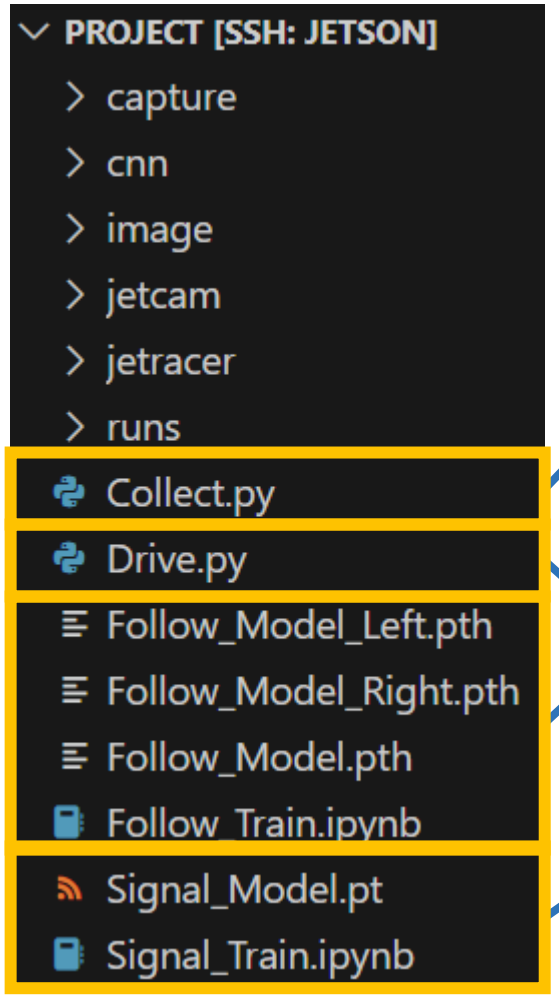
Embedded Project1
Line-Tracing and Sign-Detecting

Team 9
2019043836 김인호
2024152152 한정운

Content

Content



1. Image Collection
2. Line-Tracing-Model
3. Sign-Detecting-Model
4. Autonomous Driving

Image Collection

Image Collection

- CSI카메라를 이용(CV2)
- 이미지를 RGB로 변환하여 저장(imwrite)
- 직진, 회전, 교차로 세 가지 종류의 도로를 연결하여 중앙 차선을 따라가도록 함

```python
# 카메라에 대한 레코딩을 계속 관장하는 스레드 함수
def view():
    frame = camera.read()
    frame_index = 0

    while True:
        if video:
            print("Record")
            frame = camera.read()
            if frame is not None:
                image_filename = os.path.join("/home/ircv8/HYU-2024-Embedded/jetracer/image", f"image_{frame_index:09d}.jpg")
                cv2.imwrite(image_filename, frame[:,:,::-1])  # OpenCV는 BGR 형식을 사용하므로 RGB로 변환하여 저장
                print(f"Image saved as {image_filename}")
                frame_index += 1
                sleep(1)
```

Image Collection

- 조이스틱의 SELECT, START 버튼에 각각 시작과 종료 동작을 할당
- 보다 다양한 이미지 수집을 위해 주행, 정지를 반복하며 일정 시간마다 이미지를 레코딩
- 주행, 정지 반복을 위한 적절한 throttle값은 시행착오를 통해 선정 (0.33throttle)

```
# 3. 스레드 하나 열어서, 버튼에 대해 눌리면 카메라 영상 저장하는거
video = False
camera = CSICamera(capture_width=1280, capture_height=720, downsample=2, capture_fps=30)
thread = threading.Thread(target=view, args=())
thread.start()

# 3. 젯레이서 조이스틱으로 조작하기
while running:
    pygame.event.pump()

    throttle = -joystick.get_axis(1)
    throttle = max(throttle_range[0], min(throttle_range[1], throttle))
    steering = joystick.get_axis(2)

    #print(throttle, steering)
    car.throttle = throttle
    car.steering = steering
    if joystick.get_button(10): # select button
            video = True
    if joystick.get_button(11): # start button
            video = False
    if joystick.get_button(12): # home button
        running = False
        camera.release()
```

```
# JetRacer
car.steering = np.tanh(0.1*(x-315))
car.throttle = 0
sleep(0.1)
car.throttle = 0.1
car.throttle = 0.2
car.throttle = 0.33
sleep(0.2)
car.throttle = 0
sleep(0.1)
```

Line-Tracing-Model

Line-Tracing-Model

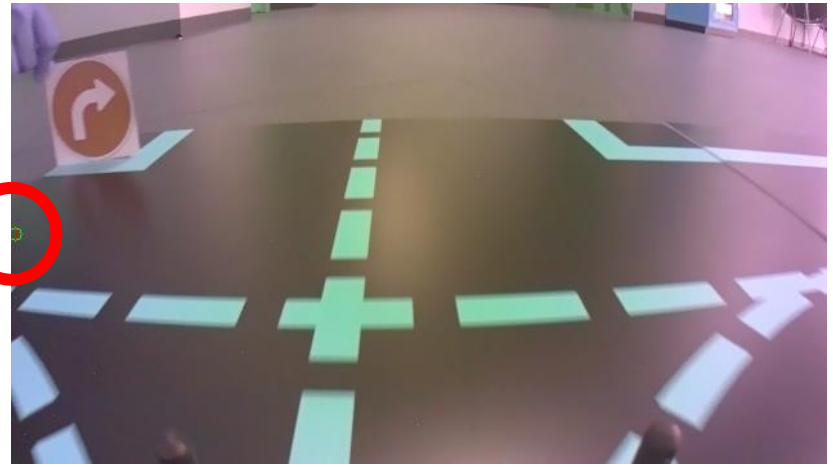
- 250장의 레코딩한 이미지를 라벨링

- 딜레이를 고려하여 이미지의 보다 먼 곳을 포착하도록 y=180으로 설정

```
import cv2
import os

from collections import OrderedDict
from ipywidgets import IntSlider, Label, Button, HBox
from ipycanvas import MultiCanvas, hold_canvas

thickness = 3
y_ratio = 0.5     # percentile of y-position from the top

# Input images
img_filename_fmt = 'image/frame_{:09d}.jpg'
ann_filename = 'image/annotation.txt'
ann_dict = OrderedDict()

num_frames = len(os.listdir(os.path.dirname(img_filename_fmt)))-1

cur_index = 0
height, width = cv2.imread(img_filename_fmt.format(cur_index)).shape[:2]
y_value = int(height * y_ratio)
```

Line-Tracing-Model

- Alexnet을 이용하여 라벨링 학습
- Loss가 Epoch 150을 넘으면 대부분 0이 되는 것을 확인 (Epoch = 150 / Learning Rate = 2e^-3)
- 교차로에 들어갈 때 경로의 좌, 우측을 라벨링함
- 교차로를 나갈 때 경로의 중심을 라벨링함
- 중심선이 여러 개 있는 상황(교차로)에서 능동적인 방향 전환을 위해 세가지 Line-Tracing Model을 제작



ex) Follow_Model_Left

```python
import torch
import torchvision

def get_model():
    model = torchvision.models.alexnet(num_classes=2, dropout=0.0)
    return model

device = torch.device('cuda')
model = get_model()
model = model.to(device)
```

```python
import torch
from cnn.center_dataset import CenterDataset

batch_size = 4

dataset = CenterDataset('image', random_hflip=False)
train_loader = torch.utils.data.DataLoader(
    dataset,
    num_workers=0,
    batch_size=batch_size,
    shuffle=True,
)
```

Epochs 150 lr 0.0020 Train

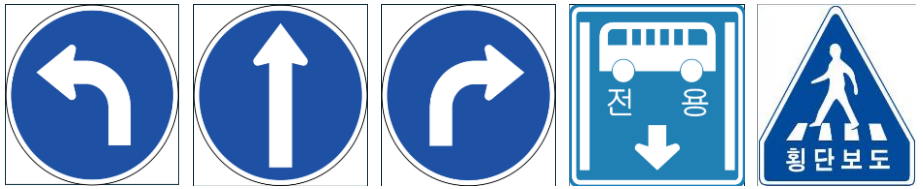
Progress

```
[0030 / 0063] loss: 0.0089 | labels: (288.75, 225.00), outpus: (273.57, 224.82)
[0040 / 0063] loss: 0.0019 | labels: (750.00, 225.00), outpus: (758.81, 224.97)
[0050 / 0063] loss: 0.0290 | labels: (362.50, 225.00), outpus: (361.45, 224.63)
[0060 / 0063] loss: 0.0051 | labels: (432.50, 225.00), outpus: (452.03, 225.19)
<<<<< Epoch 18 >>>>>
[0000 / 0063] loss: 0.0046 | labels: (377.50, 225.00), outpus: (391.40, 225.11)
[0010 / 0063] loss: 0.0149 | labels: (353.75, 225.00), outpus: (361.93, 224.42)
[0020 / 0063] loss: 0.1033 | labels: (6.25, 225.00), outpus: (67.54, 225.16)
[0030 / 0063] loss: 0.0037 | labels: (546.25, 225.00), outpus: (561.50, 225.90)
[0040 / 0063] loss: 0.0109 | labels: (11.25, 225.00), outpus: (-24.66, 224.71)
[0050 / 0063] loss: 0.0181 | labels: (266.25, 225.00), outpus: (272.95, 225.55)
[0060 / 0063] loss: 0.0017 | labels: (326.25, 225.00), outpus: (327.40, 224.92)
<<<<< Epoch 19 >>>>>
[0000 / 0063] loss: 0.0502 | labels: (6.25, 225.00), outpus: (1.13, 225.78)
[0010 / 0063] loss: 0.0056 | labels: (316.25, 225.00), outpus: (321.65, 225.23)
```

Follow_Model_Left.pth U
Follow_Model_Right.pth U
Follow_Model.pth U

Sign-Detecting-Model

Sign-Detecting-Model



- Roboflow를 이용해 이미지 라벨링 진행
- Straight, Right, Left, Bus stop, Crosswalk 5case

Sign-Detecting-Model

- **이미지 학습은 YOLO 이용**
- **80 epoch 진행**

```
from ultralytics import YOLO

model = YOLO('yolov8n.pt')
```

```
[ ] # model type, len 확인
    print(type(model.names),len(model.names))

    print(model.names)
```

```
model.train(data='/content/drive/MyDrive/hyu/traffic_sign_Data/traffic_sign_Data.yaml',epochs=80, patience=30, batch=32, imgsz=416,fliplr=0)
```

```
Epoch   GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
 1/80     1.98G        1.7      4.731       1.51         10        416: 100%|██████████| 7/7 [00:13<00:00,  1.88s/it]
         Class     Images  Instances      Box(P          R       mAP50  mAP50-95): 100%|██████████| 1/1 [00:01<00:00,  1.66s/it]                 all         25         25    0.00044     0.0667    0.00211    0.00134

Epoch   GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
 2/80     1.97G      1.646      4.333      1.426         17        416: 100%|██████████| 7/7 [00:01<00:00,  4.19it/s]
         Class     Images  Instances      Box(P          R       mAP50  mAP50-95): 100%|██████████| 1/1 [00:00<00:00,  2.54it/s]                 all         25         25     0.0005     0.0667     0.0177     0.0122

Epoch   GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
 3/80     1.97G      1.762      3.669      1.485         11        416: 100%|██████████| 7/7 [00:01<00:00,  4.26it/s]
         Class     Images  Instances      Box(P          R       mAP50  mAP50-95): 100%|██████████| 1/1 [00:00<00:00,  3.87it/s]                 all         25         25    0.00101      0.233     0.0233     0.0166
```

⋮

```
Epoch   GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
80/80     1.98G      1.008     0.7819      1.058          5        416: 100%|██████████| 7/7 [00:01<00:00,  4.10it/s]
         Class     Images  Instances      Box(P          R       mAP50  mAP50-95): 100%|██████████| 1/1 [00:00<00:00,  2.38it/s]                 all         25         25      0.962      0.931      0.979      0.568
```

Autonomous Driving

Five Mode

Infinite Loop

5 Mode base on Sign-Detecting-Result

Mode1
Straight

Mode2
Left

Mode3
Right

Mode4
Bus

Mode5
Crosswalk

Straight Mode

1. 카메라를 통해 실시간이미지 가져오기

```
# Camera
frame = camera.read()
if frame is not None:
    capture_filename = os.path.join("/home/ircv8/HYU-2024-Embedded/Project/capture", f"capture.jpg")
    cv2.imwrite(capture_filename, frame[:,:,::-1])
```

2. 이미지파일을 기본직진모델에 넣기 → 중앙선 X좌표 파악

```
# Model
capture_filename_fmt = 'capture/capture.jpg'
capture_ori = PIL.Image.open(capture_filename_fmt)
width = capture_ori.width
height = capture_ori.height
with torch.no_grad():
    capture = preprocess(capture_ori)
    output = model(capture).detach().cpu().numpy()
x, y = output[0]
x = (x / 2 + 0.5) * width
y = (y / 2 + 0.5) * height
```

Straight Mode

3. 이미지파일을 사인감지모델에 넣기 → 결과에 따른 모드 변경

```
result = model_signal.predict(source='/home/ircv8/HYU-2024-Embedded/Project/capture', save=True)
text = str(result[0].__dict__['boxes'].cls)
if(Lock==0):
    if(text[8]==']'):
        Mode = 1
    elif(text[8]=='0'):
        Mode = 4
        Lock = 11
    elif(text[8]=='1'):
        Mode = 5
        Lock = 11
    elif(text[8]=='2'):
        Mode = 2
        Lock = 11
    elif(text[8]=='3'):
        Mode = 3
        Lock = 11
    elif(text[8]=='4'):
        Mode = 1
        Lock = 11
Lock = max(0, Lock-1)
```

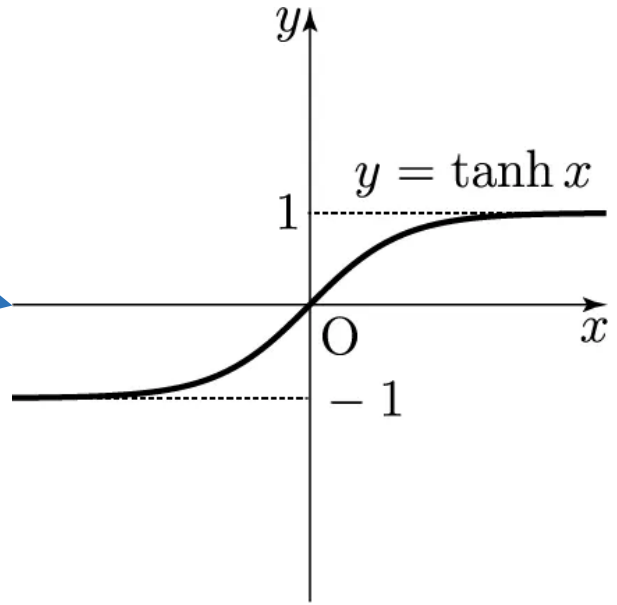
No Detection : 직진모드유지

각클래스에 맞는 모드로 변경
(다음반복때 해당모드로 적용)

Straight Mode

4. 젯레이서의 Steering 조정

```
# JetRacer
car.steering = np.tanh(0.1*(x-315))
car.throttle = 0
sleep(0.1)
car.throttle = 0.1
car.throttle = 0.2
car.throttle = 0.33
sleep(0.2)
car.throttle = 0
sleep(0.1)
```



Why Tanh?

- 젯레이서의 조향범위 [-1,1] 범위를 가짐
- 0과 가까운 범위에서 비례그래프
- X-315(중앙선 기준 변위)값을 통한 비례제어

Straight Mode

5. 젯레이서의 Throttle 조정

```
# JetRacer
car.steering = np.tanh(0.1*(x-315))
car.throttle = 0
sleep(0.1)
car.throttle = 0.1
car.throttle = 0.2
car.throttle = 0.33
sleep(0.2)
car.throttle = 0
sleep(0.1)
```

1. 정지상태에서 Steering값을 바꿈
2. Trottle을 올리고, Sleep을 통해 0.2초 전진
3. 다시 정지상태로 바꿈

이미지에 따른 Steering 변화를 확실히 적용한 뒤 전진하기 때문에 더욱 안정적인 트래킹이 가능

Left Mode

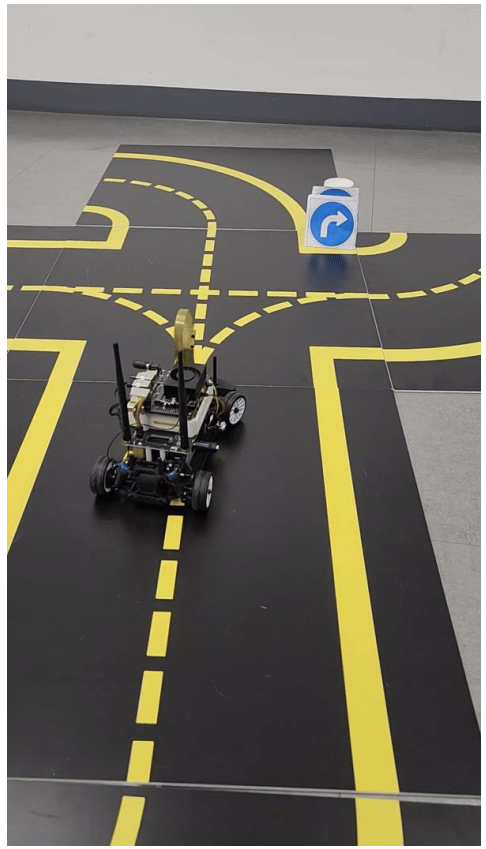
Straight Mode에서 교차로좌회전모델로만 바꿈

```
# Model
capture_filename_fmt = 'capture/capture.jpg'
capture_ori = PIL.Image.open(capture_filename_fmt)
width = capture_ori.width
height = capture_ori.height
with torch.no_grad():
    capture = preprocess(capture_ori)
    output = model_left(capture).detach().cpu().numpy()
x, y = output[0]
x = (x / 2 + 0.5) * width
y = (y / 2 + 0.5) * height
```

Right Mode

Straight Mode에서 교차로우회전모델로만 바꿈

```
# Model
capture_filename_fmt = 'capture/capture.jpg'
capture_ori = PIL.Image.open(capture_filename_fmt)
width = capture_ori.width
height = capture_ori.height
with torch.no_grad():
    capture = preprocess(capture_ori)
    output = model_right(capture).detach().cpu().numpy()
x, y = output[0]
x = (x / 2 + 0.5) * width
y = (y / 2 + 0.5) * height
```

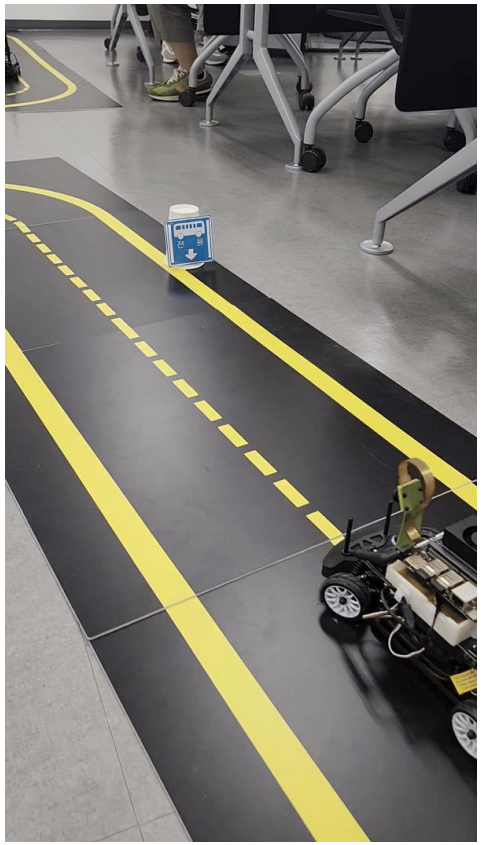


Bus Mode

Straight Mode에서 정지상태 Sleep시간을 약간늘려 서행

```
# JetRacer
car.steering = np.tanh(0.1*(x-315))
car.throttle = 0
sleep(0.2)
car.throttle = 0.1
car.throttle = 0.2
car.throttle = 0.33
sleep(0.2)
car.throttle = 0
sleep(0.2)
```

0.1s → 0.2s



Crosswalk Mode

Straight Mode에서 정지상태 Sleep시간을 매우길게하여 일시정지

```
# JetRacer
car.steering = np.tanh(0.1*(x-315))
car.throttle = 0
sleep(5)
car.throttle = 0.1
car.throttle = 0.2
car.throttle = 0.33
sleep(0.4)
car.throttle = 0
sleep(0.1)
```

0.1s → 5s

Lock

Lock 전역변수를 이용해, 일정반복동안 모드가 바뀌지 않도록 유지

- Left/Right에서 카메라 시야에 표지판이 벗어난 경우에도, 일정시간동안은 모드를 유지해야함
- Crosswalk에서 정지 후 다시 출발시, 직진모드로 Lock을 걸어서 또다시 멈추는 것을 방지

```
if(Lock==0):
    if(text[8]==']'):
        Mode = 1
    elif(text[8]=='0'):
        Mode = 4
        Lock = 11
    elif(text[8]=='1'):
        Mode = 5
        Lock = 11
    elif(text[8]=='2'):
        Mode = 2
        Lock = 11
    elif(text[8]=='3'):
        Mode = 3
        Lock = 11
    elif(text[8]=='4'):
        Mode = 1
        Lock = 11
Lock = max(0, Lock-1)
```

1. 모드변경시 락걸기
2. 반복당 락을 1씩 감소시킴
3. 0까지 내려가지 않으면 모드변경 불가

Thank You