Lane detection & Lane Annotation

Date 2023. 7 18

DGIST

Division of Automotive Technology

Lane detection & Lane Annotaion

Revision History			
Date	Duty	Author	Description
2023.7.4	Research Engineer	Junyeong, Kim	초안
2023.7.18	Research Engineer	Junyeong, Kim	이슈 사항 추가

References		
Site or name	description	date
https://github.com/Turoad/CLRNet	CLRNet git	23.7.4
https://xingangpan.github.io/projects/CULane.html	CULane dataset	23.7.4
https://github.com/wkentaro/labelme	labelme(Annotation Tool)	23.7.4

Host development Environment	
Item	Name
Middelware	= Ros melodic
OS	=< Ubuntu Linux 18.04 / 64 bit
Anaconda	
GPU 아키텍처	=< 파스칼

= : It must be set(Same version).
=< : If the version or performance is better than this one, It's fine.
≒ : If the version or performance is similar this one, It's fine.

I. Lane Detection

Lane Detection을 수행하기 위해 개발당시 CRLNet이 가장 성능이 좋은 딥러닝 모델이었기에 이 모델을 사용함.

A. 개발환경 설정 & Run (Anaconda)

1. Create a conda virtual environment and activate it (conda is optional)

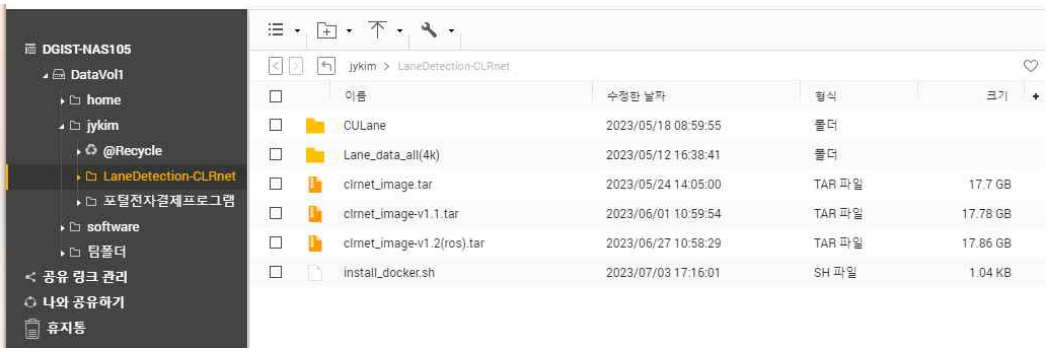
```
$ conda create -n clrnet python=3.8 -y          # 새 terminal을 엶
$ conda activate clrnet                         # clrnet이름으로 conda가상 환경 설정
                                                # clrnet가상환경 활성화
```

2. Install dependencies

```
$ conda install pytorch torchvision cudatoolkit=10.1 -c pytorch   # 새 terminal을 엶
$ pip install torch==1.8.0 torchvision==0.9.0                     # Install pytorch firstly, the cudatoolkit version should be same in your system.
$ python setup.py build develop                                   # Or you can install via pip
                                                                  # Install python packages
```

B. 개발환경 설정 & Run (Docker)

105번 NAS에서 DataVol1/jykim/LaneDetection-CLRnet 경로내 clrnet_imge-v1.2(ros).tar 파일과 install_dockeer.sh 파일을 다운 받는다.



1. Create docker environment

```
$ cd [sh파일이 존재하는 경로]          # 새 terminal을 엶
$ bash install_docker.sh              # install_docker.sh가 존재하는 경로로 이동
                                      # sh파일을 이용하여 docker환경 설치
```

* 설치 후 재부팅하는 것을 권장한다.
* Ubuntu 18.04, Ubuntu 22.04에서 정상작동함을 확인함.

2. Install docker image

```
$ cd [docker image 파일이 존재하는 경로]          # 새 terminal을 엶
$ docker load -i clrnet_image-v1.2(ros).tar      # docker image 파일이 있는 경로로 이동
$ docker images                                  # 다운 받은 docker image 설치
                                                 # 설치된 docker환경 확인
```

C. clrnet Run (Anaconda)

1. Source code download

120번 NAS/intern/폴더/Lane_detection(CLRNet)경로의 clrnet_ws 파일을 다운로드.

2. CLRNet Run

clrnet_ws/src/clrnet/src/detect2.py 파일 수정.

```python
class Detect(object):
    def __init__(self):
        cfg = Config.fromfile("/home/dgist1000/clrnet_ws/src/clrnet/src/configs/clrnet/clr_dla34_culane.py")
        cfg.load_from = "/home/dgist1000/clrnet_ws/src/clrnet/src/CULane.pth"
        cfg.show = True
        cfg.savedir = "/home/dgist/test"

        self.cfg = cfg
        self.processes = Process(cfg.val_process, cfg)
        self.net = build_net(self.cfg)
        self.net = torch.nn.parallel.DataParallel(self.net, device_ids = range(1)).cuda()
        self.net.eval()
        self.bridge = CvBridge()
        load_network(self.net, self.cfg.load_from)

        self.image_sub = rospy.Subscriber("/sensing/camera/traffic_light/image_raw", Image, self.imageCb, queue_size = 10, buff_size = 2**24)
        self.image_pub = rospy.Publisher("/lane_image", Image, queue_size=10)
```

* clr_dla34_culane.py file의 경로와 XX.pth file의 경로를 확인하여 주소 변경.
* input topic명과 output topic명 변경 후 저장.
* 현재 일부 파라미터 변경해야하지만 테스트에는 문제 없음.

```
                                                # 새 terminal을 엶
$ conda activate clrnet                         # clrnet가상환경 활성화
$ cd clrent_ws                                  # 해당 폴더로 이동
$ catkin_make                                   # ros package build
& source devel/setup.bash                       # 해당 package를 sourcing
$ python src/clrent_ws/src/detect2.py           # 해당 프로그램 실행
```

D. clrnet Run(Docker)

준비중.

E. clrnet train(Anaconda)

```
                                                                                  # 새 terminal을 엶
$ conda activate clrnet                                                           # clrnet가상환경 활성화
$ cd clrent/src                                                                   # 해당 폴더로 이동
$ sudo python main.py /home/dgist/src/configs/clrnet/clr_dla34_culane.py --gpus 0 # 학습 실행
```

*PC내 clr_dla34_culane.py가 존재하는 경로 적어줌.

F. clrnet train(Docker)

```
                                                                                  # 새 terminal을 엶
$ cd clrnet/docker                                                                # 해당 폴더로 이동
$ bash docker_run.sh                                                              # Docker실행
$ cd ~/catkin_ws/src                                                              # 해당 폴더로 이동
$ sudo python main.py /home/dgist/catkin_ws/src/configs/clrnet/clr_dla34_culane.py # 학습 실행
  --gpus 0
```

* Docker환경 내 clr_dla34_culane.py가 존재하는 경로 적어줌.

학습 완료시 work_dirs폴더가 생성되고 폴더 내부에 log파일과 학습파라미터가 존재한다.

G. Train tip

1. clr_dla34_culane.py 파라미터 수정

변수명	Description
ori_img_w	raw 이미지 width
ori_img_h	raw 이미지 height
bbox_h_start	차선이 보이기 시작하는 부분분으로 조절하면 됨
epochs	학습 반복 횟수 30회 이상 권장. 데이터가 더 많을수록 반복횟수 상향고려.
batch_size	gpu에 할당 가능한 데이터 크기 TiTAN pascal 1개 기준 32로 설정.
cut_height	차선과 관계없는 부분 자르는 용도

2. 다중 GPU사용.

학습 명령어 사용시 다중 GPU사용가능. --gpu 0 1 과 같이 명령어 사용가능. GPU 3개 장착시 --gpu 0 1 2 옵션 사용.

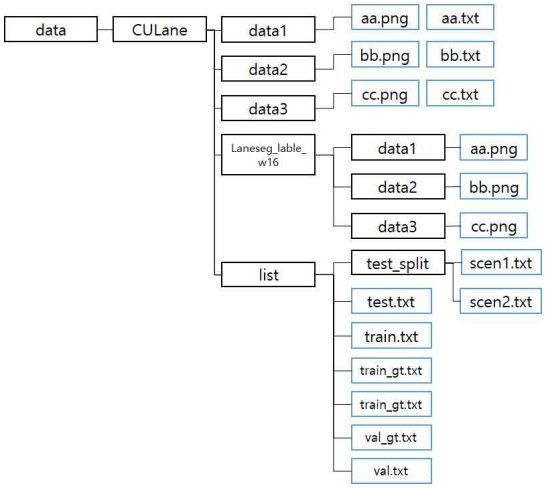
```
$ sudo python main.py home/dgist/catkin_ws/src/configs/clrnet/clr_dla34_culane.py --gpu 0 1
  # 명령어 예시(GPU 2개 사용시)
```

3. 기존 학습된 데이터 사용

기존에 학습된 파라미터에 추가로 학습하고 싶은 경우 --finetune_from [pth파일 경로]

```
$ sudo python main.py home/dgist/catkin_ws/src/configs/clrnet/clr_dla34_culane.py --finetune_form ABC.pth
  # 명령어 예시(ABC.pth의 학습 데이터를 가져와서 추가 학습 하고 싶은 경우)
```

4. 데이터셋 구성



학습 데이터는 위의 그림과 같이 구성한다. 현재는 val.txt와 test.txt의 구성을 동일하게 구성함.

5. 평가 및 test set draw

```
$ python main.py /home/dgist/catkin_ws/src/configs/clrnet/clr_dla34_culane.py --validate --load_from /home/dgist/catkin_ws/src/29.pth --gpus 0 --view
  # 명령어 예시(--view 옵션을 통해서 test set draw 가능)
```

6. Tensorboard 활용

학습을 마치고 나면 runs라는 폴더가 생기게됨. sudo python -m tensorboard.main --logidr=. 명령어를 사용하여 tensorboard활용

```
$ sudo python -m tensorboard.main --logidr=.
  # 상기 명령어 후 나오는 주소를 통해서 학습 상태 파악 가능.
```

7. GPU에 따른 환경 구성

GPU가 변경되면 가용한 cuda version이 변경되면서 연계되어 있는 library랑 dependency등을 모두 확인을 해야한다. 개발 당시에 Titan X (pascal)을 사용했으나 RTX 3080Ti에서 error가 발생.

GPU	Compute Capability
GeForce RTX 4090	8.9
GeForce RTX 4080	8.9
GeForce RTX 4070	8.9
GeForce RTX 4060	8.9
GeForce RTX 4050	8.9
GeForce RTX 3080 Ti	8.6
GeForce RTX 3080	8.6
GeForce RTX 3070 Ti	8.6
GeForce RTX 3070	8.6
GeForce RTX 3060 Ti	8.6
GeForce RTX 3060	8.6
GeForce RTX 3050 Ti	8.6
GeForce RTX 3050	8.6

Compute Capability (CUDA SDK support vs. Microarchitecture)

CUDA SDK version(s)	Tesla	Fermi	Keppler (early)	Kepler (late)	Maxwell	Pascal	Volta	Turing	Ampere	Ada Lovelace	Hopper
1.0	1.0 – 1.1										
1.1	1.0 – 1.1+x										
2.0	1.0 – 1.1+x										
2.1 - 2.3.1	1.0 – 1.3										
3.0 - 3.1	1.0 –	2.0									
3.2	1.0 –	2.1									
4.0 - 4.2	1.0 –	2.1+x									
5.0 - 5.5	1.0 –			3.5							
6.0	1.0 –			3.5							
6.5	1.1 –				5.x						
7.0 - 7.5		2.0 –			5.x						
8.0		2.0 –				6.x					
9.0 - 9.2			3.0 –				7.0				
10.0 - 10.2			3.0 –					7.5			
11.0				3.5 –					8.0		
11.1 - 11.4				3.5 –					8.6		
11.5 - 11.7.1				3.5 –					8.7		
11.8				3.5 –							9.0
12.0					5.0 –						9.0

상기 이미지를 확인하면 RTX 3080 Ti의 Compute Capability가 8.6인 것을 확인 할 수 있고 Compute Capability가 8.6일대 사용한 CUDA version은 11.1~11.4라는 것을 확인 할 수 있다. 즉, CUDA 11.1~11.4를 기반으로 다른 library와 dependency의 version을 확인해서 재설치 해야한다.

II. Lane Annotation(Ubuntu)

A. Install

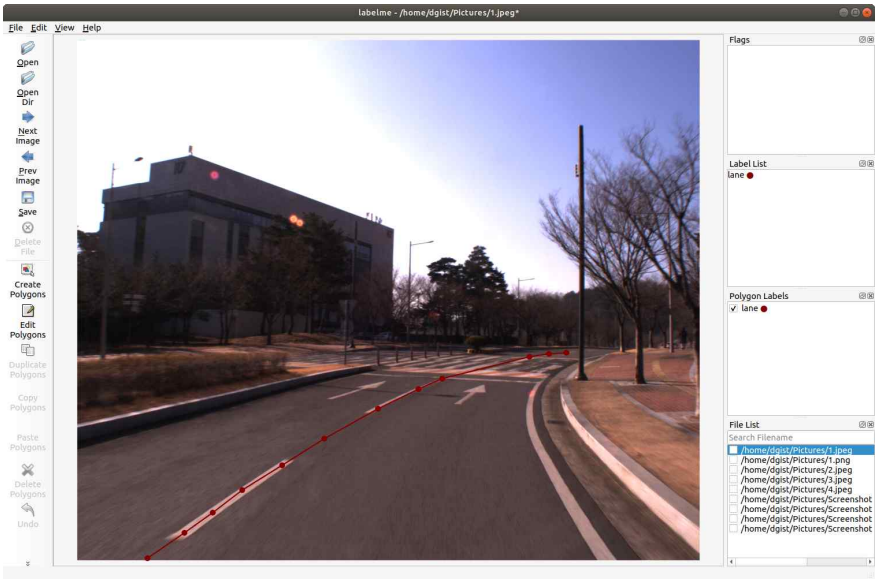
```
                                                    # 새 terminal을 엶
$ conda create —name=labelme python=3               # labelme이름으로 conda가상 환경 설정
$ source activate labelme                           # labelme가상환경 활성화
$ pip install labelme                               # labelme설치
```

B. Data Annotation

원본 이미지는 'png', label data는 'json'이라는 이름으로 폴더를 생성하여 관리한다.

```
                                                    # 새 terminal을 엶
$ conda activate labelme                            # labelme가상환경 활성화
$ labelme                                           # labelme실행
```

1. File -> Save With Image Data 체크박스 해제
2. File -> Change Output Dir통해 원하는 저장 공간 선택
   'json'이라는 이름의 폴더를 생성하고 그 경로에 저장
3. Edit -> Create LineStrip 이용 차선의 중심을 찍고 'Enter'키 누름



4. Group ID에 lane 으로 기재 후 OK 버튼 누름
5. 모든 Annotation대상에 대해서 2.~3.반복 후 완료가 되었으면 Next Image 클릭 후 Save
6. 해당 경로에 [이미지명].json file 생성 확인

C. json to txt data

현재 연구소 git을 사용할 수 없어서 NAS120/*** 경로에 업로드함.

1. labelme2culane 폴더 다운로드
2. png폴더와 json폴더는 labelme2culane폴더 내부에 위치 시킨다.

```
                                                    # 새 terminal을 엶
$ conda create —name=convert38 python=3.8           # convert38이름으로 conda가상 환경 설정
$ source activate convert38                         # convert38가상환경 활성화
$ cd labelme2culane                                 # lableme2culane 폴더로 이동
$ python MakePoints_val.py                          # MakePoints.py 실행
$ python Make_list.py                               # Make_list.py실행
```

txt폴더에 txt가 생성됨.

*Make_list.py 실행전 파일 내부에 root_path 파라미터 수정.

D. Viewer

기본 환경 셋팅은 C와 동일함. Point가 정상적으로 생성이 되었나 확인하기 위한 작업.

```
                                                # 새 terminal을 엶
$ conda create —name=convert38 python=3.8       # convert38이름으로 conda가상 환경 설정
$ source activate convert38                     # convert38가상환경 활성화
$ cd labelme2culane                             # lableme2culane 폴더로 이동
$ python Viewer.py                              # Viewer.py 실행
```

display 폴더에 이전단계에서 생성된 Point가 이미지에 표시되어 있음.

E. txt to label data

https://github.com/XingangPan/seg_label_generate CULane 공식 홈페이지 open source 사용

우리 이미지에 맞게 img size조절함.

```
                                                # 새 terminal을 엶
$ cd seg_label_generate                         # seg_label_generate폴더로 이동
$ make                                          # if you prefer makefile
$ mkdir build && cd build                       # or, if you prefer cmake
$ cmake ..
$ bash labelGen.sh                              # labelGen.sh실행
$ bash listGen.sh                               # listGen.sh 실행
```

* labelGen.sh, 와 listGen.sh실행전 파일을 열어서 경로랑 해당 파일 수정 필요.

CULane=/AAA/BBB/CCC/DDD

-l ${CULane}/list/train.txt

/AAA/BBB/CCC/DDD/list경로내에 Ⅱ.C에서 Make_list.py로 생성한 파일 2개가 존재해야함.

train.txt, test.txt에 대해서 각각 실행.

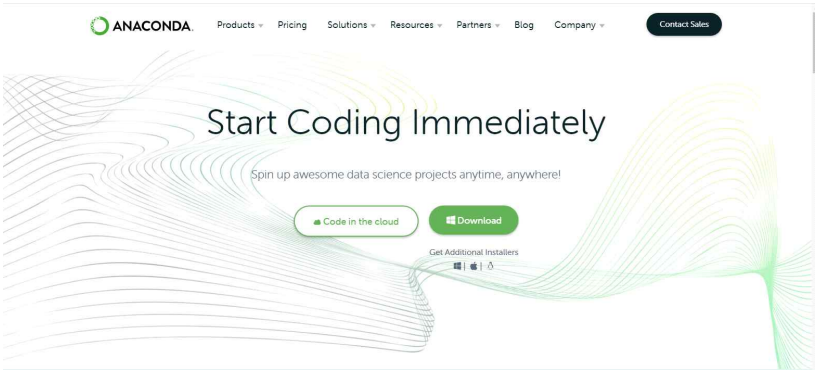
Ⅰ.G.4에 필요한 laneseg_label_w16파일과 list파일 생성됨.

시나리오별 구분은 수동으로 해야함.

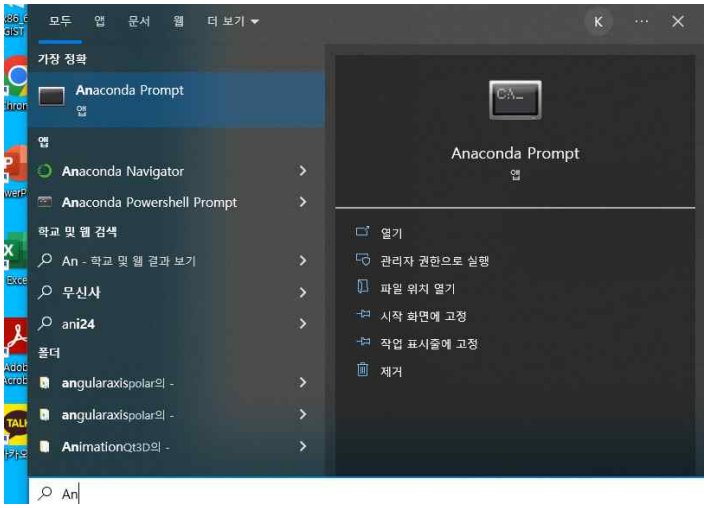
III. Lane Annotation(Window)

A. Install

1. https://www.anaconda.com/ 접속



2. Download 클릭
3. 'Anaconda3-2023.03-Windows-x86_64' 다운로드된 파일 실행
4. 모두 기본설정으로 설치
5. Anaconda Prompt 실행




```
$ conda create --name=labelme python=3          # labelme이름으로 conda가상 환경 설정
$ conda activate labelme                        # labelme가상환경 활성화
$ pip install labelme                           # labelme설치
```

B. Run

1. Anaconda Prompt 실행

```
$ conda activate labelme                        # labelme가상환경 활성화
$ labelme                                       # labelme실행
```

IV. Lane Annotation규칙

A. 최대한 차선의 정 중앙을 찍도록 노력한다.

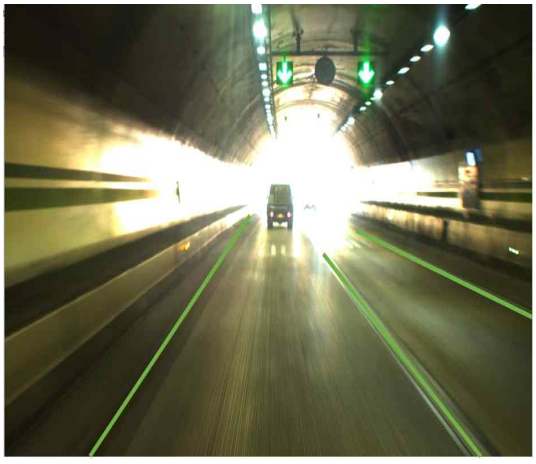
B. 점은 최대한 균일하게 차선 1개당 10개이상 점찍기.

C. 차선 기준은 내가 주행하는 방향만, 내 차선 양옆 차선 최대 4개까지만 작업.

(그림에서 작업을 진행해야 할 차선은 총 3개)

D. 차량이 차선을 가릴 경우 차선이 존재한다고 생각하고 작업 진행.

E. 빛이 차선을 가릴 경우 보이는 부분 까지만 작업 수행.



F. 점선 실선 노랑 흰색 상관없이 작업 진행.
G. 차선이 2겹일 경우 나랑 가까운 부분만 작업 진행.

H. 차선이 보이는 끝까지 최대한 작업.