



# Creating a RC CAR which follows certain object we want.

## ▼ 저자 소개



- DGIST에서 21학번에 재학중인 이석민 학생입니다.
- 과학기술전문사관 9기로써 졸업 후 국방과학연구소에서 3년동안 근무합니다.
- phone : 010-2026-0367
- Email : jhss4475@dgist.ac.kr
- Instagram : ps\_for\_human

## ▼ 프로젝트 진행 이유

- 2학년 때 인공지능기초 과목을 수강하고, 머신러닝과 딥러닝에 대해 이론적으로 꽤나 많은 지식들을 얻을 수 있었습니다.

- 하지만, 머신러닝이나 딥러닝을 이용한 실습들은 정말 기초적인 실습들 밖에 해보지 못했고, 배운 인공지능을 활용해서 제대로 된 실습을 해보고 싶었습니다.
- 그냥 실습이 아니라 재미있는 실습을 해보고 싶었고, 이에 “나만 따라오는 애완견 rc car”를 만들기로 생각했습니다.
- 대학생활동안 이론공부에 집중하였기에 한번도 프로젝트 경험이 없었습니다. 앞으로의 연구를 위해서도 꼭 프로젝트 경험이 한번은 필요하겠다고 생각했고, 이 생각이 프로젝트를 진행함에 있어서 강한 원동력이 되었다고 생각합니다.

## ▼ 소요시간, 필요한 준비물

- 저는 3주동안 거의 수업듣는 시간 빼고 전부를 투자했습니다!
- 시간이 없으시다면, 큰 프로젝트를 하기위한 작은 프로젝트들이 중간에 있어서, 그들을 위주로 공부하시면 됩니다. (RC CAR가 필요 없는 Object detection만을 연습하는 작은 프로젝트들도 있습니다.)
- RC CAR를 구입하기 위해서 20만원 정도가 필요합니다.
- 미니 프로젝트들은 노트북만 있으면 가능한 것들도 있습니다. (노트북 웹캠으로 Object detection해보기 등)

## ▼ 선수지식

- Python은 자유자재로 다룰 수 있어야 합니다. (DGIST 1학년 2학기 프로그래밍 수업을 열심히 들은 친구들이라면 충분합니다.)
- 딥러닝에 대한 지식이 있으면 좋습니다. 꼭 필요하지는 않습니다만, 알고있다면 프로젝트의 원리를 조금 더 깊게 이해할 수 있을 것입니다. 단, 모르더라도 무방합니다.
- 이외의 필요한 지식은 바로 밑에서 소개합니다.

## ▼ 기본지식 설명

- 이 Section에서는 기본적인 지식 획득이 목적이므로, 프로그램들을 실행하기 위한 자세한 컴퓨터의 환경 설정은 다루지 않았습니다. 추후 프로젝트 진행 설명 section에서 환경설정에 대해 자세히 다를 것이니, 라즈베리파이를 모니터와 키보드, 마우스와 연결해서 직접 켜보거나, putty등의 프로그램 실행은 프로젝트 진행 설명 section의 설명까지 보시고 해보시는 것을 강력하게 추천드립니다.

### 1. 라즈베리파이



- 라즈베리파이는 하나의 작은 컴퓨터라고 생각하면 됩니다.

- 우리가 데스크탑을 사면, (모니터, 키보드, 마우스, 본체)를 조립해서 사용하게 됩니다. 라즈베리파이는 이때 본체라고 생각하면 됩니다. 단, 크기를 많이 작게 만든 본체라고 생각하면 됩니다.
- 위에 초록색 라즈베리파이 사진을 보면, 라즈베리파이에 모니터를 연결할 수 있는 HDMI PORT와 (중앙 밑) 키보드와 마우스를 연결할 수 있는 USB port(오른쪽 밑)가 있는것을 관찰 할 수 있습니다.



- 위의 사진은 라즈베리파이에 모니터, 키보드, 마우스를 연결해서 우리가 아는 데스크탑과 똑같이 작동하는 모습입니다. 왼쪽 밑에 보면 라즈베리파이가 있는 것을 확인할 수 있습니다. 다시한번 말하지만, 라즈베리파이는 작은 본체라고 생각하면 됩니다.

## 2. RC CAR

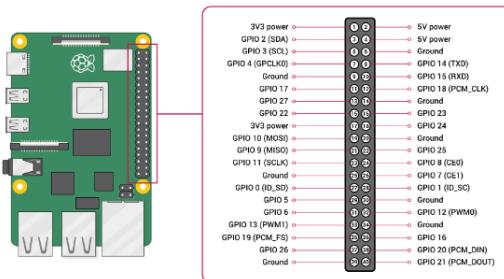


### Raspberry Pi Video Car Kit (Picar-V) for Intermediate

The PiCar-V is an open-source robot learning kit based on Raspberry Pi 4B. Equipped with a wide-angle USB webcam, it is powerful together with three whole new circuit boards and less but simpler wiring.

 <https://www.sunfounder.com/products/smart-video-car>

- 제가 사용한 RC CAR는 Picar - v라는 제품입니다. naver에서 국내배송으로 구입할 수 있습니다.
- Picar-V RC CAR의 모든 모터들의 키고 끔을 조절하는 것은 라즈베리파이입니다.
- 위의 RC CAR사진에서 가장 중앙에 있는 것이 라즈베리파이입니다.



- 라즈베리파이의 GPIO PIN에 모터들을 연결하고, GPIO PIN에 가하는 전압을 라즈베리파이가 조절하면서 GPIO PIN에 연결된 모터들을 키고 끌 수 있습니다.
- 라즈베리파이의 USB PORT 에 USB 카메라를 연결해서 라즈베리파이가 실시간으로 카메라에 찍히는 영상 데이터를 입력받을 수 있습니다.
- 이 말만 들으면 “와 이거 어떻게 구현하지” 싶을텐데요, 다행히도 파이썬으로 위 두줄의 일들을 굉장히 쉽게 해낼 수 있습니다.
- 예컨대 파이썬에서는 “goforward()” 이렇게 한줄만 입력해도, RC CAR가 앞으로 잘 갈 수 있도록 GPIO PIN의 전압을 알아서 라즈베리파이가 조절하여 모터를 동작시킵니다.
- 예컨대 파이썬에서는 “cv.imshow(video)” 이렇게 한줄만 입력해도, RC CAR에 연결된 USB카메라가 찍고있는 화면을 바로 모니터에 보여줍니다.
- 정리해볼까요? 만약 RC CAR에 연결된 라즈베리파이에 모터, 모니터와 키보드, 마우스를 연결해서 전체 컴퓨터를 킨 후에, 컴퓨터에서 밑에와 같은 python code를 실행시켰다고 생각해봅시다! (실제 코드가 아닌, 간단화 된 예시 코드입니다!)

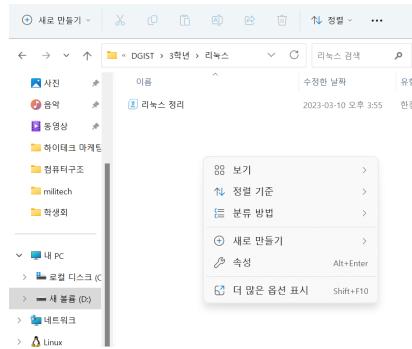
```
goforward()
cv.imshow(video)
```

- 실행시킨다면, 라즈베리파이에 연결된 USB카메라가 찍고있는 화면을 바로 모니터에 보여주고, GPIO PIN에 연결된 모터가 작동하여 바퀴가 앞으로 구르고 있을 것입니다.
- 즉 라즈베리파이에서 위와 같은 파이썬 코드를 실행시켜서 rc car를 조종할 수 있는 것입니다.
- 그런데, 위에서 소개한 방법에 의하면, 모터를 돌리기 위해서 라즈베리파이에 모니터, 키보드, 마우스 등을 연결해야 하는데요(파이썬 프로그램을 돌리고 싶으면 모니터로 당연히 보면서 실행시켜야 하니까) 그러면 RC CAR가 모니터와 연결된 상태로만 조종될 수 있다는 것과 같습니다.
- 당연히 모니터, 키보드, 마우스 없이 rc카 혼자서도 잘 주행할 수 있도록 해야 될 것입니다.
- 그런데 모터를 조종하려면 RC카의 라즈베리파이에서 파이썬 프로그램을 실행시켜야하는데 어떻게 이것이 가능할까요? 두 칸 밑에서 설명할테니 걱정 마세요!

### 3. 우분투

- 이번 프로젝트에서는 라즈베리파이의 운영체제로 우분투를 쓰게 됩니다.

- 우분투는 윈도우 같은 운영체제입니다.
- 윈도우 운영체제를 사용할 때에 마우스 클릭으로 했던 일들을 우분투에서는 command shell에 명령어를 입력함으로써 할 수 있습니다.
- 예시를 들자면 윈도우 운영체제를 사용하는 컴퓨터에서는 폴더를 만들기 위해서 다음과 같은 일을 하면 됩니다.



- 화면에 마우스 오른쪽 키를 누르고 “새로만들기”를 클릭하면 됩니다.

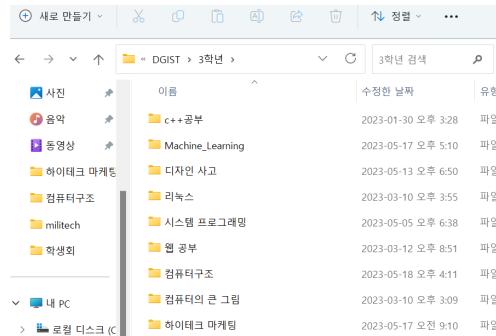
```
seokmin@DESKTOP-C76SAEC ~
이제 Linux용 Windows 하위 시스템을 Microsoft Store에서 사용할 수 있습니다.
'wsl.exe --update'를 실행하거나 https://aka.ms/wslstorepage를 방문하여 업그레이드할 수 있습니다.
Microsoft Store의 WSL을 설치하면 최신 WSL 업데이트가 더 빠르게 제공됩니다.
자세한 내용은 https://aka.ms/wslstoreinfo를 방문하세요.

seokmin@DESKTOP-C76SAE0:~$ mkdir seokmin|
```

- 이를 우분투에서 실행한다면 다음과 같이 해야합니다.
- 다음의 코드는 seokmin이라는 이름을 가진 폴더를 만드는 명령어입니다.
- 이와 같이 윈도우에서 마우스로 하던 일들을 우분투에서는 명령어로 해야한다는 차이가 있습니다.
- 그러면 우분투의 명령어들을 조금 공부해 봅시다.

```
seokmin@DESKTOP-C76SAEC ~
이제 Linux용 Windows 하위 시스템을 Microsoft Store에서 사용할 수 있습니다.
'wsl.exe --update'를 실행하거나 https://aka.ms/wslstorepage를 방문하여 업그레이드할 수 있습니다.
Microsoft Store의 WSL을 설치하면 최신 WSL 업데이트가 더 빠르게 제공됩니다.
자세한 내용은 https://aka.ms/wslstoreinfo를 방문하세요.

seokmin@DESKTOP-C76SAE0:~$ ls
assign1 assign3 assign4 assign_real sysp sysprojec systemprogramming
```

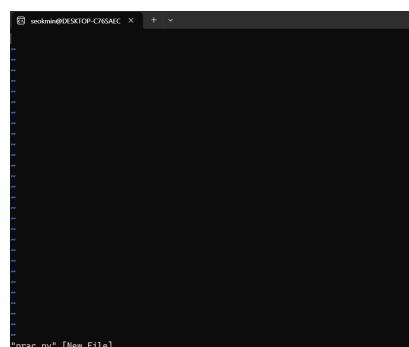


- ls : 현재 폴더에 있는 하위 폴더들과 파일들을 출력해줍니다. 윈도우에서는 그냥 폴더에 들어가있으면 화면에 나오는 것에 비해 리눅스에서는 ls라는 명령어를 입력해줘야 현재 폴더 안에 있는 파일들과 폴더들을 출력해줍니다.

```
seokmin@DESKTOP-C76SAEO:~$ ls
assign1 assign3 assign4 assign_real sysp sysprojec systemprogramming
seokmin@DESKTOP-C76SAEO:~$ cd sysp
seokmin@DESKTOP-C76SAEO:~/sysp$ |
```

- cd: 해당 폴더로 이동합니다. 윈도우에서는 이동하고 싶은 폴더를 더블클릭 하는 것에 비해, 리눅스에서는 “cd 이동하고싶은 폴더 이름”으로 명령어를 입력해줘야 합니다.
- 맨 마지막줄을 보면, “/sysp”가 seokmin@DESKTOP-C76SAEO: 다음에 추가된 것을 볼 수 있는데 이는 sysp폴더로 이동해있다는 표시입니다.

```
seokmin@DESKTOP-C76SAEO:~/sysp$ vi prac.py|
```



- vi 파일이름.py : 파일이름.py를 작성할 수 있는 vi라는 프로그램을 열어줍니다. 윈도우에서 VSCODE를 여는 것과 같은 느낌입니다. prac.py라는 python파일을 수정할 수 있는 프로그램이 열린 것을 볼 수 있습니다. 이 프로그램에서는 키보드 ‘i’를 누르면 코드 작성을 시작할 수 있고, 작성이 끝나면, “esc”를 누른 후에 “:wq”를 누르면 해당 파일을 저장하고 프로그램을 종료할 수 있습니다.

```
seokmin@DESKTOP-C76SAEO:~/sysp$ python prac.py|
```

- python 파일이름.py : 파이썬 파일을 실행하는 코드입니다. vi라는 프로그램에서 파이썬 파일을 바로 실행할 수 없습니다. 대신, command shell에서 “python 파일이름.py” 명령어를 실행하여 python 파일을 실행시킬 수 있습니다.
- Python 파일의 실행을 중간에 멈추고 싶다면 command shell에서 “control + c”를 누르면 됩니다.

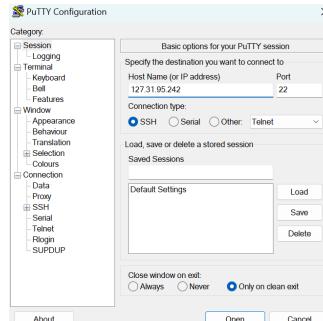
#### 4. Putty

- 그렇다면 라즈베리파이에 모니터와 마우스, 키보드를 연결한 다음, 라즈베리파이의 command shell에서 여러 명령어들을 통해 python프로그램을 짜고, 실행시킨다면 모터가 움직일 것입니다.
- 하지만 우리의 목표는 RC CAR의 라즈베리파이에 모니터와 키보드, 마우스를 연결시키지 않고 라즈베리파이 내에 있는 파이썬 프로그램을 실행시켜서 RC CAR가 움직일 수 있도록 하는 것입니다.
- 이것은 Putty라는 프로그램을 통해서 가능합니다.
- Putty라는 프로그램이 해주는 일은, 라즈베리파이와 제 노트북이 같은 와이파이에 연결되어있다면, 라즈베리파이의 command shell을 제 노트북에서 열어서 명령어를 입력할 수 있게 해줍니다.
- 오직 필요한 것은 라즈베리파이의 wifi ip address가 필요합니다.
- 같은 wifi라도, 연결한 기기에 따라서 ip address가 다릅니다.
- 이를 위해서 처음에는 라즈베리파이에 모니터와 키보드, 마우스를 연결하여 command shell을 실행시킨 후 “ifconfig” 명령어를 입력하여 라즈베리파이의 wifi ip address를 얻습니다.
- 이제 라즈베리파이에서 모니터와 키보드, 마우스를 분리합니다.

```
sekomin@DESKTOP-C76SAEO:~$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
        inet 172.31.95.242 brd 255.255.240.0 broadcast 172.31.95.255
              netmask 255.255.240.0
        ether 00:15:5d:5e:d5:70 txqueuelen 1000 (Ethernet)
              RX packets 240 bytes 229497 (229.4 KB)
              RX errors 0 dropped 0 overruns 0 frame 0
              TX packets 113 bytes 7786 (7.7 KB)
              TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

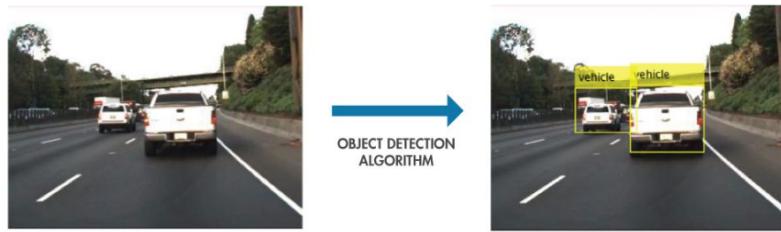
lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
        inet 127.0.0.1 brd 255.0.0.0
              netmask 255.0.0.0
        loop txqueuelen 1000 (Local Loopback)
              RX packets 0 bytes 0 (0.0 B)
              RX errors 0 dropped 0 overruns 0 frame 0
              TX packets 0 bytes 0 (0.0 B)
              TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

- inet뒤에 적힌 127.31.95.242가 바로 wifi ip address입니다.
- 이렇게 라즈베리파이의 wifi ip address를 얻은 후 노트북에서 putty앱을 실행합니다.



- 첫 화면에서 Host Name(or IP address)칸에 라즈베리파이의 wifi address를 입력하고, 맨 밑에 “Open”을 클릭하면 됩니다.
- 그 후 라즈베리파이의 username과 비밀번호를 입력하면 라즈베리파이의 command shell을 노트북에서 이용할 수 있습니다.
- 이를 통해 라즈베리파이에 키보드와 모니터 등을 연결하지 않고 노트북에서 라즈베리파이의 파이썬 프로그램을 실행시켜 RC CAR가 움직일 수 있도록 할 수 있습니다.

## 5. Object Detection



- Object Detection이란, 하나의 input 사진에서 사용자가 원하는 Object를 네모칸을 쳐 주고 그 Object가 무슨 종류인지 알려주는 것입니다. 위에 사진을 보면 이해할 수 있을 것입니다.
- Object Detection Algorithm이란, 하나의 사진을 input으로 받고, output으로 사용자가 원하는 object에 네모칸을 친 사진을 return해주는 “함수”를 의미합니다.
- 우리는 하나의 사진을 input으로 받고, 그 사진에서 사람을 네모칸 친 사진을 return해주는 “YOLO”라는 함수이자 Object Detection Algorithm을 사용할 것입니다.

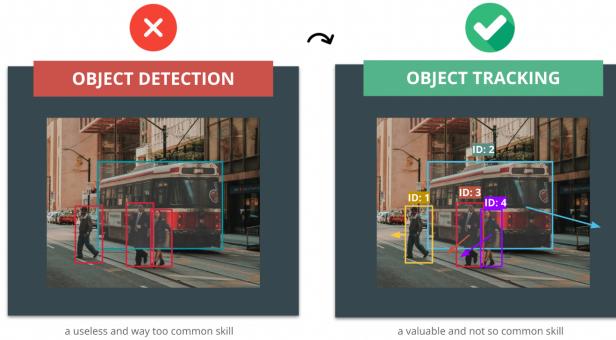
## 6. Video

- 영상은 여러 사진들을 sequential하게 보여주는 것일 뿐입니다.
- 영상에서 object detection을 한다는 것은 사진들을 sequential하게 object detection한다는 것과 같습니다.
- 영상에서 1초당 보여주는 사진의 개수를 frame이라고 합니다.
- frame이 높을수록 영상이 끊어져서 보이는 현상이 줄어들고, frame이 낮을수록 영상이 끊겨져 보이게 됩니다.

## 7. Object tracking

- Object tracking: 영상에서 Object detection을 할 때에 즉 여러 사진을 sequential하게 object detection을 할 때에, 이전 사진에서 detection했던 물체가 현재 사진에서 detection한 사람과 동일 인물인지의 여부를 고유번호를 부여해줌으로써 알려주는 것을 말합니다.
- 영상을 Object detection만 하게 되면 영상에서 즉 sequential한 사진에서 사람을 계속 detection(네모박스를 쳐주는 것)할 뿐, 이전 사진과 현재 사진의 사람이 동일인물인지의 여부는 알 수 없습니다.

- 반면, Object tracking을 하게 되면 이전 사진에서의 네모박스를 친 사람과 현재 사진의 네모박스를 친 사람이 동일인물인지의 여부를 알 수 있습니다.
- 밑의 그림을 보면, detection만 했을 때에는 네모박스 위에 사람이라고만 나오는 반면(밑의 그림에서 사람글씨는 생략되었습니다), 보통 detection한 물체의 종류는 알려줍니다.) , tracking을 했을 때에는 이전 사진과 현재 사진의 사람이 같은 사람인지 확인하기 위해서 사람 위에 고유 식별 번호가 달려있는 것을 확인할 수 있습니다.



## 8. Network programming ( Socket programming)

- 많은 프로젝트들을 하다보면, 두 기기가 정보를 주고받아야 하는 일이 발생합니다. 우리의 프로젝트도 마찬가지입니다.
- 우리의 프로젝트에서는 라즈베리파이와 노트북이 정보를 주고받아야 합니다. (이유는 “프로젝트 아이디어 - 보충 Section”에 자세히 설명되어 있습니다.)
- 이것은 Python에서 구현이 가능합니다.
- 예시로 빠르게 이해해보겠습니다.
- 라즈베리파이가 실시간으로 영상을 찍고, 그 영상을 노트북에게 보내서, 노트북이 영상을 노트북 모니터에 띄어주는 코드를 구현해봅시다. (실제 코드가 아닌, 간단화 된 예시 코드입니다.)

`raspberrypi.py`

```
while True:
    screen = usb_camera()
    send(notebook_ip_address, screen)
```

`laptop.py`

```
while True:
    screen = receive(raspberry_pi_ip_address)
    show(screen)
```

- 이와 같이 라즈베리파이와 노트북에 각각 파이썬 코드를 짜놓고, 각각의 코드를 실행시킨다면 라즈베리파이에서 실시간으로 찍은 영상을 노트북에서 볼 수 있을 것입니다.
- 라즈베리파이의 파이썬 코드는 노트북에서 Putty를 이용하여 실행시키고, 노트북의 파이썬 코드는 노트북에서 VS CODE 등을 열어서 실행시키면 노트북과 라즈베리파이 각각에서 모두 파이썬 코드가 실행 됩니다.

### ▼ Threading

- 한 python 파일에서 동시에 코드 두개를 돌리는 technic.
- 밑의 수도코드를 실행하면, 1과 0이 번갈아가며 출력된다.

```
def thread():
    while True:
        print(0)
    threading(thread).start()

while True:
    print(1)
```

- thread함수 안의 내용과 main내용(print(1)하는 while문)이 동시에 실행된다.

### ▼ 전체적인 프로젝트 아이디어

- RC CAR의 라즈베리파이의 USB Port에 연결된 카메라로 RC CAR의 시선영상을 찍습니다. 이영상을 찍히는대로 실시간으로 Object tracking합니다.
- 화면에 있는 사람들 중에 따라가고 싶은 사람의 ID를 입력합니다.
- 따라가고 싶은 사람의 화면상의 좌표와 화면 중간의 좌표의 차이를 토대로 우회전 또는 좌회전을 합니다.
- 이 과정들을 Python 프로그램을 짜 놓고, 실행시킴으로써 구현할 수 있습니다.
- 예시를 보시겠습니다. (매우 간략화된 코드입니다. 실제 코드가 아닙니다.)

example.py

```
while True:
    screen = usb_camera()
    tracking_result = object_tracking_algorithm(screen)
    if(tracking_result.center_point_x > screen.center_point_x):
        turn_right()
    else:
        turn_left()
```

- 위와 같은 프로그램을 짜 놓고 실행시켜 놓는다면, 라즈베리파이는 사람을 따라갈 수 있을 것입니다.
- 노트북에서 Putty를 통해 라즈베리파이의 위와같은 파이썬 파일을 실행시키는 것입니다.(라즈베리파이는 모니터와 키보드등의 연결 없이)
- 그렇다면 우리가 원하는 RC CAR가 만들어질 것입니다.

## ▼ 프로젝트 아이디어 - 보충

- “전체적인 프로젝트 아이디어 section”의 아이디어는 완벽해 보입니다.
- 하지만 이 아이디어에는 부족한 점이 있습니다.
- Object tracking은 많은 행렬연산을 필요로 합니다.
- 라즈베리파이는 크기를 많이 줄인 본체이므로, 계산 능력이 많이 떨어집니다.
- 따라서 라즈베리파이에서 object tracking을 한다면 tracking 계산시간이 너무 오래 걸리게 됩니다.
- 이대로 진행한다면 tracking computation delay가 10초정도 걸리게 됩니다.
- 10초전의 사진을 토대로 모터가 움직이는(좌회전 또는 우회전) 일이 벌어집니다.
- 이를 어떻게 해결할 수 있을까요?
- 이는 다음과 같은 방법으로 해결이 가능합니다.
- 라즈베리파이에서는 최소한의 연산만 수행 합니다.
- 필요한 복잡한 연산은 계산에 필요한 data를 라즈베리파이로부터 노트북으로 보내서 노트북에서 수행합니다.
- 연산 결과를 노트북에서 라즈베리파이로 재전송합니다.
- 우리의 프로젝트에 맞게 말하면, 라즈베리파이의 USB카메라에서 찍은 영상을 노트북으로 보내서, 노트북에서 object tracking을 수행하고, tracking결과에 따라 좌회전과 우회전 여부를 결정한 후에, 좌회전 또는 우회전 여부만 라즈베리파이에 다시 보내주는 것입니다.
- 이렇게 한다면, 무거운 연산들은 노트북에서 수행해주고, 라즈베리파이는 최소한의 연산만 수행할 수 있습니다.
- 위에서 배웠던 Network programming(Socket Programming)을 통해서 노트북과 라즈베리파이는 정보를 주고받을 수 있습니다.

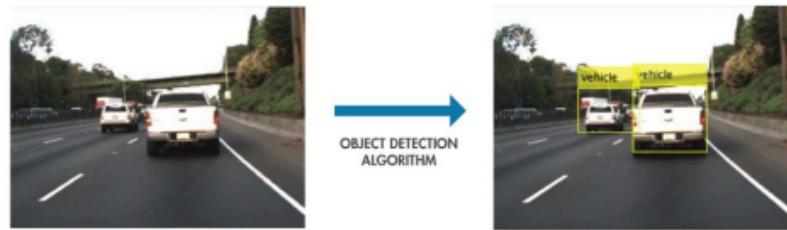
## ▼ 주의할 점

- 프로젝트를 진행하면서 발생하는 거의 모든 에러는 YOLO Algorithm과 python, pytorch, torchvision 등 등 여러 모듈의 버전 호환 문제로 인해서 발생합니다. 꼭 버전 호환에 유의해주시기 바랍니다.
- 제가 밑에 써놓은 버전 말고도 여러 호환되는 버전이 존재하기는 하기때문에 본인의 컴퓨터에 설치되어 있는 모듈들끼리 호환만 되면 꼭 제가 추천하는 버전들이 아니어도 동작이 가능합니다.
- 그럼에도 불구하고, 호환되는 버전 정보를 찾기 매우 매우 힘들기 때문에, 추천하는 버전대로 진행하시는 것을 추천합니다.
- 분명 제 컴퓨터의 환경과 여러분들의 컴퓨터의 환경이 다를 수 있습니다. 예컨대 저는 명령 프롬프트에서 사용하는 pip명령어를 사용할 수 있도록 설정해놓았습니다. 반면, 파이썬을 처음 설치하고 나면 pip명령어를 사용하기 위해서는 여러 설정이 필요할 것입니다. 이와 같이 환경설정으로 인해 많은 에러가 발생할 수 있습니다. 에러가 발생하신다면 에러를 chat gpt에게 물어보거나, 구글 검색을 통해서 해결해보세요!
- 제가 참고한 영상과 사이트들은 맨 밑에 참고자료 section에 첨부해놓았습니다!

## ▼ 프로젝트 진행 설명

### ▼ Object Detection 해보기

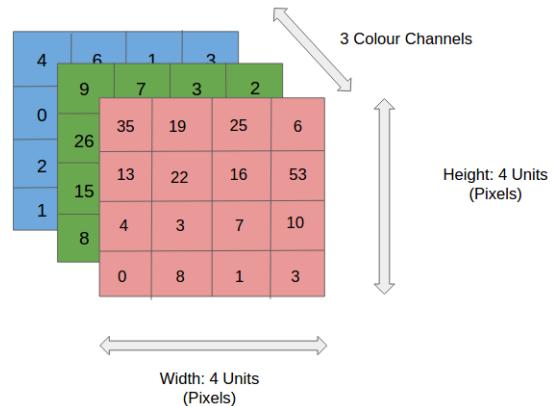
- 자 그러면, 처음으로 같이 Object detection부터 시작해 봅시다!
- Object detection이란 사진에서 user가 원하는 object를 찾아 네모박스 쳐주는 것을 말합니다.
- 조금 더 프로그래머의 관점에서 이해해 볼까요?
- Object detection(알고리즘)이란, 함수이며, input으로 사진을 받아서 output으로 input 사진에 특정 object를 찾아 네모박스를 친 사진을 return하는 것을 말합니다.



- 즉, Object detection algorithm을 Python을 사용해서 구현한다고 생각한다면, skeleton code(sudo code)는 밑의 코드와 같을 것입니다.

```
def Object_detection_algorithm(input_picture):
    #사진 가공
    ~~~~~
    ~~~~~
    return output_picture
```

- 코드가 너무 추상적이기 때문에 조금만 더 설명하자면, input\_picture은 3차원 배열일 것입니다. R,G,B 각각이 2차원을 갖는 실수 배열일 것입니다. 밑의 사진을 참고한다면 조금 더 깊게 이해할 수 있을 것입니다.



- 즉 저 3차원 배열을 지지고 봄아서 output\_picture이라는 다른 3차원 배열을 return하는 함수인 것입니다.
- 말만 들어도 상당히 구현하기 복잡할 것이라고 생각할 것입니다. 맞습니다. 상당히 복잡합니다. 이는 딥러닝기술을 통해 구현할 수 있습니다. 우리가 기억해야 할 것은 Object detection algorithm은 input과 output이 모두 사진인 함수라는 것입니다.
- 다행히도 이 함수를 미리 구현해놓은 회사가 있습니다. 이 회사의 이름은 ultralytics이며, input사진에서 사람, 핸드폰, TV, 마우스등 80여종의 물체에 네모박스를 친 후 ouput으로 결과 사진을return 해주는 Object detection 알고리즘을 개발하였습니다.
- ultralytics는 방금 소개한 함수이자 Object detection 알고리즘의 이름을 “YOLO”라고 명명하였습니다.
- 그러면 한번 YOLO Algorithm을 사용해 볼까요?
- 이제부터 설명한 내용은 밑의 유튜브의 내용을 요약하고 조금 변형한 것입니다. 그러나 궁금한 점이 생기면 밑의 유튜브 채널을 참고해보세요!

Complete YOLO v8 Custom Object Detection Tutorial | Windows & Linux  
A complete YOLO v8 custom object detection tutorial with two-classe custom dataset. I cover how to annotate custom dataset in YOLO format, setting up environment for YOLO v8, and train custom object detection model. Although I  
[https://www.youtube.com/watch?v=gRAyOPjQ9\\_s](https://www.youtube.com/watch?v=gRAyOPjQ9_s)

- 실습을 위해서는 컴퓨터에 파이썬과 YOLO알고리즘이 설치되어 있어야 할 것입니다.
- 파이썬은 3.8 Version으로 설치해주세요.
- YOLO 알고리즘은 ultralytics라는 모듈 내에 있기 때문에, ultralytics모듈을 다운로드 합니다.  
(version은 8.0.10으로 다운로드 합니다.)
- ultralytics 모듈은 명령 프롬프트에서 다음과 같은 명령으로 다운로드 할 수 있습니다.

```
명령 프롬프트
X + 
Microsoft Windows [Version 10.0.22621.1848]
(c) Microsoft Corporation. All rights reserved.

C:\Users\USER>pip install ultralytics==8.0.10
```

- 밑의 그림을 보면, 성공적으로 ultralytics module이 설치된 것을 확인할 수 있습니다.

```
🐍 versioncheck.py
```

```
1 import ultralytics  
2 print(ultralytics.__version__)
```

8.0.10

- 그러면 한번 YOLO algorithm을 이용하여 object detection을 실행해봅시다!
- 먼저 컴퓨터에 있는 사진 또는 동영상에서 특정 물체를 감지하여 특정 물체에 네모박스를 친 사진을 출력하는 활동을 해보도록 합시다.
- 밑의 코드는 “video\_sample\_edit.mp4”라는 영상파일에서 object detection을 실행시켜 새로운 동영상파일을 만들어주는 코드입니다.
- 주의할 점은 파이썬 코드가 속해있는 폴더와 같은 폴더에 동영상 파일이 속해있어야 합니다.
- 코드에서 사용한 model은 default model이므로, yolo github에서 “yolov8m.pt”을 사용해 감지할 수 있는 물체들을 확인해볼 수 있습니다. (유튜브를 보면 custom model을 만들어낼 수도 있습니다.)

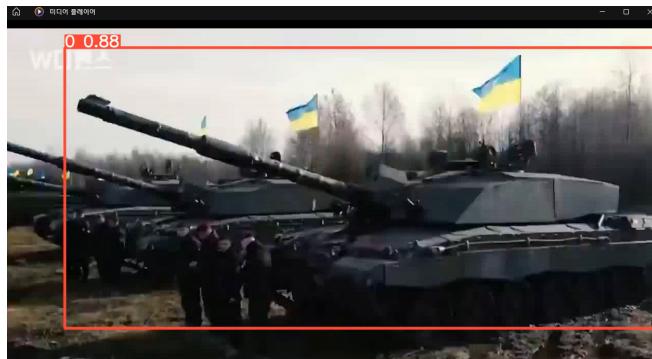
<https://github.com/ultralytics/ultralytics>

```
from ultralytics import YOLO  
model = YOLO("yolov8m.pt") #ultralytics에서 제공하는 default model  
model.predict(source="video_sample_edit.mp4", show = True, save = True)
```

- 위 코드를 실행시키고 나면, 새로운 영상파일이 만들어집니다.
- 본래 영상의 0초 부분은 밑의 사진과 같았습니다.



- 하지만 코드가 실행된 후 새로 저장된 동영상 파일을 보면, 다음과 같이 탱크에 네모박스가 쳐져있는 것을 알 수 있습니다.



- 성공적으로 실습에 성공한 것을 알 수 있습니다!!
- 이번에는 한번 실시간으로 노트북의 웹캠에서 사람, 핸드폰 등등을 감지해볼까요?
- 이번 내용은 밑의 유튜브에서 참고했습니다.

Real-Time Object Detection with YOLOv8 and Webcam: Step-by-step Tutorial  
In this video we are going to Deploy a YOLOv8 Object Detection Model in Python.  
We will see how to deploy a trained YOLOv8 model and run live inference on a webcam. This can actually run in real-time. You can also check out my YOLOv7 Course  
<https://www.youtube.com/watch?v=IHbJcOex6dk>



- 우리는 이제 실시간으로 노트북의 웹캠에서 사람, 핸드폰 등등을 감지해줄 것입니다.
- 감지 가능한 object들의 목록은 yolo v8 github에 공시되어 있습니다.

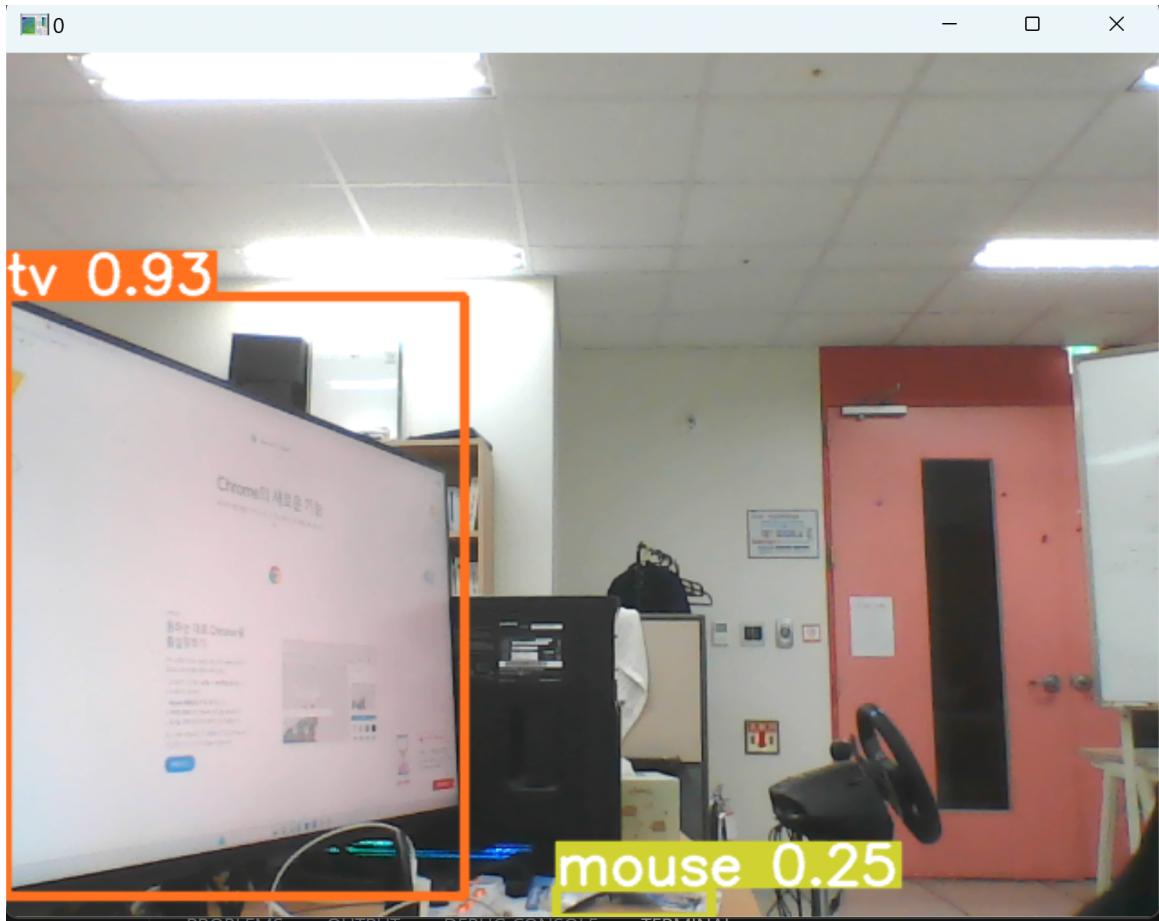
<https://github.com/ultralytics/ultralytics>

- 코드는 다음과 같습니다. 그저 source에 0을 대입한 것 말고는 위의 코드와 차이가 없는 것을 확인할 수 있습니다.

```
from ultralytics import YOLO

model = YOLO("yolov8m.pt")
result = model.predict(source=0 ,show=True)
```

- 코드를 실행시키면 real-time 동영상이 출력되는 것을 확인할 수 있습니다.



- 성공적으로 실습에 성공했습니다!

#### ▼ Object Tracking 해보기

##### Track & Count Objects using YOLOv8 ByteTrack & Supervision

Let's build together an application to track and count objects using Computer Vision. We used YOLOv8 for detection, ByteTrack for tracking, and the latest python library from Roboflow - Supervision for object counting.

 <https://www.youtube.com/watch?v=OS5ql9YBkfk>

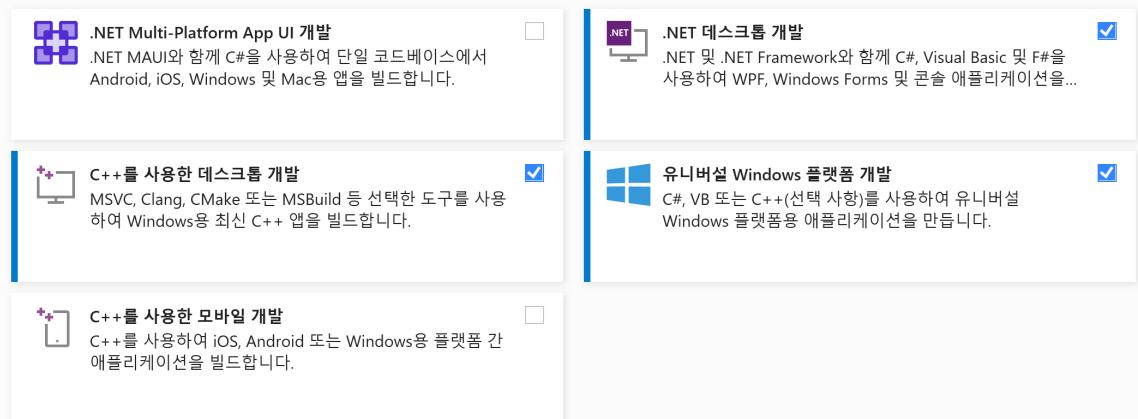


- 동영상은 사진의 연속이라고 설명했었습니다.
- 동영상에서의 Object detection은 이전 사진과 이후 사진의 어떤 물체가 동일한 물체인지에 대한 정보를 얻어낼 수 없습니다. 즉 이전 사진과 이후 사진의 어떤 물체가 동일한 물체인지에 대해서 연산을 수행하지 않습니다.
- 이전 사진과 이후 사진의 어떤 물체가 동일한 물체인지를 알기 위해서는 Object tracking 기술을 사용해야 합니다. 즉 이전 사진과 이후 사진의 어떤 물체가 동일한 물체인지에 대해서 연산을 수행해야 합니다.

- 코드는 아래와 같습니다. 여러 인터넷 검색과 동영상들을 토대로 발생하는 에러를 고치며 만들었습니다.
- error 1: visual studio c++를 설치하지 않아서 발생함.
- error 2: sys.append == 환경변수 편
- 밑의 3개가 잘 설치되어 있는지 확인해야 함.
- yolox 여기에 추가 해야댐.

```
C: > Users > USER > AppData > Local > Programs > Python > Python38 > Lib > site-packages > yolox > __init__.py > ...
```

#### 데스크톱 및 모바일 (5)



#### 게임 (2)

- 코드를 돌리기 위해서 여러 모듈들이 필요합니다. 필요한 모듈들을 다운로드 한 후 코드를 실행시키기 바랍니다.
- CLASS\_ID = [0, 64, 67, 73] 이 코드는 tracking 하고 싶은 object의 class를 의미합니다. 0은 person을 의미합니다. 숫자들과 class의 matching 정보는 yolo github에서 찾아볼 수 있습니다.
- 저도 코드를 완벽히 이해하면서 코드를 짜지 못했습니다. 여러분도 가능한 선까지만 이해하시기를 바랍니다!
- 저도 그저 여러 사이트에서 긁어온 코드를 이해할 수 있는 선에서 적절하게 혼합한 후에 에러가 발생하면 chat gpt에게 도움을 받는 식으로 코드를 구성했습니다. 거의 2~3일 정도 내내 혼합하고 에러를 고치고 했던 것 같습니다.

```
from yolox.tracker.byte_tracker import BYTETracker, STrack
from onemetric.cv.utils.iou import box_iou_batch
from dataclasses import dataclass
from supervision.draw.color import ColorPalette
from supervision.geometry.dataclasses import Point
from supervision.tools.detections import Detections, BoxAnnotator
from supervision.tools.line_counter import LineCounter, LineCounterAnnotator
from typing import List
```

```

import numpy as np
from ultralytics import YOLO
import cv2

@dataclass(frozen=True)
class BYTETrackerArgs:
    track_thresh: float = 0.25
    track_buffer: int = 30
    match_thresh: float = 0.8
    aspect_ratio_thresh: float = 3.0
    min_box_area: float = 1.0
    mot20: bool = False

# converts Detections into format that can be consumed by match_detections_with_tracks function
def detections2boxes(detections: Detections) -> np.ndarray:
    return np.hstack((
        detections.xyxy,
        detections.confidence[:, np.newaxis]
    ))


# converts List[STrack] into format that can be consumed by match_detections_with_tracks function
def tracks2boxes(tracks: List[STrack]) -> np.ndarray:
    return np.array([
        track.tlbr
        for track
        in tracks
    ], dtype=float)


# matches our bounding boxes with predictions
def match_detections_with_tracks(
    detections: Detections,
    tracks: List[STrack]
) -> Detections:
    if not np.any(detections.xyxy) or len(tracks) == 0:
        return np.empty((0,))

    tracks_boxes = tracks2boxes(tracks=tracks)
    iou = box_iou_batch(tracks_boxes, detections.xyxy)
    track2detection = np.argmax(iou, axis=1)

    tracker_ids = [None] * len(detections)

    for tracker_index, detection_index in enumerate(track2detection):
        if iou[tracker_index, detection_index] != 0:
            tracker_ids[detection_index] = tracks[tracker_index].track_id

    return tracker_ids

model = YOLO("yolov8x.pt")
CLASS_NAMES_DICT = model.model.names
CLASS_ID = [0, 64, 67, 73]
byte_tracker = BYTETracker(BYTETrackerArgs())

cap = cv2.VideoCapture(0)
# settings
LINE_START = Point(50, 1500)
LINE_END = Point(3840-50, 1500)

# Set the resolution of the video

```

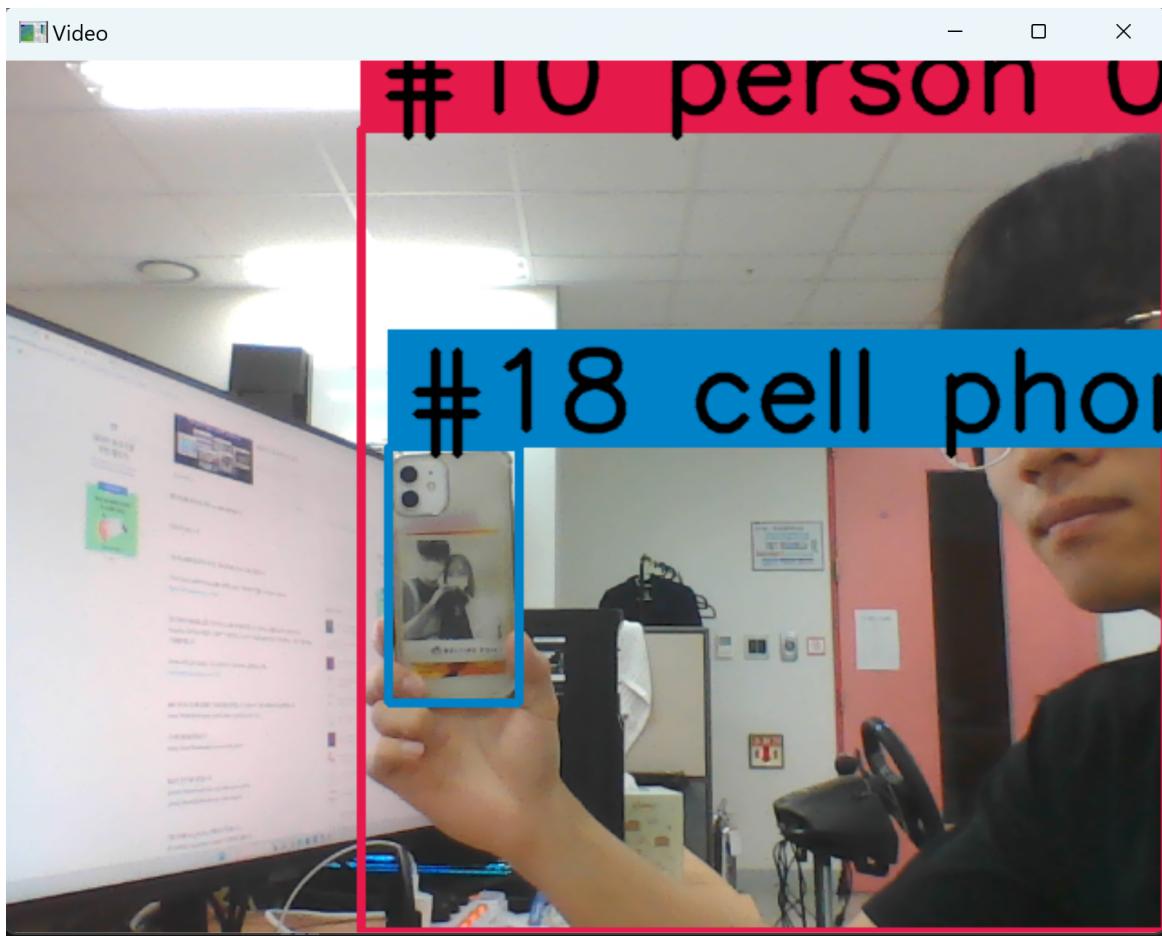
```

cap.set(cv2.CAP_PROP_FRAME_WIDTH, 640)
cap.set(cv2.CAP_PROP_FRAME_HEIGHT, 480)
box_annotator = BoxAnnotator(color=ColorPalette(), thickness=4, text_thickness=4, text_scale=2)
line_counter = LineCounter(start=LINE_START, end=LINE_END)
line_annotator = LineCounterAnnotator(thickness=4, text_thickness=4, text_scale=2)
while True:
    ret, frame = cap.read()
    results = model.predict(frame)
    detections = Detections(
        xyxy=results[0].boxes.xyxy.cpu().numpy(),
        confidence=results[0].boxes.conf.cpu().numpy(),
        class_id=results[0].boxes.cls.cpu().numpy().astype(int)
    )
    # filtering out detections with unwanted classes
    mask = np.array([class_id in CLASS_ID for class_id in detections.class_id], dtype=bool)
    detections.filter(mask=mask, inplace=True)
    tracks = byte_tracker.update(
        output_results=detections2boxes(detections=detections),
        img_info=frame.shape,
        img_size=frame.shape
    )
    tracker_id = match_detections_with_tracks(detections=detections, tracks=tracks)
    detections.tracker_id = np.array(tracker_id)
    mask = np.array([tracker_id is not None for tracker_id in detections.tracker_id], dtype=bool)
    detections.filter(mask=mask, inplace=True)
    # format custom labels
    labels = [
        f"#{tracker_id} {CLASS_NAMES_DICT[class_id]} {confidence:0.2f}"
        for _, confidence, class_id, tracker_id
        in detections
    ]
    # updating line counter
    line_counter.update(detections=detections)
    frame = box_annotator.annotate(frame=frame, detections=detections, labels=labels)
    line_annotator.annotate(frame=frame, line_counter=line_counter)
    cv2.imshow("Video", frame)
    cv2.waitKey(1)

cap.release()
cv2.destroyAllWindows()

```

- 코드를 실행하고 나면, 다음과 같이 이제는 object위에 object의 고유번호가 표시되는 것을 알 수 있습니다.



## ▼ 프로젝트 시작

- 프로젝트를 이제 한번 시작해봅시다!
- 우리가 해야할 일을 정리해 볼까요?
- 크게는 다음과 같습니다.
- 라즈베리파이의 USB 카메라에서 찍은 영상을 컴퓨터로 보내서 컴퓨터에서 영상을 object tracking을 통해 분석한다.
- 분석한 정보를 토대로 라즈베리파이에게 모터들의 속도정보를 보내준다.
- 세부적으로 코드를 뜯어볼까요? 하나하나 코드를 설명드리도록 하겠습니다.
- 먼저 라즈베리 파이에서 실행하는 코드부터 설명드리도록 하겠습니다.
- 맙은 라즈베리파이에서 실행하는 전체 코드입니다.

```
import cv2
import numpy as np
import socket
import threading
import time
from picar import front_wheels, back_wheels
```

```

from picar.SunFounder_PCA9685 import Servo
import picar
import struct

# 카메라 각도 건들이지 말자.

picar.setup()
rear_wheels_enable = True
front_wheels_enable = True

hmn = 12
hmx = 37
smn = 96
smx = 255
vmn = 186
vmx = 255

bw = back_wheels.Back_Wheels()
fw = front_wheels.Front_Wheels()
pan_servo = Servo.Servo(1) #왼쪽 오른쪽
tilt_servo = Servo.Servo(2)
picar.setup()

bw.backward()
fw.offset = 90 #50~120 , 원~오
pan_servo.offset = 90
tilt_servo.offset = 0

bw.speed = 0
fw.turn(0)
pan_servo.write(0)
tilt_servo.write(0)

# Define the host and port for the server (your laptop)
HOST = '172.20.10.3'
PORT = 5555

# Start the video capture from the USB camera
cap = cv2.VideoCapture(0)

# Set the resolution of the video
cap.set(cv2.CAP_PROP_FRAME_WIDTH, 640)
cap.set(cv2.CAP_PROP_FRAME_HEIGHT, 480)

# Create a socket object and connect to the server
client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
client_socket.connect((HOST, PORT))

# Start sending the video frames to the server
def send_cam_video():
    while True:
        # Capture a frame from the USB camera
        ret, frame = cap.read()

        # Convert the frame to a byte array
        data = cv2.imencode('.jpg', frame)[1].tostring()

        # Send the length of the frame data and then the frame data itself
        client_socket.sendall((str(len(data))).encode().ljust(16) + data)
        time.sleep(0.5)

```

```

th1 = threading.Thread(target=send_cam_video)
th1.start()

cnt = 0

while True:
    # actuator 정보 받아서 rc카 조종하기
    dta = client_socket.recv(1024)
    ac_list = list(dta.decode().split(','))
    act_list = list(map(int,ac_list))
    print(act_list[0])
    bw.speed = act_list[0]
    fw.turn(act_list[1])
    cnt += 1

# Close the client socket and release the video capture
client_socket.close()
cap.release()

```

- 하나하나 뜯어봅시다!!
- step1: 밑의 코드는 필요한 module들을 import하고 초기의 모터들의 상태를 설정해주는 코드입니다. 주석으로 자세하게 설명하였습니다.

```

# 필요한 모듈 가져오기
import cv2
import numpy as np
import socket
import threading
import time
from picar import front_wheels, back_wheels
from picar.SunFounder_PCA9685 import Servo
import picar
import struct

# RC카를 사용하겠다.
picar.setup()
rear_wheels_enable = True
front_wheels_enable = True

#원지는 잘 모르는 초기 rc카 변수 값.(rc car홈페이지에 공을 tracking하는 코드가 있는데, 이를 참고)
hmn = 12
hmx = 37
smn = 96
smx = 255
vmn = 186
vmx = 255

# 초기 변수 설정. 이제부터 bw는 뒷바퀴를 의미하고, fw는 앞바퀴를 의미함.
bw = back_wheels.Back_Wheels()
fw = front_wheels.Front_Wheels()
pan_servo = Servo.Servo(1) #왼쪽 오른쪽
tilt_servo = Servo.Servo(2)
picar.setup()

# 뒷바퀴 운동 방향 설정(앞/뒤 , 모터를 잘못 연결했는지 뒤로 가라고 해야 실제로 앞으로 감.)
bw.backward()
# 앞바퀴 각도 설정(50~120사이의 값. 90이 중간임. 하드웨어마다 달라서 직접 실험 해봐야 함.)

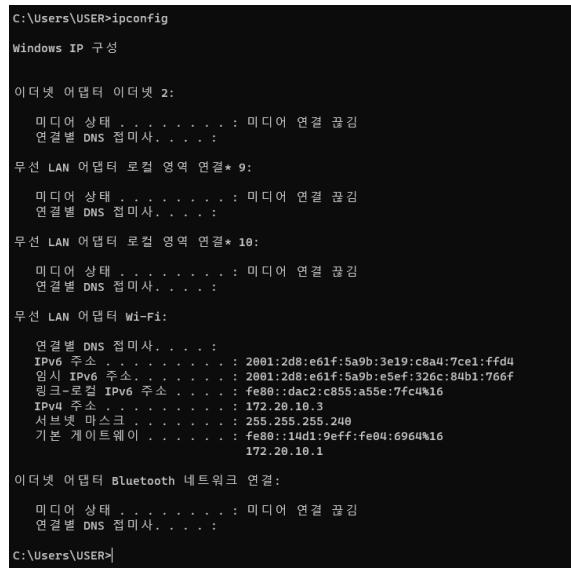
```

```

fw.offset = 90 #50~120 , 원~오
pan_servo.offset = 90
tilt_servo.offset = 0
#방향, 속도 등 초기화.
bw.speed = 0
fw.turn(0)
pan_servo.write(0)
tilt_servo.write(0)

```

- step 2: cv2 모듈 초기설정과 socket programming을 위한 초기 설정.
- HOST부분에 노트북의 wifi ip주소를 입력해야합니다. (노트북이랑 메세지를 주고받아야 하니까) 이는 노트북의 명령 프롬프트에서 ipconfig명령어를 통해서 얻을 수 있습니다.



```

#소켓 프로그래밍 초기설정
# Define the host and port for the server (your laptop)
HOST = '172.20.10.3' # 컴퓨터 wifi ip주소.
PORT = 5555

#cv2 모듈 초기설정
# Start the video capture from the USB camera
cap = cv2.VideoCapture(0)

#cv2 모듈 초기설정
# Set the resolution of the video
cap.set(cv2.CAP_PROP_FRAME_WIDTH, 640)
cap.set(cv2.CAP_PROP_FRAME_HEIGHT, 480)

#소켓 프로그래밍 초기 설정
# Create a socket object and connect to the server
client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
client_socket.connect((HOST, PORT))

```

- step3: 노트북에게 USB카메라로 찍은 영상 BYTE형태로 전송하기.

- 나중에 노트북으로부터 모터구동 정보를 받아 모터를 움직이는 것도 동시에 실행해야 하기 때문에, threading을 이용했습니다.

```
# Start sending the video frames to the server
def send_cam_video():
    while True:
        # Capture a frame from the USB camera
        ret, frame = cap.read()

        # Convert the frame to a byte array
        data = cv2.imencode('.jpg', frame)[1].tostring()

        # Send the length of the frame data and then the frame data itself
        client_socket.sendall((str(len(data))).encode().ljust(16) + data)
        time.sleep(0.5)
th1 = threading.Thread(target=send_cam_video)
th1.start()
```

- step 4: 노트북으로부터 모터 구동 정보 받아서 모터 움직이기.
- 노트북으로부터 list를 byte형태로 전송받습니다. list의 첫번째 원소에는 speed에 관한 정보가 들어 있고, 두번째 원소에는 앞바퀴 각도에 대한 정보가 들어있습니다.
- 전송받은 byte형태의 list를 python list로 decoding하고 이에 맞게 모터를 구동시키는 코드입니다.

```
while True:
    # actuator 정보 받아서 rc카 조종하기
    dta = client_socket.recv(1024)
    ac_list = list(dta.decode().split(','))
    act_list = list(map(int, ac_list))
    print(act_list[0])
    bw.speed = act_list[0]
    fw.turn(act_list[1])
    cnt += 1
```

- 다음으로는 노트북의 전체 코드입니다. 밑에서 세부적으로 설명하도록 하겠습니다.

```
from yolox.tracker.byte_tracker import BYTETracker, STrack
from onemetric.cv.utils.iou import box_iou_batch
from dataclasses import dataclass
from supervision.draw.color import ColorPalette
from supervision.geometry.dataclasses import Point
from supervision.tools.detections import Detections, BoxAnnotator
from supervision.tools.line_counter import LineCounter, LineCounterAnnotator
from typing import List
import numpy as np
from ultralytics import YOLO
import cv2
import socket
import threading
import queue
import time
```

```

import struct

@dataclass(frozen=True)
class BYTETrackerArgs:
    track_thresh: float = 0.25
    track_buffer: int = 30
    match_thresh: float = 0.8
    aspect_ratio_thresh: float = 3.0
    min_box_area: float = 1.0
    mot20: bool = False

    # converts Detections into format that can be consumed by match_detections_with_tracks function
    def detections2boxes(detections: Detections) -> np.ndarray:
        return np.hstack((
            detections.xyxy,
            detections.confidence[:, np.newaxis]
        ))

    # converts List[STrack] into format that can be consumed by match_detections_with_tracks function
    def tracks2boxes(tracks: List[STrack]) -> np.ndarray:
        return np.array([
            track.tlbr
            for track
            in tracks
        ], dtype=float)

    # matches our bounding boxes with predictions
    def match_detections_with_tracks(
        detections: Detections,
        tracks: List[STrack]
    ) -> Detections:
        if not np.any(detections.xyxy) or len(tracks) == 0:
            return np.empty((0,))

        tracks_boxes = tracks2boxes(tracks=tracks)
        iou = box_iou_batch(tracks_boxes, detections.xyxy)
        track2detection = np.argmax(iou, axis=1)

        tracker_ids = [None] * len(detections)

        for tracker_index, detection_index in enumerate(track2detection):
            if iou[tracker_index, detection_index] != 0:
                tracker_ids[detection_index] = tracks[tracker_index].track_id

        return tracker_ids

    model = YOLO("yolov8x.pt")
    model.to('cuda')
    CLASS_NAMES_DICT = model.model.names
    CLASS_ID = [0]
    byte_tracker = BYTETracker(BYTETrackerArgs())
    model.predict("test.png")

    # settings
    LINE_START = Point(50, 1500)
    LINE_END = Point(3840-50, 1500)

    # Set the resolution of the video

```

```

box_annotator = BoxAnnotator(color=ColorPalette(), thickness=4, text_thickness=4, text_scale=2)
line_counter = LineCounter(start=LINE_START, end=LINE_END)
line_annotator = LineCounterAnnotator(thickness=4, text_thickness=4, text_scale=2)

# Define the host and port for the server (your laptop)
HOST = '172.20.10.3'
PORT = 5555

# Create a socket object and bind it to the host and port
server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server_socket.bind((HOST, PORT))
server_socket.listen(1)

# Accept a client connection
print('Waiting for connection...')
client_socket, address = server_socket.accept()
print(f'Connected to {address[0]}')
gogo = 0
person_Detect = False
person_id = -10
def is_go():
    global person_Detect
    global gogo
    global person_id
    if(person_Detect):
        gogo = int(input("let's get it? 1 or 0 \n"))
        if (gogo == 1):
            person_id = int(input("which person do you want to track?"))
    time.sleep(2)
is_go()

th2 = threading.Thread(target = is_go)
th2.start()

while True:
    # Receive the length of the frame data and then the frame data itself
    data = b''
    while len(data) < 16:
        data += client_socket.recv(16 - len(data))
    frame_length = int(data)
    data = b''
    while len(data) < frame_length:
        data += client_socket.recv(frame_length - len(data))

    # Convert the received data back into an OpenCV frame
    frame = cv2.imdecode(np.frombuffer(data, dtype=np.uint8), cv2.IMREAD_COLOR)
    results = model.predict(frame)
    detections = Detections(
        xyxy=results[0].boxes.xyxy.cpu().numpy(),
        confidence=results[0].boxes.conf.cpu().numpy(),
        class_id=results[0].boxes.cls.cpu().numpy().astype(int)
    )
    # filtering out detections with unwanted classes
    mask = np.array([class_id in CLASS_ID for class_id in detections.class_id], dtype=bool)
    detections.filter(mask=mask, inplace=True)
    tracks = byte_tracker.update(
        output_results=detections2boxes(detections=detections),
        img_info=frame.shape,
        img_size=frame.shape
    )

```

```

tracker_id = match_detections_with_tracks(detections=detections, tracks=tracks)
detections.tracker_id = np.array(tracker_id)
mask = np.array([tracker_id is not None for tracker_id in detections.tracker_id], dtype=bool)
detections.filter(mask=mask, inplace=True)
# format custom labels
labels = [
    f"#{tracker_id} {CLASS_NAMES_DICT[class_id]} {confidence:.2f}"
    for _, confidence, class_id, tracker_id
    in detections
]
# updating line counter
line_counter.update(detections=detections)
frame = box_annotator.annotate(frame=frame, detections=detections, labels=labels)
line_annotator.annotate(frame=frame, line_counter=line_counter)
if 0 in detections.class_id:
    print("Detect person")
    person_Detect = True
    if(gogo==1):
        if(person_id in tracker_id):
            print("go")
            for i in range(len(tracker_id)):
                if(person_id == tracker_id[i]):
                    idx = i
                    bottom_right_x = detections.xyxy[idx][0]
                    bottom_right_y = detections.xyxy[idx][1]
                    top_left_x = detections.xyxy[idx][2]
                    top_left_y = detections.xyxy[idx][3]
                    centor_x = (top_left_x + bottom_right_x)/2
                    height = top_left_y-bottom_right_y
                    real_center_x = 320
                    real_height = 480
                    act1 = [0,0,0,0]
                    if(height > 480):
                        act1[0] = 0
                    else:
                        act1[0] = 20
                    if (centor_x-real_center_x < 100 and centor_x-real_center_x > -100):
                        act1[1] = 90
                    elif(centor_x > real_center_x):
                        act1[1] = 110
                    else:
                        act1[1] = 55
                    data1 = ','.join(str(elem) for elem in act1)
                    print(data1)
                    client_socket.send(data1.encode())
            else:
                print("stop")
                act2 = [0,90,0,0]
                data2 = ','.join(str(elem) for elem in act2)
                client_socket.send(data2.encode())
        else:
            print("stop")
            act2 = [0,90,0,0]
            data2 = ','.join(str(elem) for elem in act2)
            client_socket.send(data2.encode())
    else:
        act2 = [0,90,0,0]
        data2 = ','.join(str(elem) for elem in act2)
        client_socket.send(data2.encode())
        person_Detect = False

```

```

cv2.imshow("Video",frame)
cv2.waitKey(1)

```

- step1: 초기 설정
- 여러 모듈들을 사용하기 위한 초기설정을 진행합니다. 세부적인 코드는 저도 사실 잘 알지 못합니다.  
ㅎㅎ

```

from yolox.tracker.byte_tracker import BYTETracker, STrack
from onemetric.cv.utils.iou import box_iou_batch
from dataclasses import dataclass
from supervision.draw.color import ColorPalette
from supervision.geometry.dataclasses import Point
from supervision.tools.detections import Detections, BoxAnnotator
from supervision.tools.line_counter import LineCounter, LineCounterAnnotator
from typing import List
import numpy as np
from ultralytics import YOLO
import cv2
import socket
import threading
import queue
import time
import struct

@dataclass(frozen=True)
class BYTETrackerArgs:
    track_thresh: float = 0.25
    track_buffer: int = 30
    match_thresh: float = 0.8
    aspect_ratio_thresh: float = 3.0
    min_box_area: float = 1.0
    mot20: bool = False

    # converts Detections into format that can be consumed by match_detections_with_tracks function
    def detections2boxes(detections: Detections) -> np.ndarray:
        return np.hstack((
            detections.xyxy,
            detections.confidence[:, np.newaxis]
        ))

    # converts List[STrack] into format that can be consumed by match_detections_with_tracks function
    def tracks2boxes(tracks: List[STrack]) -> np.ndarray:
        return np.array([
            track.tlbr
            for track
            in tracks
        ], dtype=float)

    # matches our bounding boxes with predictions
    def match_detections_with_tracks(
        detections: Detections,
        tracks: List[STrack]
    ) -> Detections:
        if not np.any(detections.xyxy) or len(tracks) == 0:
            return np.empty((0,))

```

```

tracks_boxes = tracks2boxes(tracks=tracks)
iou = box_iou_batch(tracks_boxes, detections.xyxy)
track2detection = np.argmax(iou, axis=1)

tracker_ids = [None] * len(detections)

for tracker_index, detection_index in enumerate(track2detection):
    if iou[tracker_index, detection_index] != 0:
        tracker_ids[detection_index] = tracks[tracker_index].track_id

return tracker_ids

```

- step2: 초기설정(2)
- YOLO 알고리즘을 사용하기 위한 초기설정을 진행합니다.
- CLASS\_ID = [0]는 사람만을 DETECT하겠다는 것을 의미합니다.
- model.predict("test.png") 이 코드는 본격적으로 라즈베리파이로부터 동영상을 받아서 DETECTION을 실행하기 전에, 미리 모델을 캐쉬에 올려놓아서 후에 DETECTION의 Delay를 조금이라도 줄여주는 역할을 수행합니다. (이를 하지 않는다면, 코드 실행 초기에 delay가 발생하는 것을 확인할 수 있습니다.)
- model.to('cuda') 코드는 GPU를 사용하겠다는 코드입니다. GPU가 없으신 분들은 이 코드를 제거해주시기 바랍니다.

```

model = YOLO("yolov8x.pt")
model.to('cuda')
CLASS_NAMES_DICT = model.model.names
CLASS_ID = [0]
byte_tracker = BYTETracker(BYTETrackerArgs())
model.predict("test.png")

# settings
LINE_START = Point(50, 1500)
LINE_END = Point(3840-50, 1500)

# Set the resolution of the video

box_annotator = BoxAnnotator(color=ColorPalette(), thickness=4, text_thickness=4, text_scale=2)
line_counter = LineCounter(start=LINE_START, end=LINE_END)
line_annotator = LineCounterAnnotator(thickness=4, text_thickness=4, text_scale=2)

```

- Step 3: 소켓 프로그래밍 설정과 interface 만들기
- 이 코드에서는 소켓 프로그래밍을 위한 기본설정을 실시합니다.
- is\_go 함수를 보면, detect한 사람 중에 id가 몇번인 사람을 따라갈 것인지에 대해 질문하는 코드입니다. 만약 뒤에 소개할 tracking 코드에서 사람을 detection하면 사람을 따라갈 것인지 물어보고 (global variable 이용), 따라간다고 하면, 몇번 사람을 따라갈 것인지를 물어보고 global variable에 저장해 두어 뒤에 tracking code에서 이용할 수 있도록 합니다.

```

# Define the host and port for the server (your laptop)
HOST = '172.20.10.3'
PORT = 5555

# Create a socket object and bind it to the host and port
server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server_socket.bind((HOST, PORT))
server_socket.listen(1)

# Accept a client connection
print('Waiting for connection...')
client_socket, address = server_socket.accept()
print(f'Connected to {address[0]}')
gogo = 0
person_Detect = False
person_id = -10
def is_go():
    global person_Detect
    global gogo
    global person_id
    if(person_Detect):
        gogo = int(input("let's get it? 1 or 0 \n"))
        if (gogo == 1):
            person_id = int(input("which person do you want to track?"))
    time.sleep(2)
    is_go()

th2 = threading.Thread(target = is_go)
th2.start()

```

- STEP 4: tracking code
- 아래는 Tracking을 수행하는 code 입니다.
- 먼저 라즈베리파이로부터 사진을 전송받습니다.
- 받은 사진을 통해서 tracking을 수행합니다.
- 만약 사람이 발견되면, 앞의 threading code가 실행되며 몇번 사람을 따라갈 것인지 물게되고, global variable에 따라갈 사람의 id가 저장됩니다.
- global variable에 저장된 따라갈 사람의 id와 일치하는 사람이 tracking되면, tracking의 결과물인 사진에서 그 사람의 중심좌표를 추출합니다. 그 사람의 중심좌표와 사진의 중심좌표 사이의 차이를 토대로 모터구동 정보를 구합니다.
- 구한 모터구동 정보를 라즈베리파이에게 전송합니다.

```

while True:
    # Receive the length of the frame data and then the frame data itself
    data = b''
    while len(data) < 16:
        data += client_socket.recv(16 - len(data))
    frame_length = int(data)
    data = b''
    while len(data) < frame_length:

```

```

    data += client_socket.recv(frame_length - len(data))

    # Convert the received data back into an OpenCV frame
    frame = cv2.imdecode(np.frombuffer(data, dtype=np.uint8), cv2.IMREAD_COLOR)
    results = model.predict(frame)
    detections = Detections(
        xyxy=results[0].boxes.xyxy.cpu().numpy(),
        confidence=results[0].boxes.conf.cpu().numpy(),
        class_id=results[0].boxes.cls.cpu().numpy().astype(int)
    )
    # filtering out detections with unwanted classes
    mask = np.array([class_id in CLASS_ID for class_id in detections.class_id], dtype=bool)
    detections.filter(mask=mask, inplace=True)
    tracks = byte_tracker.update(
        output_results=detections2boxes(detections=detections),
        img_info=frame.shape,
        img_size=frame.shape
    )
    tracker_id = match_detections_with_tracks(detections=detections, tracks=tracks)
    detections.tracker_id = np.array(tracker_id)
    mask = np.array([tracker_id is not None for tracker_id in detections.tracker_id], dtype=bool)
    detections.filter(mask=mask, inplace=True)
    # format custom labels
    labels = [
        f"#{tracker_id} {CLASS_NAMES_DICT[class_id]} {confidence:0.2f}"
        for _, confidence, class_id, tracker_id
        in detections
    ]
    # updating line counter
    line_counter.update(detections=detections)
    frame = box_annotator.annotate(frame=frame, detections=detections, labels=labels)
    line_annotator.annotate(frame=frame, line_counter=line_counter)
    if 0 in detections.class_id:
        print("Detect person")
        person_Detect = True
        if(gogo==1):
            if(person_id in tracker_id):
                print("go")
                for i in range(len(tracker_id)):
                    if(person_id == tracker_id[i]):
                        idx = i
                bottom_right_x = detections.xyxy[idx][0]
                bottom_right_y = detections.xyxy[idx][1]
                top_left_x = detections.xyxy[idx][2]
                top_left_y = detections.xyxy[idx][3]
                centor_x = (top_left_x + bottom_right_x)/2
                height = top_left_y-bottom_right_y
                real_center_x = 320
                real_height = 480
                act1 = [0,0,0,0]
                if(height > 480):
                    act1[0] = 0
                else:
                    act1[0] = 20
                if (centor_x-real_center_x < 100 and centor_x-real_center_x > -100):
                    act1[1] = 90
                elif(centor_x > real_center_x):
                    act1[1] = 110
                else:
                    act1[1] = 55
                data1 = ','.join(str(elem) for elem in act1)

```

```

        print(data1)
        client_socket.send(data1.encode())
    else:
        print("stop")
        act2 = [0,90,0,0]
        data2 = ','.join(str(elem) for elem in act2)
        client_socket.send(data2.encode())
    else:
        print("stop")
        act2 = [0,90,0,0]
        data2 = ','.join(str(elem) for elem in act2)
        client_socket.send(data2.encode())
    else:
        act2 = [0,90,0,0]
        data2 = ','.join(str(elem) for elem in act2)
        client_socket.send(data2.encode())
        person_Detect = False
        cv2.imshow("Video",frame)
        cv2.waitKey(1)

```

### ▼ 최종 실행 방법

- RC카의 전원을 킨다.
- RC카의 전원을 키게되면 자동으로 라즈베리파이에 전류가 공급된다.(하드웨어 특성)
- 라즈베리파이의 전원을 킨다.
- 라즈베리파이에 모니터와 키보드를 연결하여 WIFI IP ADDRESS를 얻는다.
- ip address를 따로 기록해두고, 모니터와 키보드를 분리한다.
- 노트북의 전원을 킨다.
- 노트북에서 putty를 실행하여 라즈베리파이의 command shell을 노트북에서 연다.
- 노트북에서 노트북 python code를 실행한다.
- 노트북에서 라즈베리파이의 command shell에서 라즈베리파이 python code를 실행한다.
- 완성~ 자신을 잘 따라오는 rc car를 볼 수 있다.

person tracking rc car using PICAR-V and YOLO algorithm  
Using YOLO Algorithm and PICAR-V, i can make person tracking rc car.

 [https://www.youtube.com/watch?v=RU-VkQ\\_2yKE](https://www.youtube.com/watch?v=RU-VkQ_2yKE)



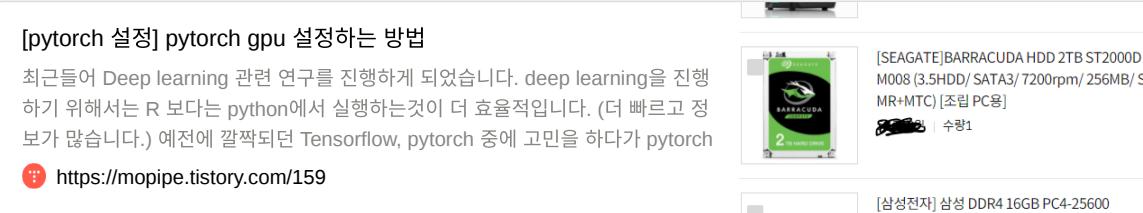


### ▼ GPU 설정하기(선택)

- 프로젝트를 하면서 YOLO 알고리즘의 상당한 computation delay가 걸리는 것을 확인할 수 있을 것입니다. 즉 YOLO함수에서의 계산시간이 너무 많이 걸려서 10초전 영상을 바탕으로 RC카가 이동하는 등의 현상을 관찰할 수 있을 것입니다.
- 이 문제는 YOLO Algorithm을 CPU가 아닌 GPU를 이용하여 연산한다면 해결할 수 있습니다.
- 제가 참고한 설정 사이트입니다.
- 이대로만 따라하면 됩니다!

[pytorch 설정] pytorch gpu 설정하는 방법

최근들어 Deep learning 관련 연구를 진행하게 되었습니다. deep learning을 진행하기 위해서는 R 보다는 python에서 실행하는것이 더 효율적입니다. (더 빠르고 정 보가 많습니다.) 예전에 깔짝되던 Tensorflow, pytorch 중에 고민을 하다가 pytorch



- 유튜브와 여러 사이트들을 참고하여 GPU를 사용해보세요!
- 이 역시 torch, torchvision, numpy 등등과의 모듈 버전 호환에 꼭 주의하세요!
- 버전 호환 정보는 참고자료의 맨 밑에 나와있습니다.
- 이것도 설정하시는데 상당히 고생하실 것입니다. 저도 2일 꼬박 걸렸습니다!! 파이팅입니다!!

### ▼ 참고자료

- python으로 object detection 해보기(이미 있는 사진 or 영상)

Complete YOLO v8 Custom Object Detection Tutorial | Windows & Linux

A complete YOLO v8 custom object detection tutorial with two-class custom dataset. I cover how to annotate custom dataset in YOLO format, setting up environment for YOLO v8, and train custom object detection model. Although I

 [https://www.youtube.com/watch?v=gRAyOPjQ9\\_s](https://www.youtube.com/watch?v=gRAyOPjQ9_s)



- 여러 사진 dataset들을 구할 수 있는 사이트.

#### tank | Roboflow Universe Search

Open source computer vision datasets and pre-trained models. Showing projects matching "tank" by subject, page 1.

🔗 <https://universe.roboflow.com/search?q=tank>



- 실시간으로 노트북 웹캠이나 USB Camera 사용해서 object detection 해보기.

#### Real-Time Object Detection with YOLOv8 and Webcam: Step-by-step Tutorial

In this video 📺 we are going to Deploy a YOLOv8 Object Detection Model in Python. We will see how to deploy a trained YOLOv8 model and run inference on a webcam. This can actually run in real-time. You can also check out my YOLOv7 Course

▶ <https://www.youtube.com/watch?v=IHbJcOex6dk>



- 혹시 gpu를 컴퓨터에서 사용할 수 있다면, gpu 사용할 때 version 호환정보.

#### PyTorch

An open source machine learning framework that accelerates the path from research prototyping to production deployment.

🔗 <https://pytorch.org/get-started/previous-versions/>



- 제가 발생했던 에러.(github 쓰는게 정답)

#### Is there a way to install cython-bbox for Windows?

I have Visual Studio installed and am trying to run a Python3 program which depends on the cython\_bbox package. I tried to install it with pip install cython-bbox on a Windows Anaconda3 environment...

⚠ <https://stackoverflow.com/questions/60349980/is-there-a-way-to-install-cython-bbox-for-windows>



- 버전 추천(with numpy 1.22.4)

```
Ultralytics YOLOv8.0.10 🚀 Python-3.8.10 torch-1.13.1+cu116 CUDA:0
```