

미래형자동차 교육 워크숍

Subject: Computer Vision

Automation Lab.



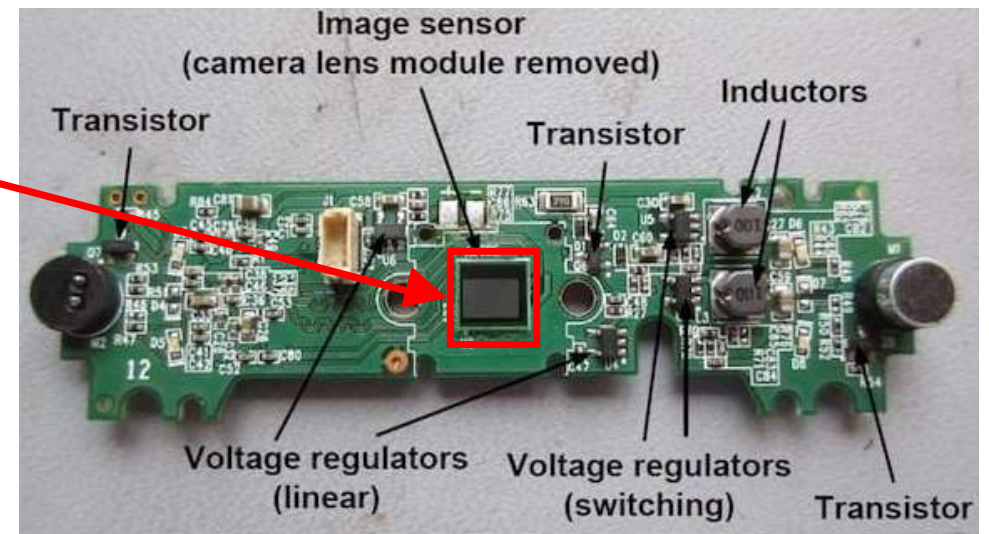
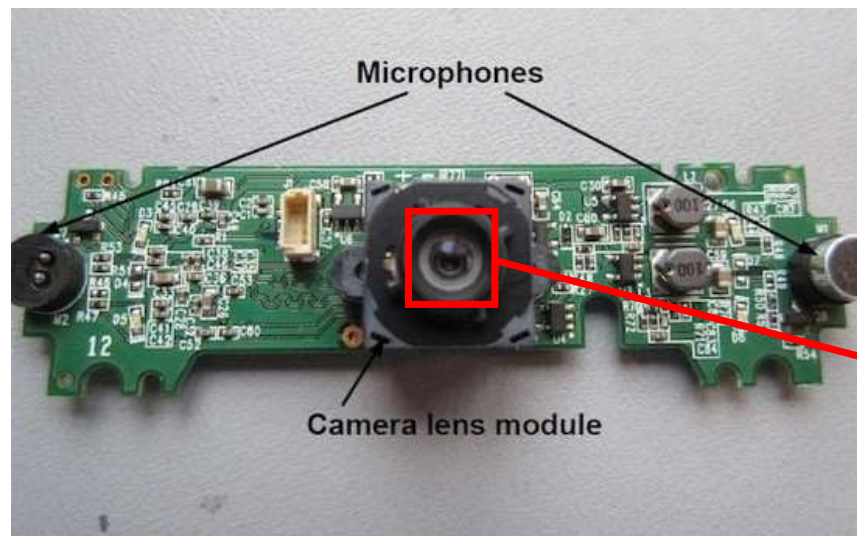
Introduction

■ Logitech C920 HD PRO Webcam



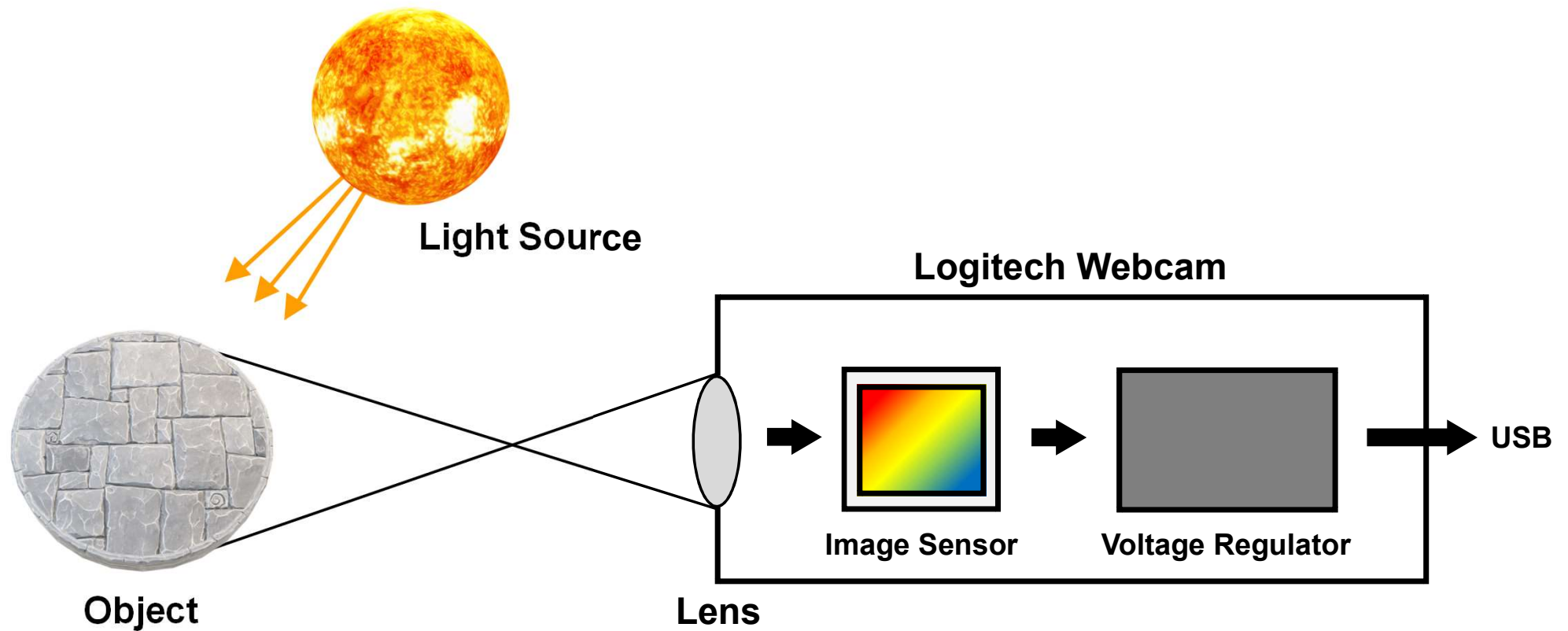
Introduction

■ Webcam Internal Structure



Introduction

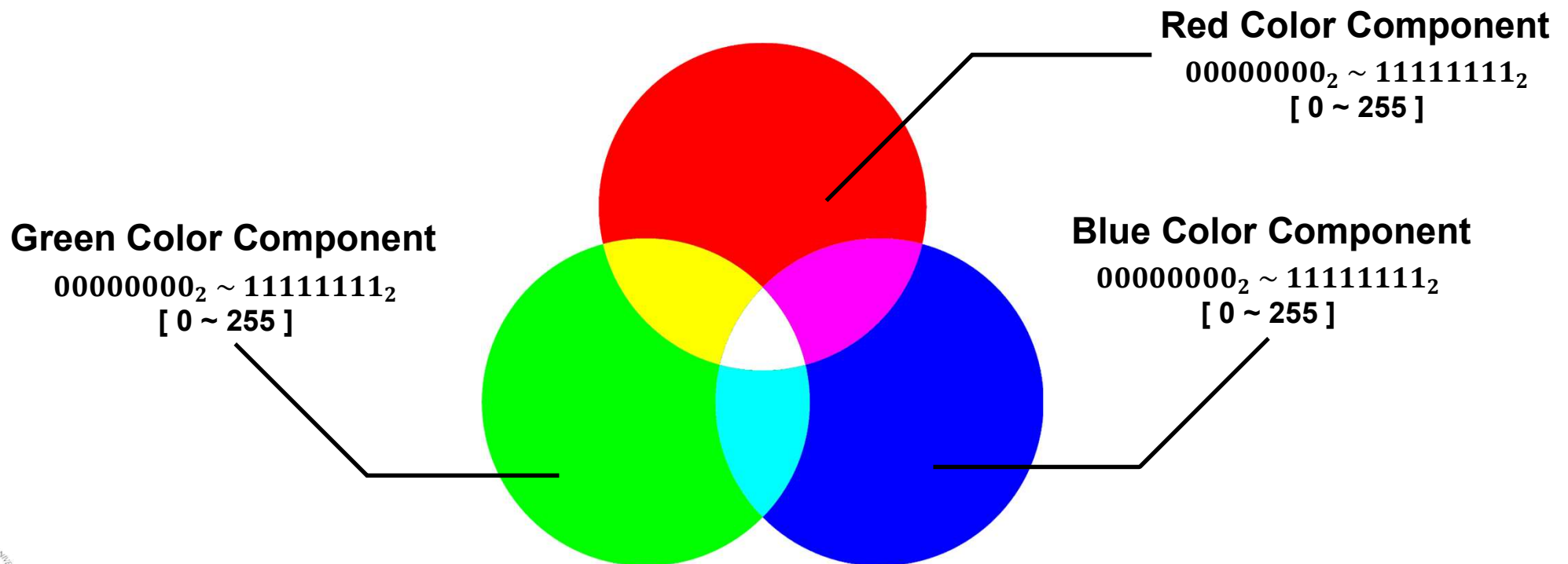
■ Image Formation



Introduction

■ 3-type Primary Color [RGB Color]

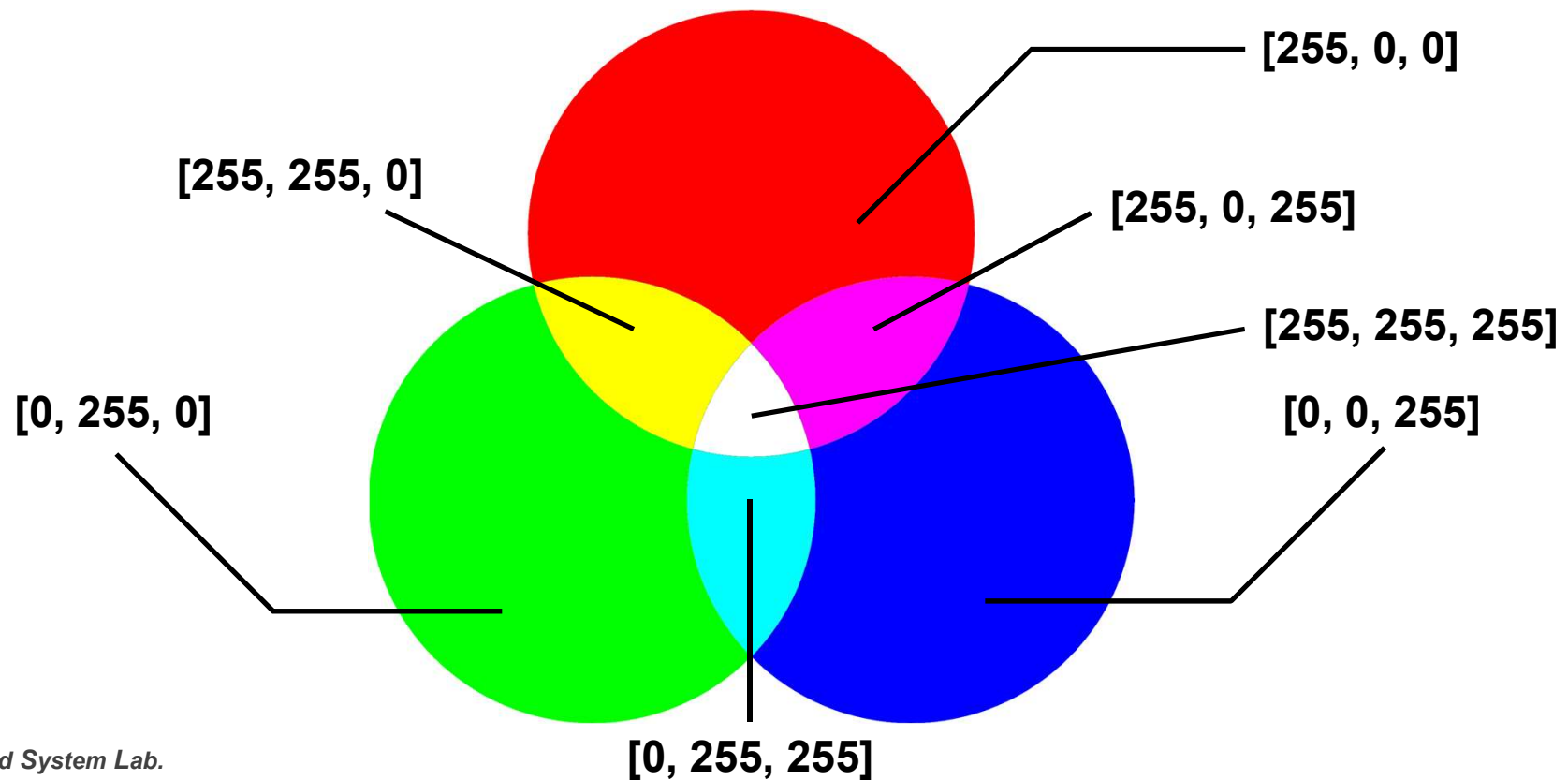
- 8-bit Color Depth: 컴퓨터가 색상을 표현하는 방법 (Most Common Method)
- 총 "256×256×256=16,777,216가지" 색상 표현 가능



Introduction

■ 3-type Primary Color [RGB Color]

→ Digital Value로 여러 가지 색상을 표현할 수 있음 [Red, Green, Blue]

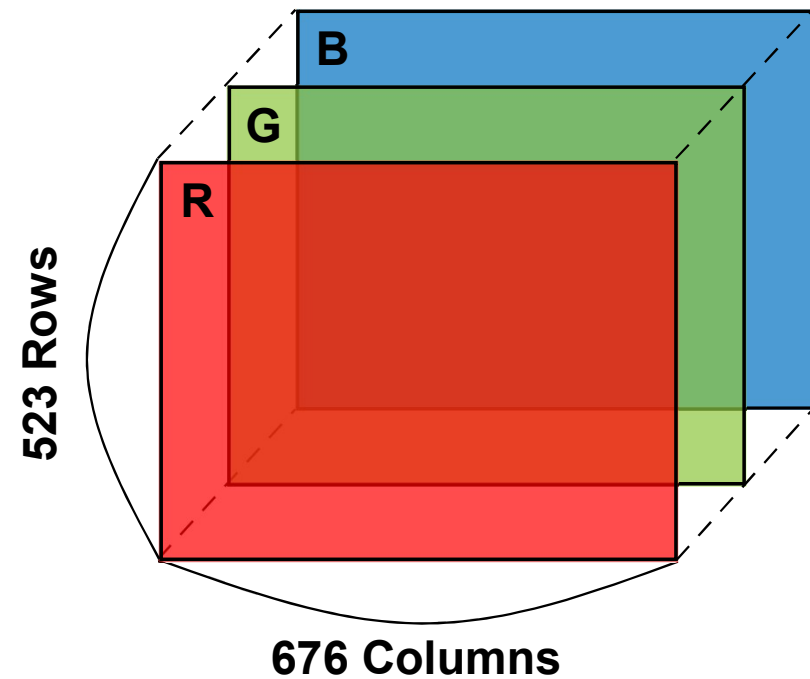
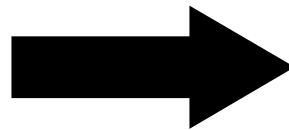


Introduction

■ RGB Image Matrix



676 × 523 pixels

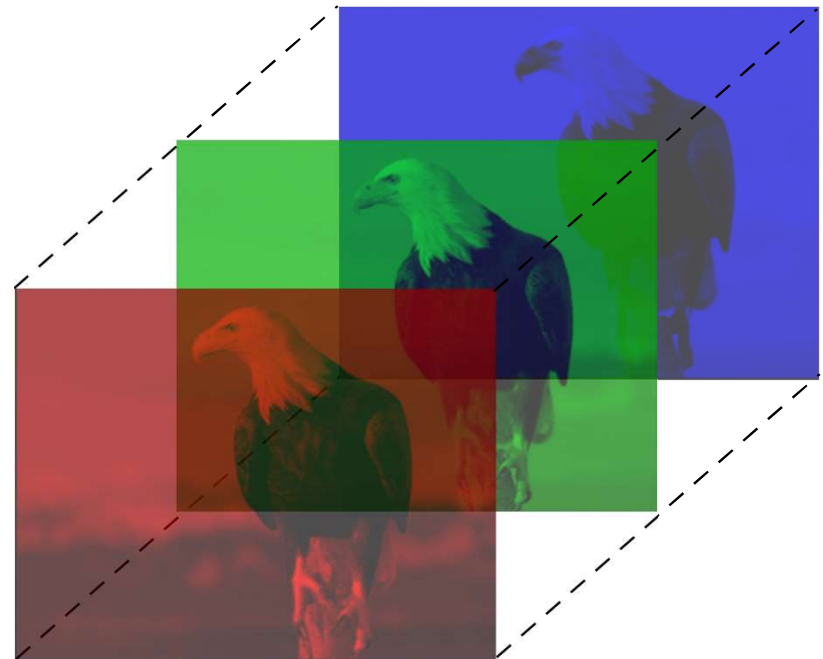
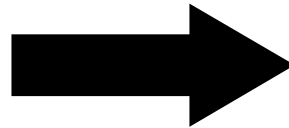


Introduction

■ RGB Image Matrix



676 × 523 pixels



■ RGB Image Matrix (Python)



Embedded System Lab.

Exercise

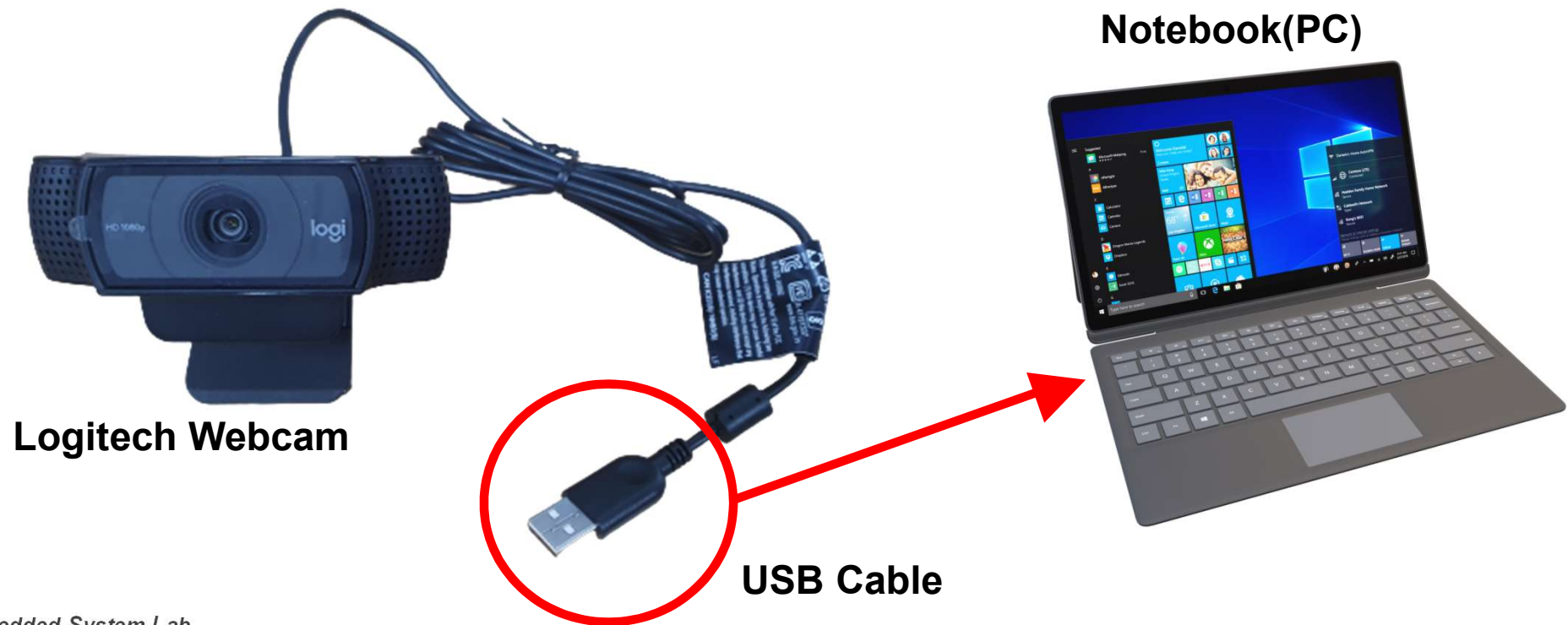
Automation Lab.



Exercise 1

■ Webcam Hardware Setting

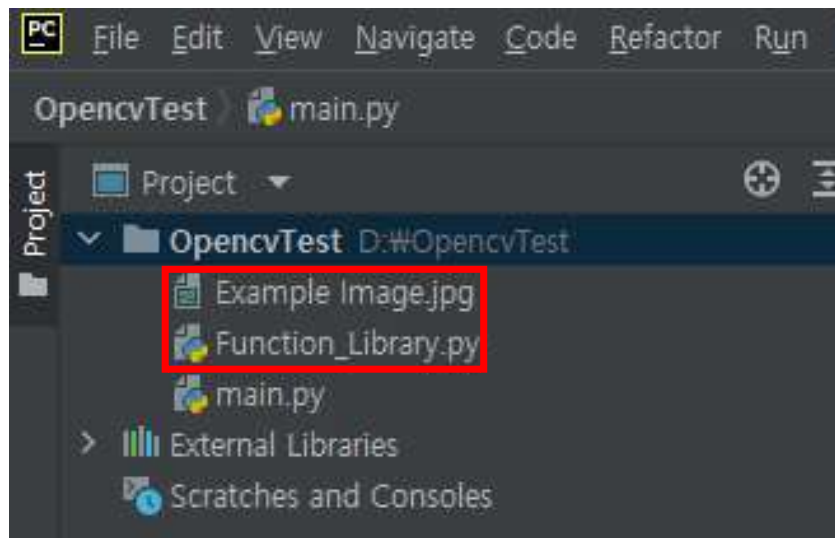
→ Webcam USB 케이블을 노트북(PC) USB 포트에 직접 연결



Exercise 1

■ Execute Pycharm

→ "Function_Library.py", "Example Image.jpg"를 Python Project에 삽입

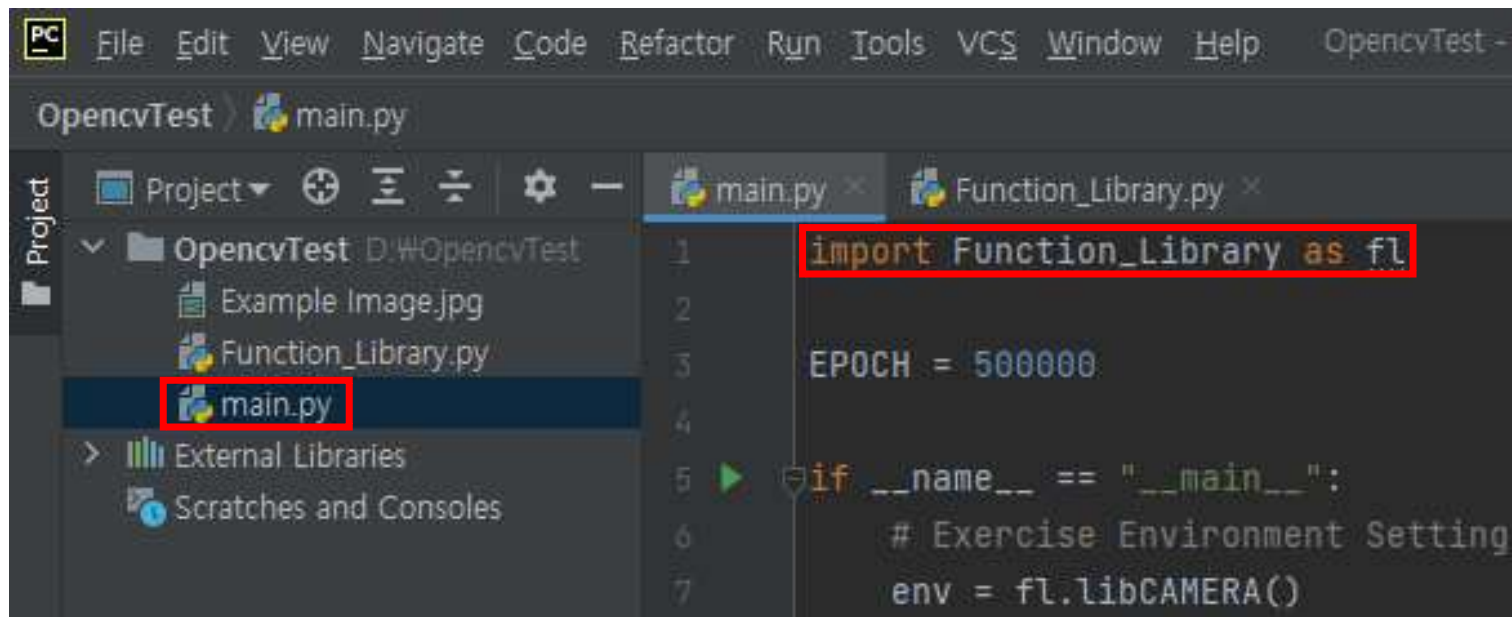


Example Image.jpg

Exercise 1

■ Import Library

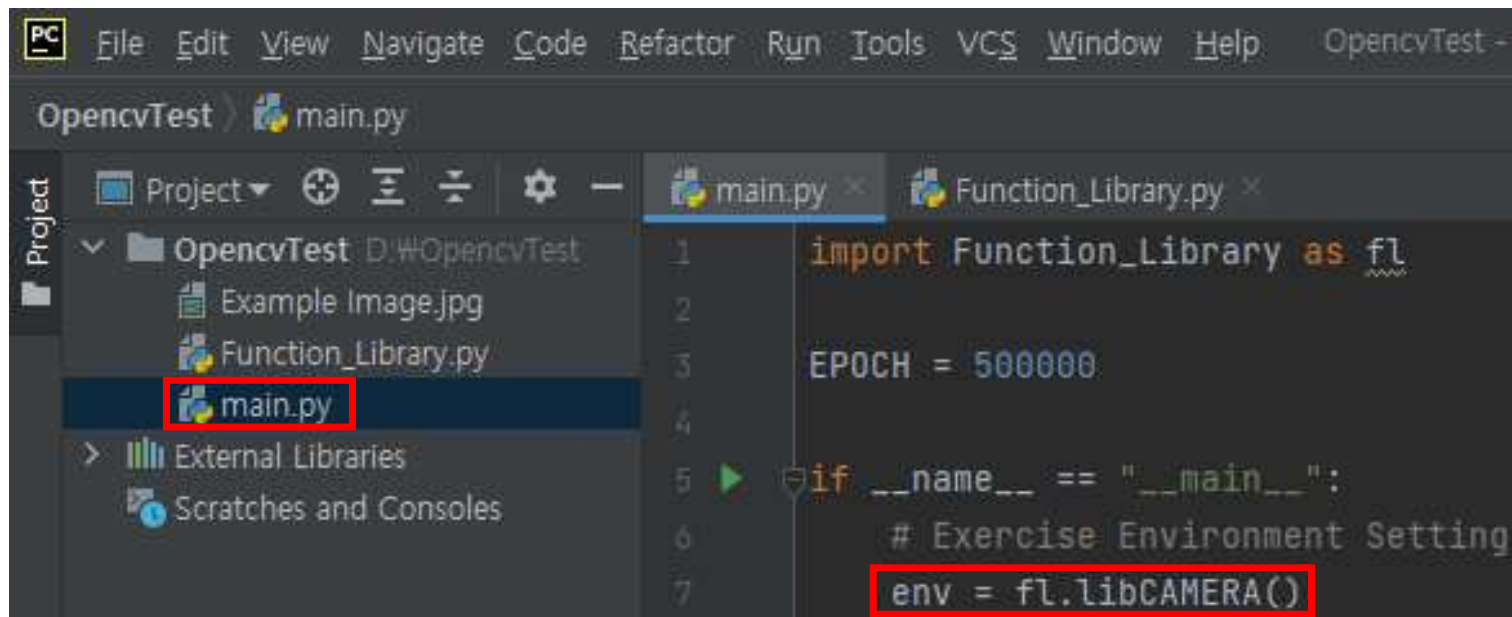
→ Main Code 생성 후, "Function_Library.py"를 불러옴



Exercise 1

■ Declare Environment

- "Function_Library.py"의 libCAMERA() Class를 불러옴
- libCAMERA() Class는 실습에 필요한 모든 함수들의 집합체



The screenshot shows an IDE window titled 'OpencvTest - 1'. The left sidebar displays a project tree for 'OpencvTest' with files 'Example Image.jpg', 'Function_Library.py', and 'main.py'. The 'main.py' file is selected and highlighted with a red rectangle. The main editor area shows the code for 'main.py' with line numbers 1 through 7. The code includes an import statement for 'Function_Library' as 'fl', a constant 'EPOCH = 500000', and a main execution block. Within the main block, the line 'env = fl.libCAMERA()' is highlighted with a red rectangle. The code is as follows:

```
1 import Function_Library as fl
2
3 EPOCH = 500000
4
5 if __name__ == "__main__":
6     # Exercise Environment Setting
7     env = fl.libCAMERA()
```

Exercise 1

■ RGB Color Value Extracting

→ 예시 샘플에 대한 Red/Green/Blue 값을 추출

```
if __name__ == "__main__":  
    # Exercise Environment Setting  
    env = fl.libCAMERA()  
  
    """ Exercise 1: RGB Color Value Extracting """  
    ##### YOU MUST EDIT ONLY HERE #####  
    example = env.file_read("./Example Image.jpg")  
    R, G, B = env.extract_rgb(example, print_enable=True)  
    quit()  
    #####
```

Exercise 1

■ “file_read()” Description

→ 지정한 경로에서 원하는 이미지 파일을 디지털 값으로 불러오는 함수

```
def file_read(self, img_path):
```

①

②

```
return np.array(cv2.imread(img_path))
```

① Input: 원하는 파일 경로를 입력

② Output: 컬러 이미지 파일의 경우, 디지털 값(3차원 배열)을 출력

→ (Size: Column × Row × Channel, 이때 Channel은 R/G/B 3가지에 해당함)

Exercise 1

■ “extract_rgb()” Description

→ 디지털 값으로 구성된 이미지 데이터에서 Red/Green/Blue 성분에 해당하는 값을 각각 분리하는 함수

```
def extract_rgb(self, img, print_enable=False):
```

①

②

③

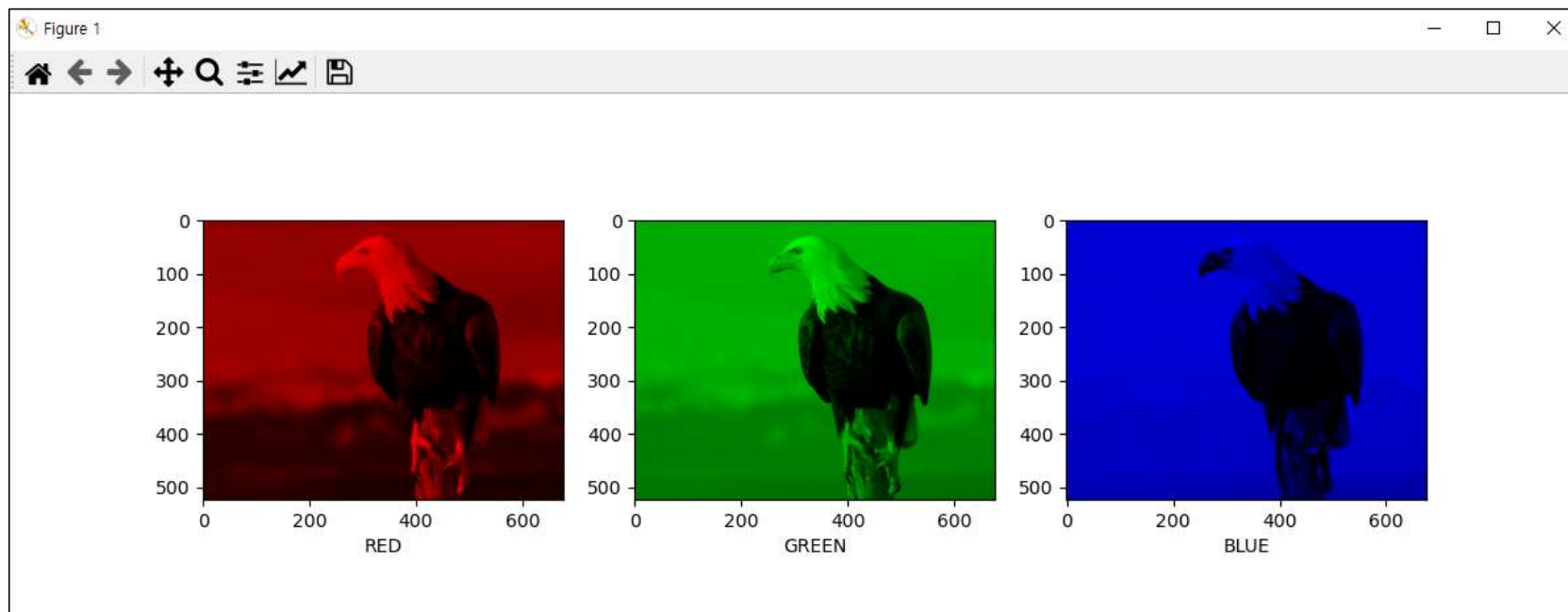
```
return img_red[:, :, RED], img_green[:, :, GREEN], img_blue[:, :, BLUE]
```

- ① Input1: 색상을 분리하고 싶은 컬러 이미지 데이터(3차원 배열) 입력
- ② Input2: 분리한 색상 별 그림을 표시할 것인지 여부 입력
- ③ Output: 컬러 이미지에서 채널 당 분리한 색상 값을 출력 (Red/Green/Blue)

Exercise 1

■ Exercise 1 Result

→ 코드를 실행하면, 다음과 같은 결과 창이 표시됨



Exercise 2

■ Webcam Real-time Reading

→ Logitech Webcam과 Pycharm을 연동하여 실시간으로 영상 출력

```
if __name__ == "__main__":
    # Exercise Environment Setting
    env = fl.libCAMERA()

    """ Exercise 1: RGB Color Value Extracting """
    ##### YOU MUST EDIT ONLY HERE #####
    # example = env.file_read("./Example Image.jpg")
    # R, G, B = env.extract_rgb(example, print_enable=True)
    # quit()
    #####

    # Camera Initial Setting
    ch0, ch1 = env.initial_setting(capnum=2)

    # Camera Reading..
    for i in range(EPOCH):
        _, frame0, _, frame1 = env.camera_read(ch0, ch1)

    """ Exercise 2: Webcam Real-time Reading """
    ##### YOU MUST EDIT ONLY HERE #####
    env.image_show(frame0, frame1)
    #####
```

Exercise 2

■ “initial_setting()” Description

→ PC에 연결된 카메라들을 Pycharm과 연동시키는 함수 (하드웨어 연결 설정)

```
def initial_setting(self, cam0port=0, cam1port=1, capnum=1):
```

④ ① ② ③

```
return channel0, channel1
```

- ① Input1: 카메라 0번에 대한 물리적 포트 번호 입력
- ② Input2: 카메라 1번에 대한 물리적 포트 번호 입력
- ③ Input3: PC에 연결한 카메라 개수 입력 (항상 2로 고정)
- ④ Output: 하드웨어 설정이 완료된 카메라의 채널 객체 정보 출력

Exercise 2

■ “camera_read()” Description

→ Webcam(Camera)을 이용하여 현재 순간의 프레임을 디지털 값으로 불러오는 함수

```
def camera_read(self, ①cap1, ②cap2=None):  
  
    for idx in range(0, self.capnum):  
        ret, frame = capset[idx].read()  
        result.extend([ret, frame])  
  
    ③return result
```

① Input1: 카메라 0번에 대한 채널 객체 정보 입력

② Input2: 카메라 1번에 대한 채널 객체 정보 입력

③ Output: capnum에 따라 Camera 채널마다 순간 프레임을 포착함

→ Camera가 두 대일 경우, 4가지 출력이 나옴 (*ret0*, **frame0**, *ret1*, **frame1**)

Exercise 2

■ “image_show()” Description

→ Webcam(Camera)을 이용하여 현재 순간의 프레임을 디지털 값으로 불러오는 함수

```
def image_show(①self, ②frame0, frame1=None):  
    if frame1 is None:  
        cv2.imshow('frame0', frame0)  
    else:  
        cv2.imshow('frame0', frame0)  
        cv2.imshow('frame1', frame1)
```

① Input1: 카메라 0번에 대한 순간 프레임 데이터 입력

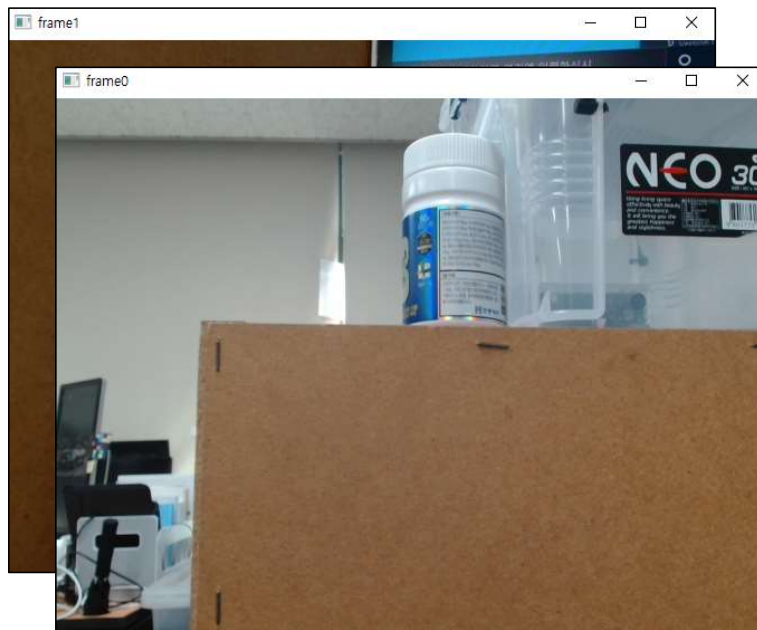
② Input2: 카메라 1번에 대한 순간 프레임 데이터 입력

→ Output은 따로 없음 (figure 창을 통해 실시간으로 순간 프레임을 출력함)

Exercise 2

■ Exercise 2 Result

- 코드를 실행하면, 다음과 같은 결과를 확인할 수 있음
- 영상 출력 도중, 소문자 'q'를 입력하면 프로그램을 종료할 수 있음



```
Run: main x
C:\anaconda3\envs\py39\python.exe D:/OpencvTest/main.py
OpenCV Version: 4.5.5
Camera Channel0 is enabled!
Camera Channel1 is enabled!
```

Exercise 3

■ Object Detection (Traffic Light Circle)

→ 신호등 예제 샘플을 출력하여 Webcam이 정확하게 인지하는지 확인

```
# Camera Reading..
for i in range(EPOCH):
    _, frame0, _, frame1 = env.camera_read(ch0, ch1)

    """ Exercise 2: Webcam Real-time Reading """
    ##### YOU MUST EDIT ONLY HERE #####
    # env.image_show(frame0, frame1)
    #####

    """ Exercise 3: Object Detection (Traffic Light Circle) """
    ##### YOU MUST EDIT ONLY HERE #####
    color = env.object_detection(frame0, sample=10, print_enable=True)
    #####
```


Exercise 3

■ “object_detection()” Description

→ 원하는 특정 색상의 물체(Object)만을 감지할 수 있는 함수 (신호등 색상 인식)

```
def object_detection(self, img, sample=0, mode="circle", print_enable=False):
```

①

②

③

④

- ① Input1: 카메라로 받은 순간 프레임 데이터 입력
- ② Input2: 신호등 색상을 인지할 때 사용하는 샘플의 개수 입력 (Hyper-parameter)
→ 원하는 값으로 변경해도 됨 (가급적 이미 맞춰져 있는 값을 사용할 것)
- ③ Input3: Hough 변환의 Mode 값 입력
→ 신호등의 원형 객체를 감지하기 위해 “circle” Mode로 설정
- ④ Input4: 출력 결과에 대한 그림과 색상 값을 표시할 것인지 여부 입력

Exercise 3

■ “object_detection()” Description

→ 원하는 특정 색상의 물체(Object)만을 감지할 수 있는 함수 (신호등 색상 인식)

```
        if count > sample / 2:
            result = COLOR[color]
            cv2.circle(replica, center, int(circle[2]), (0, 0, 255), 2)

    if print_enable:
        if result is not None:
            print("Traffic Light: ", result)
            self.image_show(replica)

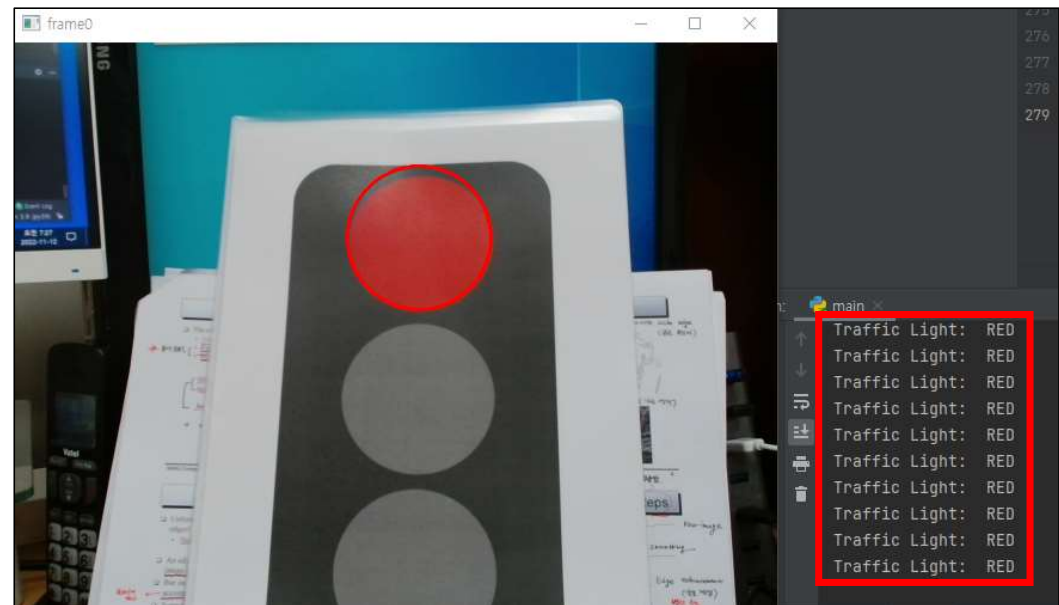
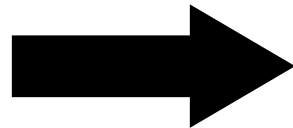
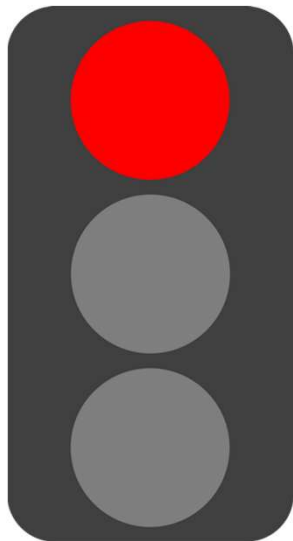
    ⑤ return result
```

⑤ Output: 물체 감지 과정에서 최종 결정된 색상 값(Red/Yellow/Green)을 출력
→ 각각 Red: 0 / Yellow: 3 / Green: 1로 설정되어 있음

Exercise 3

■ Exercise 3 Result

→ 신호등 예제 샘플에 대해 다음과 같은 결과를 얻을 수 있음



Exercise 4

■ Specific Edge Detection (Traffic Line)

→ 차선 예제 샘플을 출력하여 Webcam이 정확하게 인지하는지 확인

```
# Camera Reading..
for i in range(EPOCH):
    _, frame0, _, frame1 = env.camera_read(ch0, ch1)

    """ Exercise 2: Webcam Real-time Reading """
    ##### YOU MUST EDIT ONLY HERE #####
    # env.image_show(frame0, frame1)
    #####

    """ Exercise 3: Object Detection (Traffic Light Circle) """
    ##### YOU MUST EDIT ONLY HERE #####
    # color = env.object_detection(frame0, sample=16, print_enable=True)
    #####

    """ Exercise 4: Specific Edge Detection (Traffic Line) """
    ##### YOU MUST EDIT ONLY HERE #####
    direction = env.edge_detection(frame0, width=500, height=120,
                                   gap=40, threshold=150, print_enable=True)
    #####
```

Exercise 4

■ “edge_detection()” Description

→ 특정 Edge Line을 감지할 수 있는 함수 (차선 방향 인식)

```
def edge_detection(self, img, width=0, height=0, gap=0, threshold=0, print_enable=False):
```

①

②

③

④

⑤

⑥

- ① Input1: 카메라로 받은 순간 프레임 데이터 입력
- ② Input2: 관심영역[ROI]의 최대 가로 길이 입력
- ③ Input3: 관심영역[ROI]의 최소 세로 길이 입력
- ④ Input4: Pixel 분석에서 비교 대상과의 거리 차이 입력
- ⑤ Input5: Pixel 분석에서 특정 Edge Line을 구분하기 위한 길이 조건 입력
- ⑥ Input6: 출력 결과에 대한 그림과 방향 값을 표시할 것인지 여부 입력

Exercise 4

■ “edge_detection()” Description

→ 특정 Edge Line을 감지할 수 있는 함수 (차선 방향 인식)

```
if np.abs(grad) < FORWARD_THRESHOLD:
    prediction = FORWARD
elif grad > 0:
    prediction = RIGHT
elif grad < 0:
    prediction = LEFT

# real_lines.append([xa, ya, xb, yb])
cv2.line(replica, (xa, ya), (xb, yb), color=[0, 0, 255], thickness=2)
new_lines.append([xa, ya, xb, yb])

if print_enable:
    if prediction is not None:
        print("Vehicle Direction: ", DIRECTION[prediction])
        self.image_show(replica)

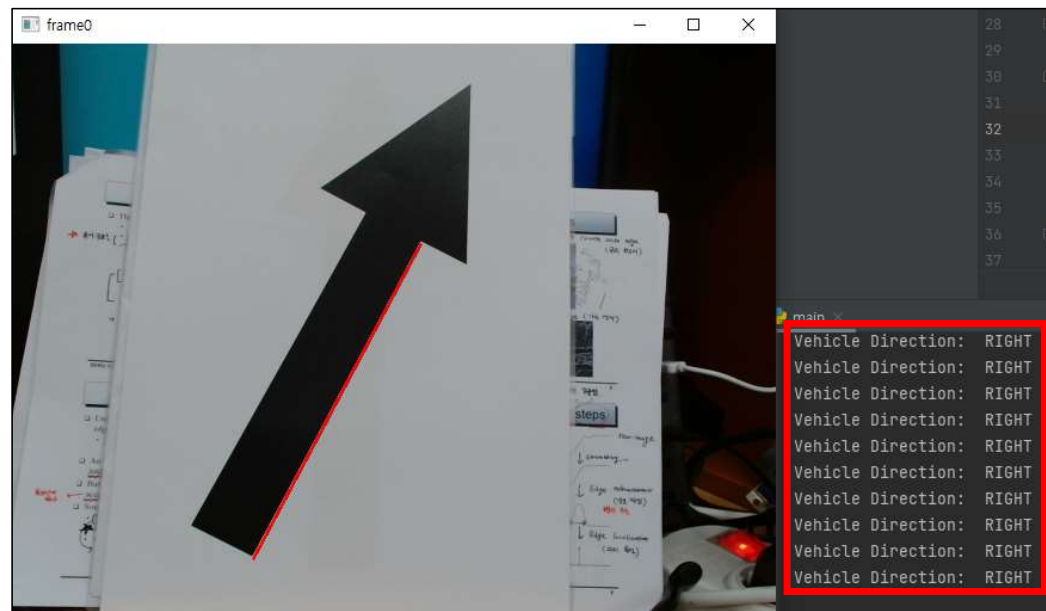
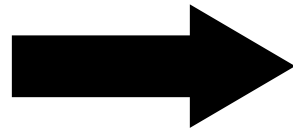
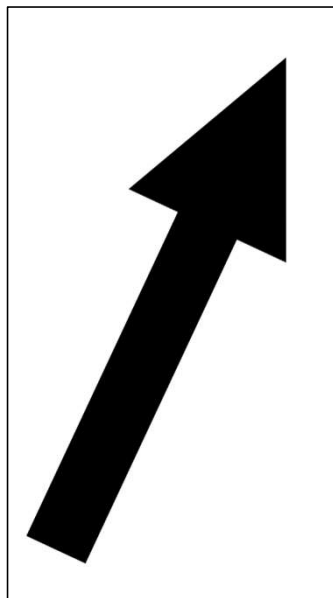
return prediction
```

⑦ Output: 특정 Edge Line에 대한 직선의 기울기로 차선 방향을 예측함
→ 정방향: FORWARD(0) / 우측: RIGHT(2) / 좌측: LEFT(1)

Exercise 4

■ Exercise 4 Result (1)

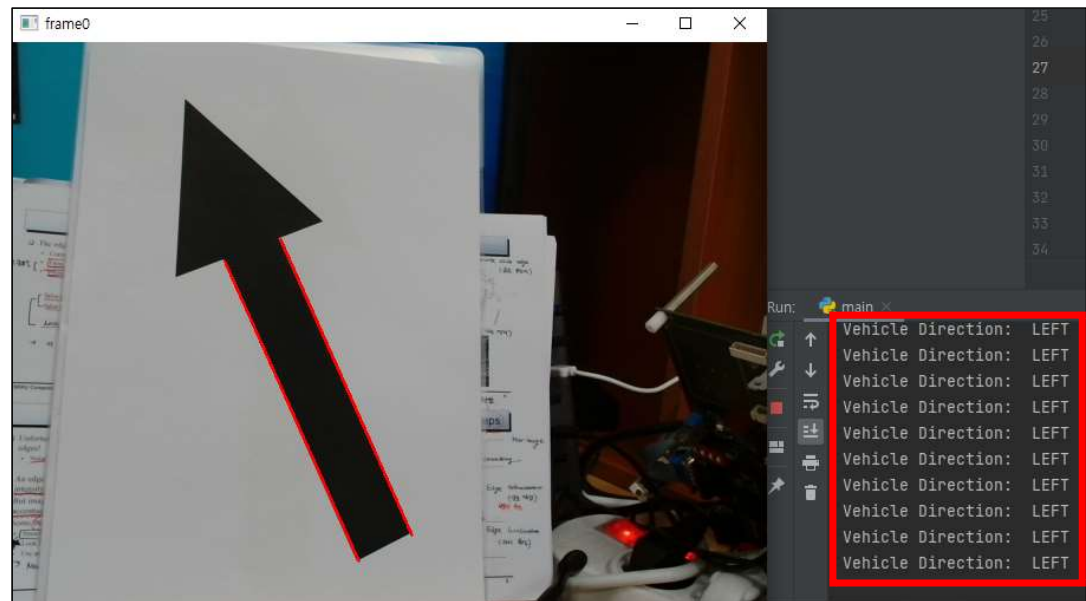
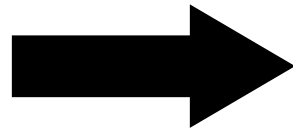
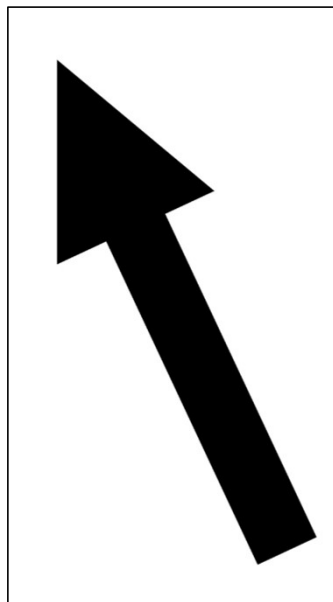
→ 차선 예제 샘플에 대해 다음과 같은 결과를 얻을 수 있음



Exercise 4

■ Exercise 4 Result (2)

→ 차선 예제 샘플에 대해 다음과 같은 결과를 얻을 수 있음



Thank You!

Automation Lab.

