Jan 10	9:40	15:00	20:19	1.2262	0.7252	7:32	12:30	17:32
11	2020	Jan 11	9:40	15:01	20:22	1.2203	0.7251	7:33	12:30	17:33
12	2020	Jan 12	9:36	14:58	20:21	1.2143	0.725	7:33	12:30	17:33
13	2020	Jan 13	9:36	15:00	20:23	1.2084	0.7248	7:30	12:31	17:34
14	2020	Jan 14	9:36	15:01	20:26	1.2024	0.7247	7:31	12:31	17:35
15	2020	Jan 15	9:36	15:02	20:29	1.1963	0.7246	7:31	12:31	17:35
16	2020	Jan 16	9:36	15:03	20:31	1.1903	0.7245	7:32	12:32	17:36
17	2020	Jan 17	9:35	15:05	20:34	1.1842	0.7243	7:33	12:32	17:37
18	2020	Jan 18	9:35	15:06	20:36	1.1781	0.7242	7:29	12:32	17:37
19	2020	Jan 19	9:31	15:03	20:35	1.172	0.7241	7:30	12:37	17:38
20	2020	Jan 20	9:30	15:04	20:37	1.1658	0.7239	7:30	12:37	17:38
21	2020	Jan 21	9:30	15:05	20:39	1.1596	0.7238	7:31	12:37	17:43
22	2020	Jan 22	9:29	15:06	20:42	1.1534	0.7236	7:27	12:37	17:43
23	2020	Jan 23	9:29	15:07	20:44	1.1472	0.7235	7:28	12:37	17:44
24	2020	Jan 24	9:28	15:07	20:46	1.1409	0.7234	7:28	12:38	17:44
25	2020	Jan 25	9:28	15:08	20:49	1.1346	0.7232	7:28	12:38	17:44
26	2020	Jan 26	9:27	15:09	20:51	1.1283	0.7231	7:29	12:38	17:45
27	2020	Jan 27	9:26	15:10	20:53	1.1219	0.723	7:25	12:38	17:45
28	2020	Jan 28	9:26	15:11	20:55	1.1155	0.7228	7:25	12:38	17:49
29	2020	Jan 29	9:25	15:11	20:58	1.1091	0.7227	7:26	12:39	17:50
30	2020	Jan 30	9:20	15:08	20:56	1.1026	0.7226	7:26	12:39	17:50
31	2020	Jan 31	9:19	15:09	20:58	1.0961	0.7224	7:22	12:39	17:50
32	2020	Feb 1	9:19	15:09	21:00	1.0896	0.7223	7:22	12:39	17:50
33	2020	Feb 2	9:18	15:10	21:02	1.0831	0.7222	7:22	12:39	17:54
34	2020	Feb 3	9:17	15:10	21:04	1.0765	0.722	7:23	12:39	17:55
35	2020	Feb 4	9:16	15:11	21:06	1.0699	0.7219	7:23	12:39	17:55

그림 1 :금성의 자료 텍스트 형태

금성의 뜨는 시간, 위상, 태양의 뜨는 시간, 지는 시간 거리 등의 자료가 있다.



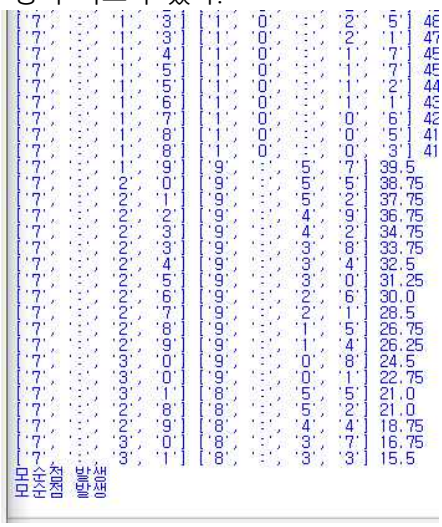


그림 2 실행 결과

금성과 수성의 회합 궤도를 그릴수 있었다.

알수 있는점

1.금성과 수성의 공전궤도는 타원 궤도다.

:프로그램 실행 결과 중 만약 궤도가 원이라고 가정하고 좌표를 구하는 프로그램중 x값이 허수

가 나올 경우 '모순점 발생'을 프린트 하게 하고 넘어가도록 했다. 완전한 원이라고 가정 했을 시 모순이 발생했다는 점에서 타원 궤도임을 알 수 있다.

2.금성과 수성의 이각을 알 수 있다.

:이를 프로그램을 통해 직접 프린트 하도록 하였고 그림 2에 표현되어있다.