

# 차량지능기초 과제3

딥러닝 차선 검출 알고리즘(LaneNet)을 이해하고 Inference Code 작성

20191620 소프트웨어학부 심혜린

Colab: <https://colab.research.google.com/drive/1-IWh97LsYHB61mc4p3xxP2UCTy8VQMe7?usp=sharing>

GitHub: <https://github.com/ShimHyerin/2021-VehicleIntelligence/tree/main/HW3>

📎 LaneNet\_Report3 <https://colab.research.google.com/drive/1-IWh97LsYHB61mc4p3xxP2UCTy8VQMe7>

## 전체 Code

```
from google.colab import drive  
drive.mount('/content/drive')
```

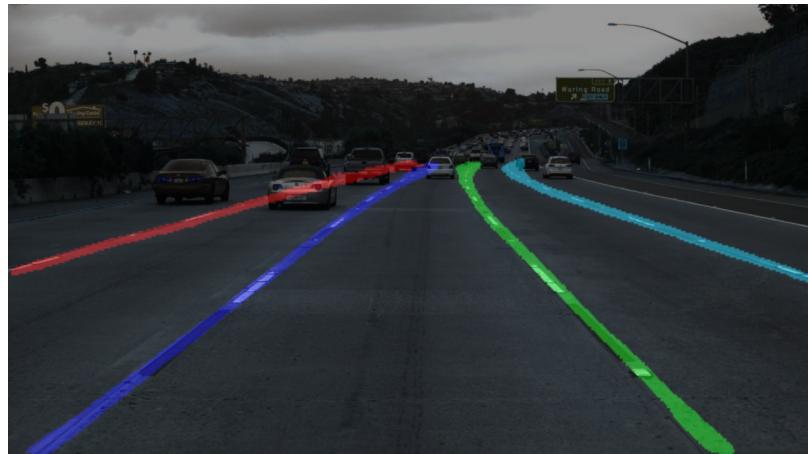
▼ Mounted at /content/drive

```
!git clone https://github.com/Lhyejin/LaneNet-PyTorch.git
```

```
/content/LaneNet-PyTorch  
dataset LICENSE requirements.txt Train.ipynb  
img Notebook-experiment test.py TUSIMPLE  
Lanenet README.md Train_aug.ipynb utils  
Requirement already satisfied: torch==1.2.0 in /usr/local/lib/python3.7/dist-packages (from -r requirements.txt (line 1)) (1.2.0)  
Requirement already satisfied: torchvision==0.4.0 in /usr/local/lib/python3.7/dist-packages (from -r requirements.txt (line 2)) (0.4.0)  
Requirement already satisfied: scikit-learn==0.22.1 in /usr/local/lib/python3.7/dist-packages (from -r requirements.txt (line 3)) (0.22.1)  
Requirement already satisfied: numpy==1.18.2 in /usr/local/lib/python3.7/dist-packages (from -r requirements.txt (line 4)) (1.18.2)  
Requirement already satisfied: pillow>=4.1.1 in /usr/local/lib/python3.7/dist-packages (from -r requirements.txt (line 2)) (7.1.2)  
Requirement already satisfied: six in /usr/local/lib/python3.7/dist-packages (from -r requirements.txt (line 2)) (1.15.0)  
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.7/dist-packages (from -r requirements.txt (line 3)) (1.0.1)  
Requirement already satisfied: scipy>=0.17.0 in /usr/local/lib/python3.7/dist-packages (from -r requirements.txt (line 3)) (1.4.1)
```

```
%cd LaneNet-PyTorch//  
!ls  
!pip install -r requirements.txt
```

```
import os.path as ops  
import numpy as np  
import torch  
import cv2  
import time  
import os  
import matplotlib.pyplot as plt  
import sys  
from tqdm import tqdm  
import imageio  
from dataset.dataset_utils import TUSIMPLE  
from Lanenet.model2 import Lanenet  
from utils.evaluation import gray_to_rgb_emb, process_instance_embedding  
from google.colab.patches import cv2_imshow  
  
device = 'cuda' if torch.cuda.is_available() else 'cpu'  
  
# Load the Model  
model_path = './TUSIMPLE/Lanenet_output/lanenet_epoch_39_batch_8.model'  
LaneNet_model = Lanenet(2, 4)  
LaneNet_model.load_state_dict(torch.load(model_path, map_location=torch.device(device)))  
LaneNet_model.to(device)  
  
def inference(gt_img_org):  
    # BGR 순서  
    org_shape = gt_img_org.shape  
    gt_image = cv2.resize(gt_img_org, dsize=(512, 256), interpolation=cv2.INTER_LINEAR)  
    gt_image = gt_image / 127.5 - 1.0  
    gt_image = torch.tensor(gt_image, dtype=torch.float)  
    gt_image = np.transpose(gt_image, (2, 0, 1))  
    gt_image = gt_image.to(device)  
    # lane segmentation  
    binary_final_logits, instance_embedding = LaneNet_model(gt_image.unsqueeze(0))  
    binary_final_logits, instance_embedding = binary_final_logits.to('cpu'), instance_embedding.to('cpu')  
    binary_img = torch.argmax(binary_final_logits, dim=1).squeeze().numpy()  
    binary_img[0:65,:] = 0  
  
    # lane clustering & segemented frame embedding  
    rbg_emb, cluster_result = process_instance_embedding(instance_embedding, binary_img,  
                                                          distance=1.5, lane_num=4)  
  
    rbg_emb = cv2.resize(rbg_emb, dsize=(org_shape[1], org_shape[0]), interpolation=cv2.INTER_LINEAR)  
    a = 0.6  
    frame = a * gt_img_org[..., ::-1] / 255 + rbg_emb * (1 - a)  
    frame = np.rint(frame * 255)  
    frame = frame.astype(np.uint8)  
  
    return frame  
  
image = cv2.imread('TUSIMPLE/test_clips/1494452927854312215/1.jpg')  
cv2_imshow(inference(image))
```



```

def video2segemented_video(video_path, filename):
    cap = cv2.VideoCapture(video_path+filename)
    success,img = cap.read()
    if success == False:
        print("No video File")
        return 0
    add_img = np.hstack((img, inference(img)))
    cv2_imshow(add_img)

    frames = []
    add_frames = []
    fps = 30
    height, width, layers = img.shape
    size = (width, height)
    a_height, a_width, a_layers = add_img.shape
    add_size = (a_width, a_height)

    while success:
        success,img = cap.read()
        if success==False: break
        add_img = np.hstack((img, inference(img)))
        add_frames.append(add_img)
        frames.append(inference(img))

    file = filename.split('.')
    outfile = file[0]+"_fin."+file[1]
    outfile_add = file[0]+"_add."+file[1]
    print("Check out the output video from viedo_path on your Google drive!")
    print("output: ", outfile)
    output = cv2.VideoWriter(video_path+outfile, cv2.VideoWriter_fourcc(*'DIVX'), fps, size)
    print("output: ", outfile_add)
    output_add = cv2.VideoWriter(video_path+outfile_add, cv2.VideoWriter_fourcc(*'DIVX'), fps, add_size)
    for i in range(len(frames)):
        output.write(frames[i])
        output_add.write(add_frames[i])
    output.release()
    output_add.release()
    pass

```

```

# Test
video_path = "/content/drive/MyDrive/LaneNetReport/" # input your video path
video2segemented_video(video_path, "test.mp4")
video2segemented_video(video_path, "test2.mp4")

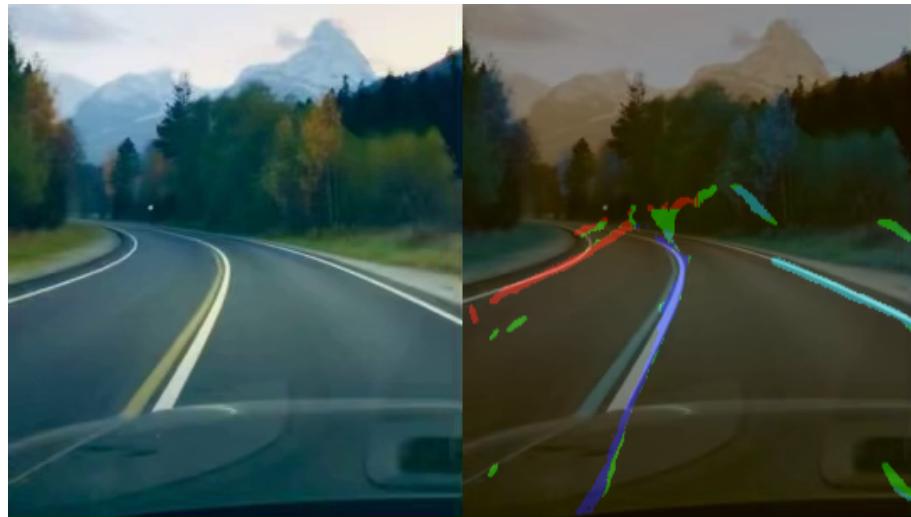
```



Check out the output video from viedo\_path on your Google drive!

output: test\_fin.mp4

output: test\_add.mp4



Check out the output video from viedo\_path on your Google drive!

output: test2\_fin.mp4

output: test2\_add.mp4

## video2segemented\_video 함수 설명

```
def video2segemented_video(video_path, filename):
```

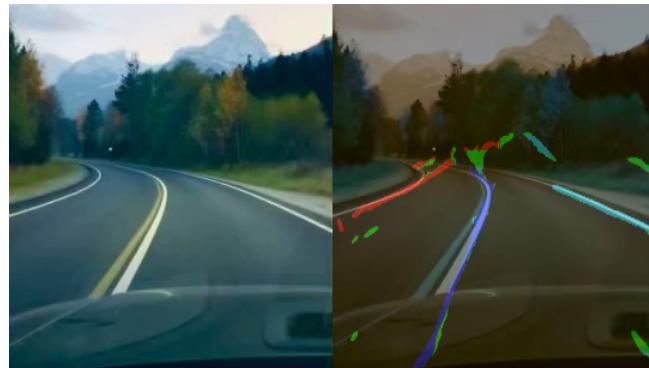
video2segemented\_video 함수의 인자는 video\_path, filename으로 정한다. filename은 동영상 파일명이며, 저장할 영상의 이름을 새롭게 지정해주기 위해 인자를 따로 받도록 설정하였다.

```
cap = cv2.VideoCapture(video_path+filename)
success,img = cap.read()
if success == False:
    print("No video File")
    return 0
```

영상이 성공적으로 읽어지면 success 변수는 True 값을 갖는다. 만약 영상을 제대로 읽지 못한다면 False 값이 되므로 함수가 더이상 실행되지 않고 종료되도록 하였다.

```
add_img = np.hstack((img, inference(img)))
cv2_imshow(add_img)
```

변수 `img` 는 영상을 정상적으로 캡처한 이미지를 담고 있다. `inference(img)` 를 통해 차선을 검출한 이미지를 생성하고, 차선을 검출하지 않은 이미지와 검출한 이미지 2개를 하나의 이미지로 합쳐서 `add_img` 변수에 저장하였다. 차선을 검출하기 전과 후를 쉽게 비교하기 위해 `add_img` 이미지 파일을 출력하였다. 출력한 이미지는 다음과 같다



`add_img` 출력 결과

```
frames = []
add_frames = []
fps = 30
height, width, layers = img.shape
size = (width, height)
a_height, a_width, a_layers = add_img.shape
add_size = (a_width, a_height)
```

함수에 사용할 변수들을 초기화해주었다. `frames` 는 차선 검출한 이미지 데이터를 담을 리스트, `add_frames` 는 검출 전후를 비교한 이미지 데이터를 담을 리스트다. `fps`는 30으로 설정하였고 출력될 영상의 크기는 이미지의 크기를 참고하도록 하였다.

```
while success:
    success, img = cap.read()
    if success==False: break
    add_img = np.hstack((img, inference(img)))
    add_frames.append(add_img)
    frames.append(inference(img))
```

프레임 분할 및 차선 검출을 하는 부분이다. 영상이 끝나거나 영상을 더이상 읽을 수 없으면 `success` 는 `False` 값을 가지며 이 경우 while문을 탈출한다. 차선 검출 전후를 비교한 이미지 데이터인 `add_img` 는 `add_frames` 리스트에 추가하고, 차선 검출한 이미지(`inference(img)`)는 `frames` 리스트에 추가한다.

```
file = filename.split('.')
outfile = file[0]+"_fin."+file[1]
outfile_add = file[0]+"_add."+file[1]
```

```

print("Check out the output video from viedo_path on your Google drive!")
print("output: ", outfile)
output = cv2.VideoWriter(video_path+outfile, cv2.VideoWriter_fourcc(*'DIVX'), fps, size)
print("output: ", outfile_add)
output_add = cv2.VideoWriter(video_path+outfile_add, cv2.VideoWriter_fourcc(*'DIVX'), fps, add_size)
for i in range(len(frames)):
    output.write(frames[i])
    output_add.write(add_frames[i])
output.release()
output_add.release()
pass

```

동영상으로 변환 및 저장하는 부분이다. 새로운 파일명을 설정해주기 위해 filename을 split하였다. `outfile`은 차선 검출 영상 파일명, `outfile_add`는 차선 검출 전후를 비교한 영상 파일명 변수이다. 각각 `filename_fin.mp4`, `filename_add.mp4`로 저장되도록 하였다. openCV의 `VideoWriter()`을 활용하여 영상 변환 후 `write()`를 통해 파일을 저장한다. MPEG-4로 출력하기 위해 DIVX 코덱을 사용하였고, fps와 영상 size는 위에서 미리 초기화한 것과 같다.

## 출력 결과

코드(Colab) 및 출력 파일은 git Hub에 첨부되어 있다. 영상은 free 소스를 활용하였다. 아쉽게도 차선이 아닌 부분을 차선으로 인식하는 경우가 있었다. 영상의 잡음 제거를 위한 알고리즘(Gray Scale, 필터 추가 등)을 추가한다면 더욱 깔끔한 결과를 얻을 수 있을 것 같다.

ShimHyerin/2021-VehicleIntelligence  
Contribute to ShimHyerin/2021-VehicleIntelligence development by creating an account on GitHub.

<https://github.com/ShimHyerin/2021-VehicleIntelligence/tree/main/HW3>

1 Contributors 0 Issues 0 Stars 0 Forks