파이썬을 이용한 텔로 에듀 Objection Detection/컨트롤-파이썬에서 OpenCV를 활용한 이미지 처리 기초

---

python vs. micropython - 파이썬과 마이크로파이썬의 탄생기원

파이썬

- ABC언어의 후속프로젝트로 1989년 설계하기 시작해서 나중에 파이썬으로 불리게 되었음
- 파이썬 언어 철학은 코드 가독성을 높이고, 프로그래머가 더 적은 코드로 더 많은 작업을 수행할 수 있도록 설계되었음

  - 파이썬의 명확하고 간결한 문법에 반영되어 있음
  - 파이썬은 다양한 프로그래밍 패러다임(절차적, 객체 지향, 함수형 프로그래밍)을 지원함
  - 파이썬의 첫 번째 공식 릴리스는 1991년에 공개되었으며 계속 다목적 프로그래밍 언어로 업데이트되고 있음

마이크로파이썬

- 2013년에 영국의 프로그래머 "데미안 조지(Damien George)"에 의해 개발되었음
- 소형 전자 장치와 마이크로컨트롤러에서 동작하는 파이썬 구현을 만들기 위해 마이크로파이썬을 개발하기 시작했음

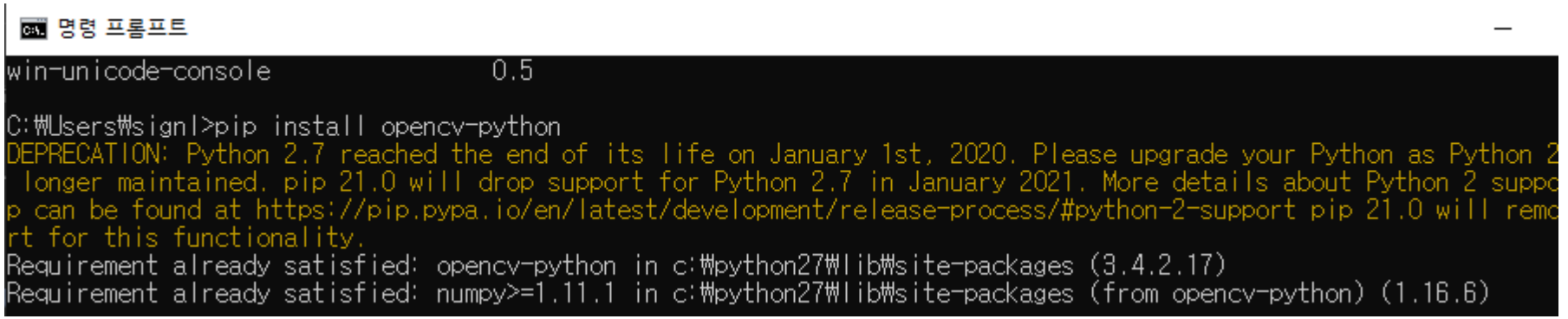
- 임베디드 시스템에서의 프로그래밍은 주로 C 언어로 이루어졌으나, 파이썬의 높은 생산성과 간결한 문법이 임베디드 프로그래밍에도 유용하다는 생각에서 출발했음
- 마이크로파이썬은 메모리와 성능 제약이 있는 소형 마이크로컨트롤러에서도 효율적으로 동작할 수 있도록 설계되었음
- 기존의 파이썬 인터프리터를 소형화하고 최적화한 것임
    - 다양한 마이크로컨트롤러 보드, 특히 ESP8266, ESP32, STM32 등에서 널리 사용됨

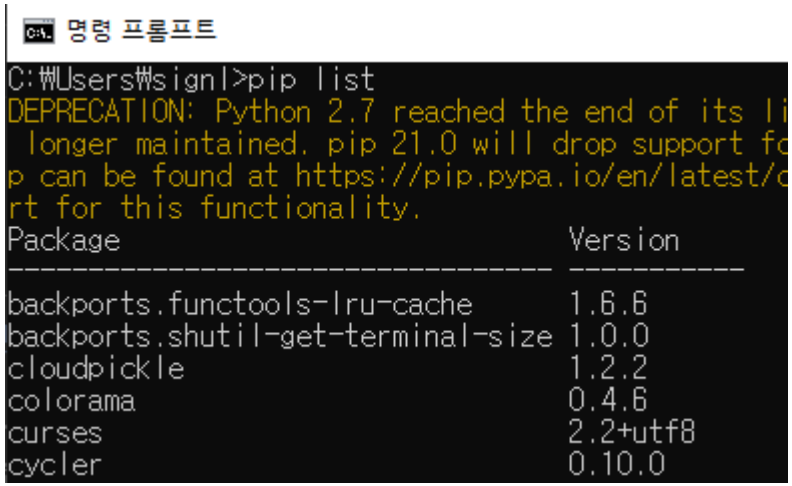
파이썬과 OpenCV의 만남

- 파이썬은 그 간결함과 다양한 라이브러리 지원으로 인해 이미지 처리 분야에서도 널리 사용되고 있음
- OpenCV(Open Source Computer Vision Library)는 오픈 소스 컴퓨터 비전 라이브러리로서, 이미지 처리와 관련된 다양한 기능을 제공함
    - OpenCV는 C++로 작성되었지만, 파이썬 바인딩을 통해 파이썬에서도 모든 기능을 사용할 수 있음
    - 이를 통해 개발자는 파이썬의 간결한 문법을 활용하여 복잡한 이미지 처리 작업을 쉽게 수행할 수 있음

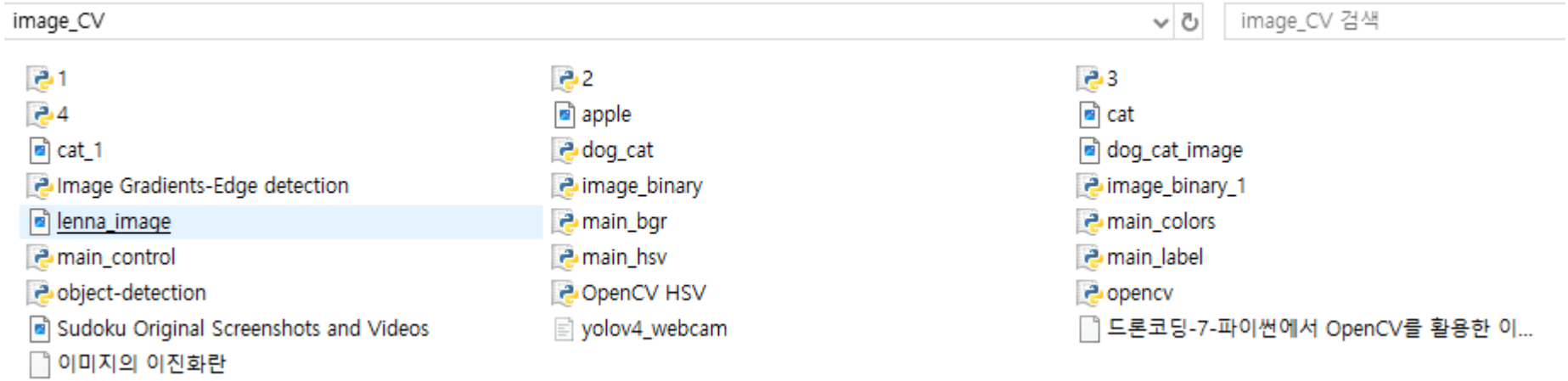
OpenCV를 파이썬에서 사용 - OpenCV를 활용한 기본 이미지 처리

- pip를 통해 opencv-python 패키지를 설치해야 함
- OpenCV는 이미지의 크기 변경, 회전, 필터링 등 기본적인 이미지 처리 기능을 제공함





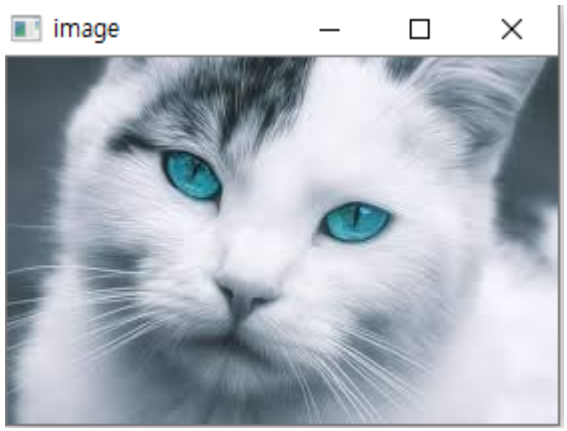
OpenCV를 사용하여 이미지를 불러오고 화면에 표시 - python IDE에서 실행할 것!!!!!!



- OpenCV를 사용하여 이미지를 불러오고 화면에 표시 - 1.py

```
#https://f-lab.kr/insight/python-opencv-basics
#OpenCV를 사용하여 이미지를 불러오고 화면에 표시

import cv2

#이미지 파일을 읽어옴
img = cv2.imread('cat.jpg')

#이미지를 화면에 표시
cv2.imshow('image', img)

#키 입력을 대기한다
#waitKey() 함수
#키 입력을 기다리는 대기 함수
#인자 값으로 0 : 무한 대기 / ms(밀리세컨) 단위의 시간을 입력하면
#해당 시간만큼 대기 (1000ms = 1초)
#waitKey의 리턴 값은 키보드로 입력한 키와 동일한 아스키코드 값
cv2.waitKey(0)

```

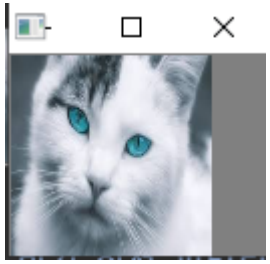
```
20 # 모든 윈도우를 종료
21 cv2.destroyAllWindows()
```



- 이미지의 크기 변경, 회전, 필터링 등 기본적인 이미지 처리 기능 - 2.py

```
1 import cv2
2
3 # 이미지 파일을 읽어온다
4 img = cv2.imread('cat.jpg')
5
6 # 이미지의 크기를 변경한다
7 resized_img = cv2.resize(img, (100, 100))
8
9 # 변경된 이미지를 화면에 표시한다
10 cv2.imshow('Resized Image', resized_img)
11
12 # 키 입력을 대기한다
13 cv2.waitKey(0)
14
15 # 모든 윈도우를 종료한다
```

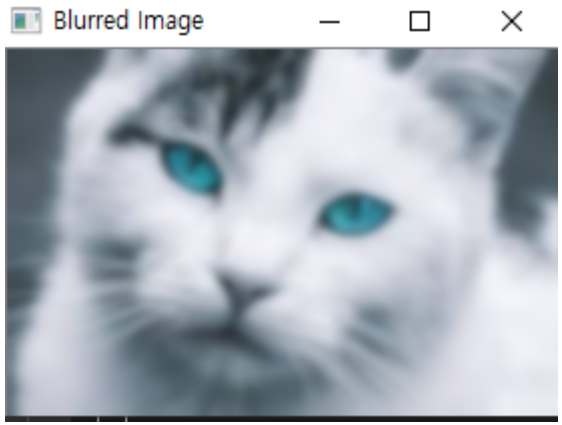
```
cv2.destroyAllWindows()

```



- 다양한 이미지 필터링 기능을 제공 - 가우시안 블러를 적용하여 이미지를 부드럽게 처리 - 3.py

```
import cv2

# 이미지 파일을 읽어온다
img = cv2.imread('cat.jpg')

# 가우시안 블러를 적용한다
# 이미지의 노이즈를 줄이고 부드러운 효과를 주기 위해 사용되는 필터
# a function that blurs an image using a Gaussian filter
# with a kernel of size 5 x 5 and a standard deviation of 0.
# 커널 사이즈는 보통 양수의 홀수로 지정
# 예시 : (3,3) (5,5), (7,7)
# 커널 사이즈가 커질 수록 흐림효과 증가

#blurred_img = cv2.GaussianBlur(img, (5, 5), 0)


blurred_img = cv2.GaussianBlur(img, (11, 11), 0)

# 블러 처리된 이미지를 화면에 표시한다
cv2.imshow('Blurred Image', blurred_img)
```

```
21
22 # 키 입력을 대기한다
23 cv2.waitKey(0)
24
25 # 모든 윈도우를 종료한다
26 cv2.destroyAllWindows()
```



이미지의 특징을 추출하고, 객체를 인식하고, 이미지 간의 매칭을 수행하는 등 고급 이미지 처리 기능도 제공 - 이미지에서 특정 색상의 객체를 검출

- BGR 색상 공간의 이미지를 HSV 색상 공간으로 변환할 수 있음
  - 색공간 - 이미지의 색상을 숫자로 표현하는 방식
  - 색상 공간 - RGB, HSV, YCbCr 등이 대표적인 색공간
    - BGR - 빨강(Blue), 초록(Green), 파랑(Red)을 뜻하는 색상 공간
    - HSV - 색조(Hue), 채도(Saturation), 명도(Value)를 뜻하는 색상 공간
      - HSV는 색상 자체를 알려주므로 직관성이 좋음
      - HSV를 사용하면 색상 변경, 채도 증감, 값 조정 등 개별 색상 속성을 쉽게 조정할 수 있음

- 색상공간 변환 - 4.py

```
1 import cv2
2 import numpy as np
3
4 # 이미지 파일을 읽어온다
5 img = cv2.imread('cat_1.jpg')
6
```

```
# BGR 색상 공간에서 HSV 색상 공간으로 변환한다
# 이미지의 색상을 숫자로 표현하는 방식을 색공간이라고 하며
# RGB, HSV, YCbCr 등이 대표적인 색공간
hsv_img = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)


# 파란색 범위를 정의한다
lower_blue = np.array([110, 50, 50])
upper_blue = np.array([130, 255, 255])


# 이미지에서 파란색 객체만 추출한다
mask = cv2.inRange(hsv_img, lower_blue, upper_blue)

# 마스크를 원본 이미지에 적용한다
# 마스크(mask)는 이미지의 일부를 가리거나 숨기는 기능을 의미
# 포토샵, 피그마, 유니티 등에서 사용할 수 있음
# 밝은 영역(흰색) 어두운 영역(검은색)으로 나누어져 효과가 나타남
# 즉 흰색 영역은 이미지가 보이게 되고 검은색 영역은 가려지게 됨
result_img = cv2.bitwise_and(img, img, mask=mask)

# 결과 이미지를 화면에 표시한다
cv2.imshow('Blue Objects', result_img)

# 키 입력을 대기한다
cv2.waitKey(0)

# 모든 윈도우를 종료한다
cv2.destroyAllWindows()
```

텔로 에듀의 비디오를 수신하여 포착된 물체를 기본적인 화상저리 알고리즘을 통해 감지하고 이를 이용하여 기체를 컨트롤해보자!!!

화상처리의 기본적인 방법인 색감지를 통한 물체 감지 기법을 다룸

- 2치화, 라벨링, 면적&중심 계산 알고리즘 등

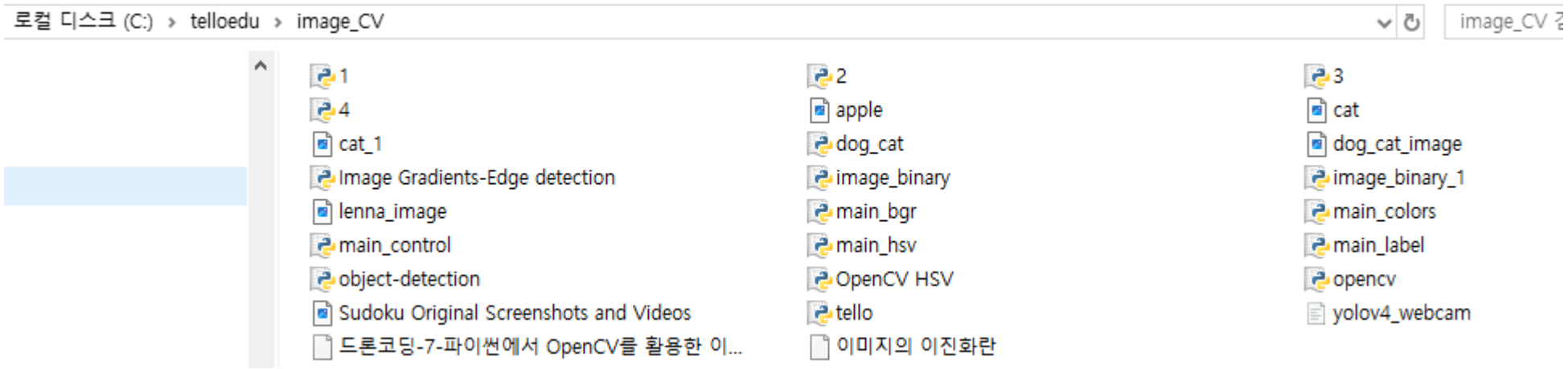
화상처리 라이브러리 패키지인 OpenCV의 라벨링 함수를 활용하여 오렌지색의 칼라콘(임의의 물체)을 식별하고 화면 중심 위치와의 차이를 이용해 기체를 컨트롤하는 솔루션을 구현하는 것임

- 색상기반으로 물체 감지 및 화상처리하고 기체를 컨트롤하는 순서는 다음과 같음
    - 텔로의 RGB 화상 -> HSV 화상 변환 -> HSV 화상의 2 치화 -> HSV 라벨링 -> 면적 & 중심 계산 -> 물체 감지 -> 화면 중심 좌표와의 차이 -> 기체의 컨트롤

HSV 화상으로 변환하는 이유 - 보다 색상 표현이 단순해져 화상처리가 용이하기 때문임

- 2치화를 수행하는 이유 - 다양한 밝기 레벨을 명과 암으로 단순화하는 처리를 통해 특정 색상만을 구별해내는 것이 가능해지기 때문임
- 이후 감지된 부분에 번호를 매기고 그 위치의 중심을 찾기 위한 면적과 중심 계산과정을 거치는 것임

실습 진행을 위한 프로그램을 다운로드 및 설치



- C:\telloedu 하부에 image_CV 폴더를 생성
- https://github.com/hsgucci404/tello/tree/master/Tello_CV_color/main_colors.py를 main_colors.py를 다운로드하여 image_CV 폴더에 복사
- C:\telloedu\Tello-Python-master\Tello_Video 폴더의 tello.py를 image_CV 폴더에 복사

    - https://github.com/dji-sdk/Tello-Python

- 텔로를 Wifi에 접속하고 main_colors.py를 실행

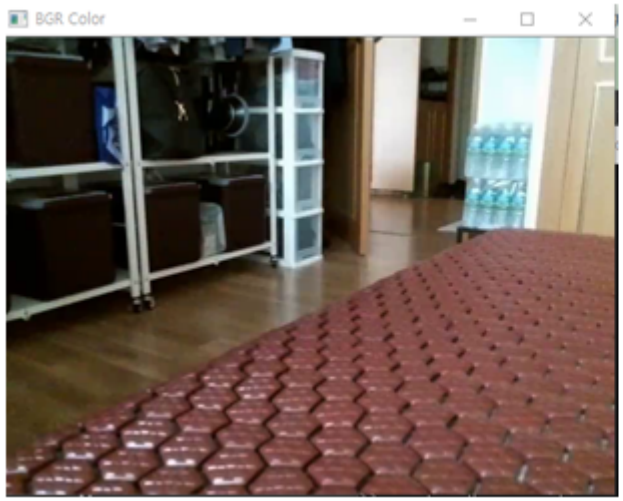
    - 텔로는 직접 접속모드(AP 접속모드)이어야 함

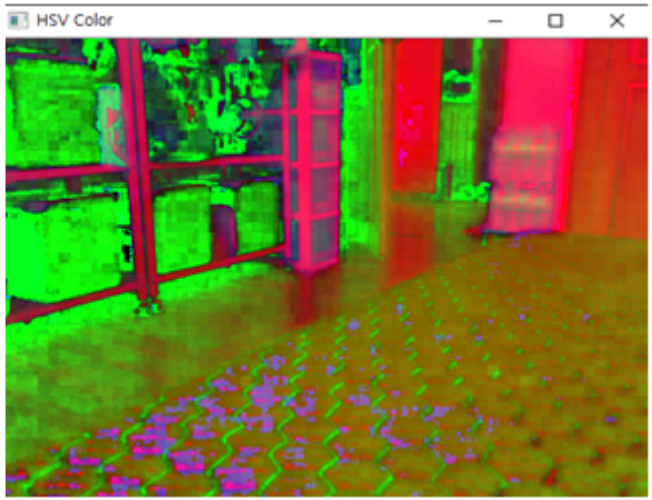
        - 프로그램이 정상적으로 작동하면 상태표시창과 RGB Color 화면과 HSV Color 화면이 나타남
        - RGB는 일반적인 색상을 가리키며 일반적인 비디오 컬러 화면을 의미
        - HSV(Hue=색상, Saturation=채도, Value=명도)의 색공간으로 뱐환한 컬러를 의미

main_colors.py - C:₩telloedu₩image_CV₩main_colors.py (2.7.15)

File Edit Format Run Options Window Help

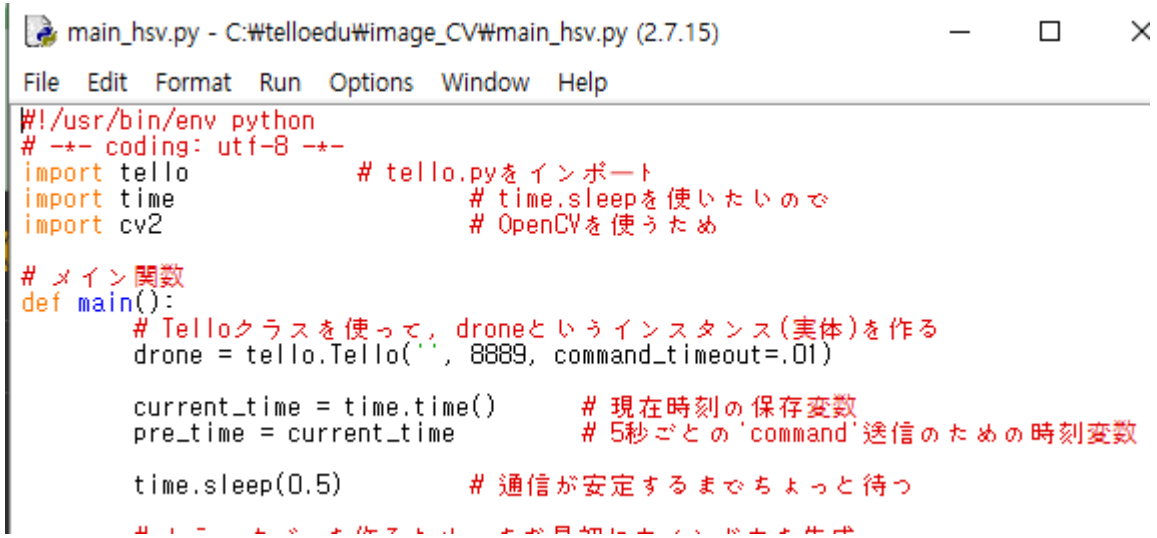
```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
import tello             # tello.pyをインポート
import time                      # time.sleepを使いたいので
import cv2                       # OpenCVを使うため

# メイン関数
def main():
        # Telloクラスを使って，droneというインスタンス(実体)を作る
        drone = tello.Tello('', 8889, command_timeout=.01)

        current_time = time.time()       # 現在時刻の保存変数
        pre_time = current_time          # 5秒ごとの'command'送信のための

        time.sleep(0.5)          # 通信が安定するまでちょっと待つ
```





HSV 변환 화상에 대한 2치화를 수행하는 것을 테스트 해보자!!!

- https://github.com/hsgucci404/tello/blob/master/Tello_CV_color/에서 main_hsv.py를 다운로드하여 image_CV 폴더에 복사



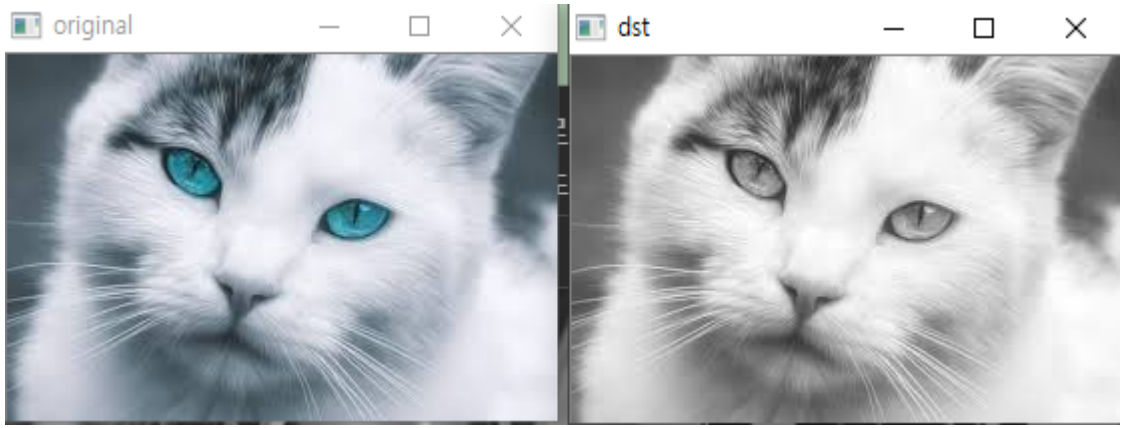
이미지의 이진화란

- 원본 색상 또는 그레이스케일 이미지를 작은 검은색과 흰색 픽셀로 구성된 디지털 이진 이미지로 변환하는 과정
- 이미지 이진화는 이미지 분석의 기초가 되며, 입자 식별을 위한 이미지 처리에서 중요한 단계임

- 이미지 이진화 - OpenCV HSV.py

```
import cv2
import numpy as np
import cv2

# 이미지 파일을 읽어온다
img = cv2.imread('cat.jpg')

cv2.imshow('original', img)

# 채도 낮추기
saturationScale = 0.01

# 이미지의 색상 공간을 변환하는 데 사용
```

```
# 이 함수를 사용하여 이미지를 다양한 색상 공간으로 변환
# 기본 BGR의 컬러를. GRAY COLOR_BGR2GRAY, COLOR_GRAY2BGR COLOR_RGB2GRAY, COLOR_GRAY2RGB
hsvImage = cv2.cvtColor(img , cv2.COLOR_BGR2HSV)

# HSV 이미지를 float32 형태로 변환
hsvImage = np.float32(hsvImage)

# 채널로 분리하는 함수  ( 다차원일 경우 사용)
H, S, V = cv2.split(hsvImage)    # 분리됨

# 유용한함수. np.clip 함수 이용하면 0보다 작으면 0으로 맞추고,
# 255보다 크면 255로 맞추라 할수 있음
S = np.clip( S * saturationScale , 0,255 ) # 계산값, 최소값, 최대값

# 여기서는 saturation값만 조정
# H,S,V 나눈 채널을 다시 합치는 함수
hsvImage = cv2.merge( [ H,S,V ] )

# 위에서 float으로 작업했으로, 다시 uint8로 변경해야함
hsvImage = np.uint8(hsvImage)

# BGR로 다시 변경해야 , 우리가 눈으로 확인 가능
imgBgr = cv2.cvtColor(hsvImage, cv2.COLOR_HSV2BGR)
cv2.imshow('dst', imgBgr)

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



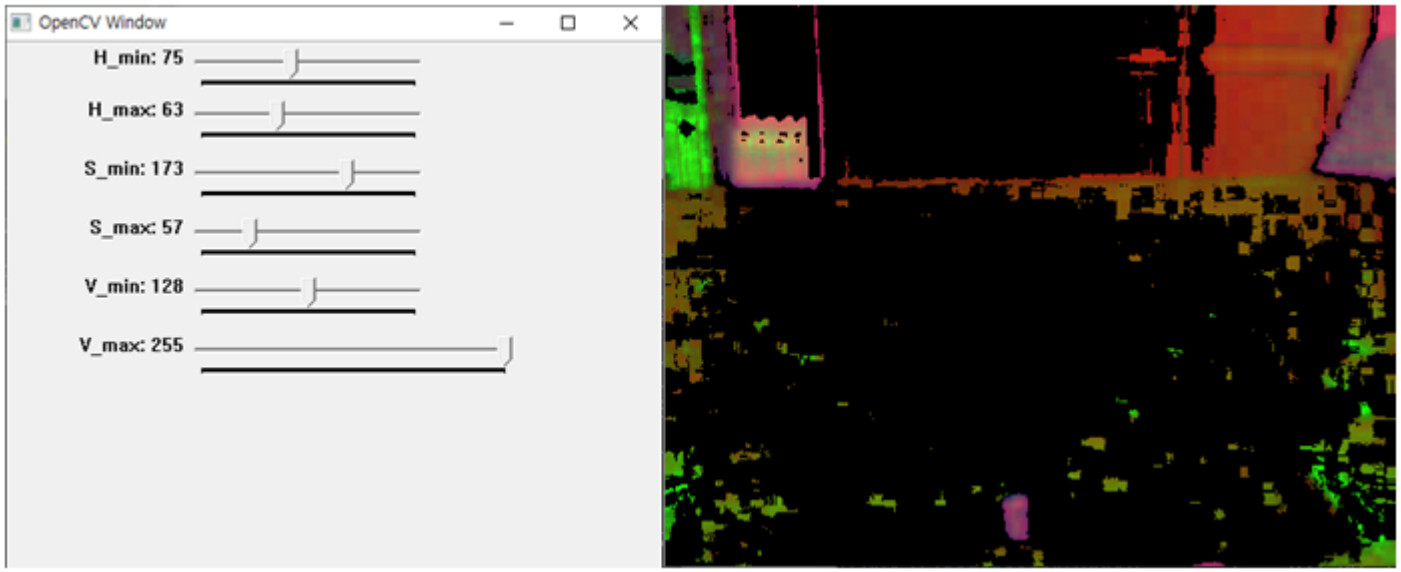
텔로 에듀를 Wifi에 접속하고 main_hsv.py 파일을 실행

- 텔로는 직접 접속모드(AP 접속모드)이어야 함

main_hsv.py - C:₩telloedu₩image_CV₩main_hsv.py (2.7.15)

File Edit Format Run Options Window Help

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
import tello             # tello.pyをインポート
import time                      # time.sleepを使いたいので
import cv2                       # OpenCVを使うため

# メイン関数
def main():
        # Telloクラスを使って，droneというインスタンス(実体)を作る
        drone = tello.Tello('', 8889, command_timeout=.01)

        current_time = time.time()      # 現在時刻の保存変数
        pre_time = current_time         # 5秒ごとの'command'送信のための時刻変数

        time.sleep(0.5)         # 通信が安定するまでちょっと待つ
```



- 반드시 HSV 조절할 것!!!!

  - 프로그램이 정상직으로 작동하면 상태표시창과 "OpenCV Window" 화면이 생성됨
  - HSV 변환 화상과 H, S, V값을 조절할 수 있는 트랙바가 나타남
  - 최저값과 최대값을 조절하면 화면에 나타나는 색과 범위가 조절되어 특정 색상만을 띄울수 있음(즉, 특정 색을 가진 물체를 식빌하는 것이 가능)

- 텔로의 HSV 변환 화상을 이용하여 2치화, 라벨링, 면적 & 중심계산을 해보자!!!

  - 원리는 2치화된 영상에서 특정 값을 갖는 화소를 찾아내는 것으로 2치화되어 물체의 색상과 나머지 주변의 색상으로 분리
  - https://github.com/hsgucci404/tello/blob/master/Tello_CV_color/에서 main_label.py를 다운로드하여 image_CV 폴더에 복사

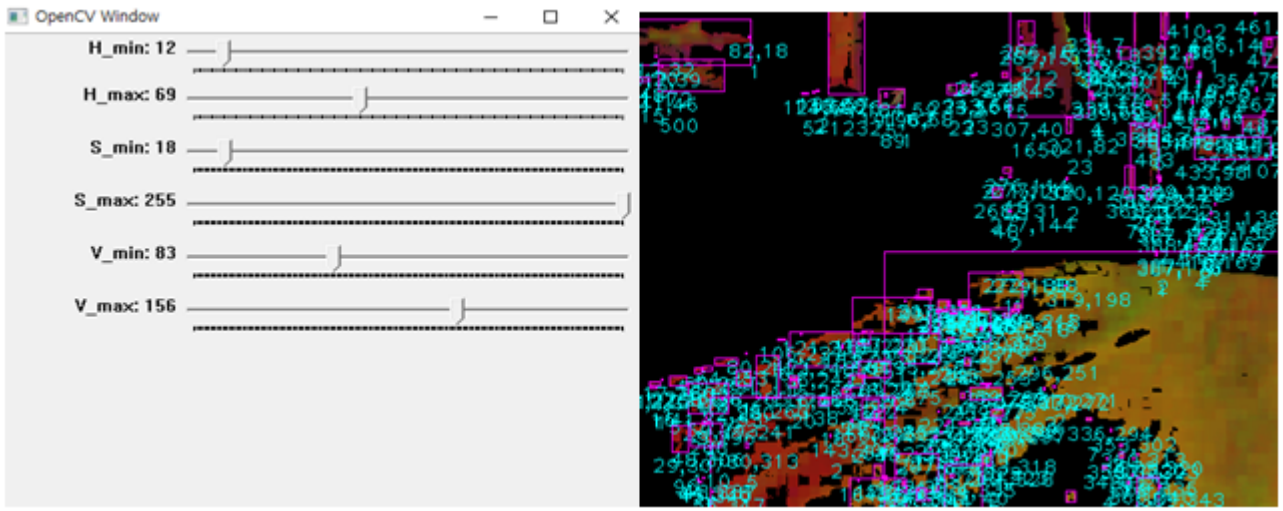
main_label.py - C:₩telloedu₩image_CV₩main_label.py (2.7.15)

File Edit Format Run Options Window Help

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
import tello             # tello.pyをインポート
import time                      # time.sleepを使いたいので
import cv2                       # OpenCVを使うため
import numpy as np

# メイン関数
def main():
        # Telloクラスを使って, droneというインスタンス(実体)を作る
        drone = tello.Tello('', 8889, command_timeout=.01)

        current_time = time.time()       # 現在時刻の保存変数
        pre_time = current_time          # 5秒ごとの'command'送信のための時刻

        time.sleep(0.5)          # 通信が安定するまでちょっと待つ
```

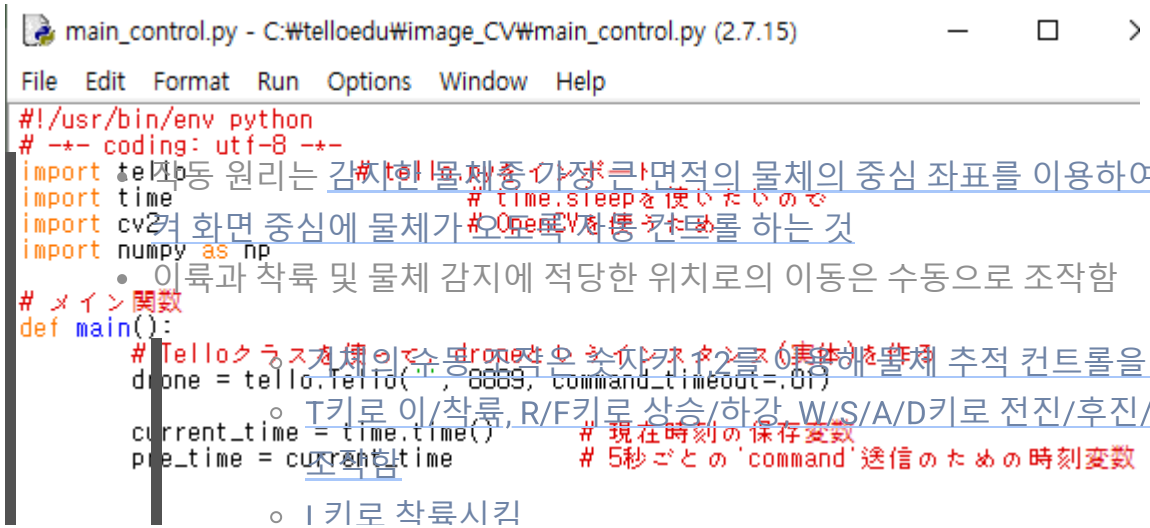
- 텔로 에듀를 Wifi에 접속하고 main_label.py 파일을 실행

  - 텔로는 직접 접속모드(AP 접속모드)이어야 함
  - 프로그램이 정상적으로 작동하면 상태표시창과 "OpenCV Window" 화면이 생성됨
  - HSV 변환 화상과 H, S, V값을 조절할 수 있는 트랙바가 나타남
  - 최저값과 최대값을 조절하면 화면에 나타나는 색과 범위가 조절되어 특정 색상만을 띄울수 있음
  - 물체의 중심 위치(x, y)와 면적이 숫자로 표현됨



감지한 물체를 텔로가 따라서 움직이는 물체 추적을 살펴보자!!!

- https://github.com/hsgucci404/tello/blob/master/Tello_CV_color/에서 main_control.py를 다운로드하여 image_CV 폴더에 복사



- 자동 원리는 감지된 물체 중 가장 큰 면적의 물체의 중심 좌표를 이용하여 화면 중심으로부터 벗어난 양만큼 텔로를 회전시켜 화면 중심에 물체가 오도록 제어 컨트롤 하는 것
- 이륙과 착륙 및 물체 감지에 적당한 위치로의 이동은 수동으로 조작함
  - 제어 명령은 스마트폰 앱을 이용해 텔로 추적 컨트롤을 on/off 시킴
    - T키로 이/착륙, R/F키로 상승/하강, W/S/A/D키로 전진/후진/좌이동/우이동, Q/E키로 좌회전/우회전 조작함
    - I 키로 착륙시킴