

1) 자율주행 인지에 관련된 3종 이상의 공개 Data Set 조사, 정리

- 데이터 설명, 양, 세부 요소 (Feature), 데이터 예시, 활용 예 등 데이터를 이해하는데 필요한 정보

1. cityscapes dataset

공식 사이트: <https://www.cityscapes-dataset.com/>

git repository (cityscapesScripts):

<https://github.com/mcordts/cityscapesScripts>

데이터 설명: 50개의 다른 도시에 대한, 주행 정보를 담고 있는 데이터 셋

데이터 종류 및 양:

주행비디오

5000여장의 픽셀 단계에서의 높은 수준의 폴리곤 데이터

20000여장의 괜찮은 수준의 폴리곤 데이터 제공

세부요소: 데이터는 이렇게 정리되어 있다.

{root}/{type}{video}/{split}/{city}/{city}_{seq:0>6}_{frame:0>6}_{type}{ext}

{root}: 데이터의 가장 상위 폴더이름

{type}: 데이터타입 (가령, leftImg8bit는 좌측 8비트 이미지이다.) 자세한 사항은

공식 레퍼지토리의 README.md 파일의 설면 참고

{video}: 영상

{split}: 데이터가 있는 폴더위치 (인 것 같다..)

{city}: 데이터가 기록된 도시

{city}_{seq:0>6}_{frame:0>6}_{type}{ext}:{도시이름}_{sequence
number}_{frame number}_{데이터타입}{확장파일..?}

데이터 예시: 5000여장의 high-leveled annotation 데이터 예시



20000여장의 fine annotation 데이터 예시



다양한 종류의 비디오

<https://www.cityscapes-dataset.com/examples/#dense-pixel-annotations>

사이트 참고

활용:

공식사이트를 통해 데이터를 다운 받을 수 있으며

cityscapesscripts라는 파이썬 모듈로 사용가능 사용가능하다.
명령어와 콘텐츠는 공식 git repository 참고

2. Barkeley DeepDrive

공식사이트 : <https://bdd-data.berkeley.edu/index.html>

git repository (bdd100k) :

<https://github.com/bdd100k/bdd100k>

데이터 설명 : 100K수준의 1100시간가량의 다양한 환경에서의 주행 비디오 clip을 가진 데이터셋

데이터 종류 및 양:

1100시간가량의 다양한 환경에서의 주행 비디오 clip

100,000개의 2D 박스 주석이 달린 이미지 데이터

10,000여개의 높은 수준의 폴리곤 데이터

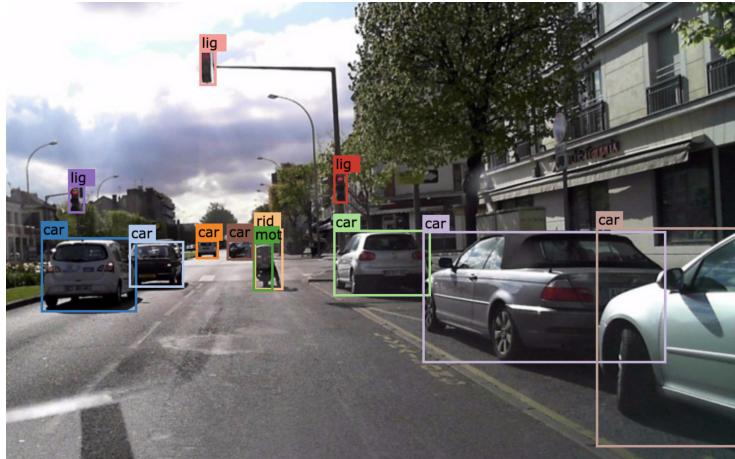
100,000주행가능 도로 데이터

100,000여개의 도로선이 표시된 이미지 데이터

세부요소: 비디오 뿐 아니라 GPS와 IMU data, 시간대를 제공한다.

데이터 예시:

100,000개의 2D 박스 주석이 달린 이미지 데이터



10,000여개의 높은 수준의 폴리곤 데이터



100,000주행가능 도로 데이터



100,000여개의 도로선이 표시된 이미지 데이터



활용:

공식사이트를 통해 데이터의 다운이 가능하며,

git repository의 toolkit을 활용할 수 있다.

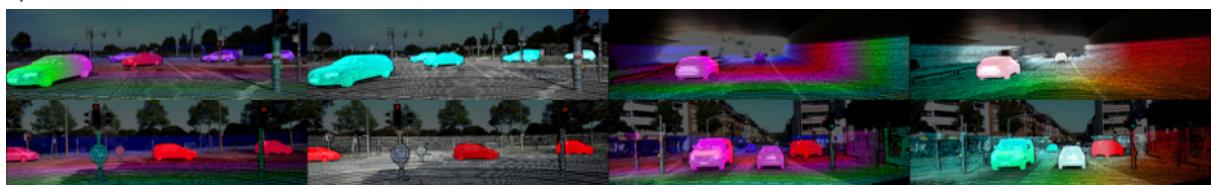
3. The KITTI Vision Benchmark Suite

공식사이트 : <http://www.cvlibs.net/datasets/kitti/>

데이터설명 : 독일의 중소도시 Karlsruhe에서 기록된 데이터를
optical flow, visual odometry, 3D object detection ,3D tracking 등
에 관한 각각의 데이터를 제공
laser scanner와 GPS 등을 활용한 정보를 포함하고 있다.

데이터 예시:

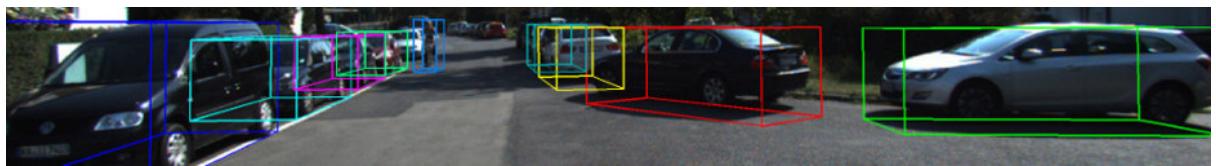
optical flow



visual odometry



3D object detection



Semantic Segmentation Evaluation



활용 : 공식 사이트의 각각의 카테고리에 데이터와 tool kit(development kit)이 함께 제공되고 있다.

2) 자율주행 인지에 관련된 2종 이상 Open Source 조사, 정리

- 코드 설명, 구성, 활용 및 결과 등 코드를 이해하는데 필요한 정보

1. OpenCV를 활용한 차선인식 코드

OpenCV : <https://opencv.org/>

git repository(road_lane_line_detection) :

https://github.com/georgesung/road_lane_line_detection

코드설명:

코드는 컴퓨터 비전에 대한 오픈 소스 라이브러리인 OpenCV를 이용하여 현재 주행 차량의 차선을 인식하는 코드이다.

이 코드는 openCV의 함수를 이용한 4가지 과정을 통해 차선을 표시한다

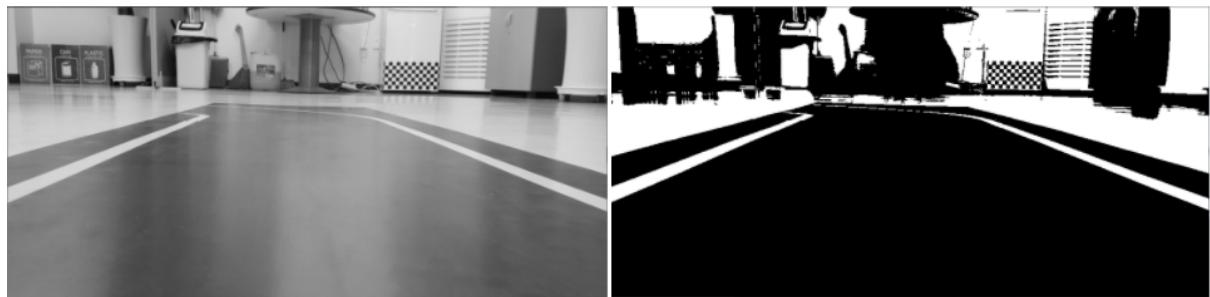
1. 색상 제거(흑백 화면으로 변경) (활용 함수 cvtColor())

화면을 단순화 시키기 위해 우선 색을 제거한다.



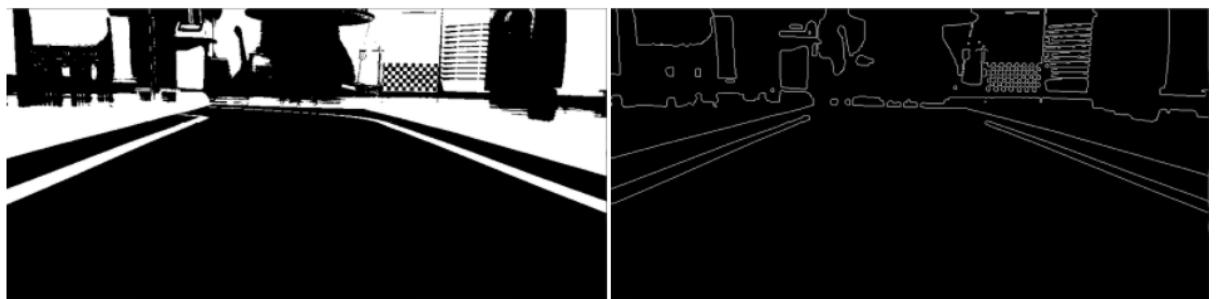
2. 이진화 (회색부분을 없애고 흰색 혹은 검은 색으로 극단적으로 바꿈) (활용함수 threshold())

화면을 더욱 단순하게 표현하여 해석을 용이하게 하기위해 다양한 회색으로 표현되던 화면을 흰색과 검은색만으로 구분시킨다.



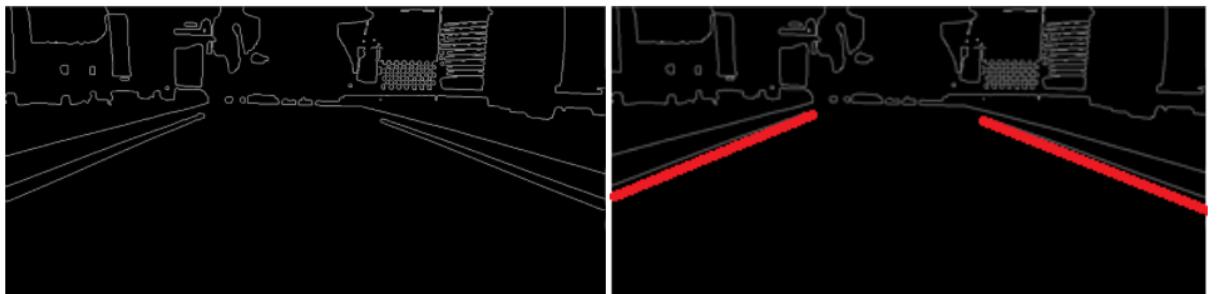
3. 윤곽선 검출 (활용 함수 Canny())

흰색과 검은색으로 나눠진 이미지의 윤곽선을 추출한다.



4. 직선 검출 (활용함수 HoughLines())

윤곽선에서 하단에서 상단으로 스캔하며 가장 먼저 검출되는 직선을 검출하여 차선으로 판단합니다.



활용 : 검출한 차선의 각도 정보를 이용하여 주행 방향을 결정하는 등의 활용이 가능할 것이다.

2.YOLO(You Only Looks Once) (이미지(?)처리를 위한 오픈소스 라이브러리)

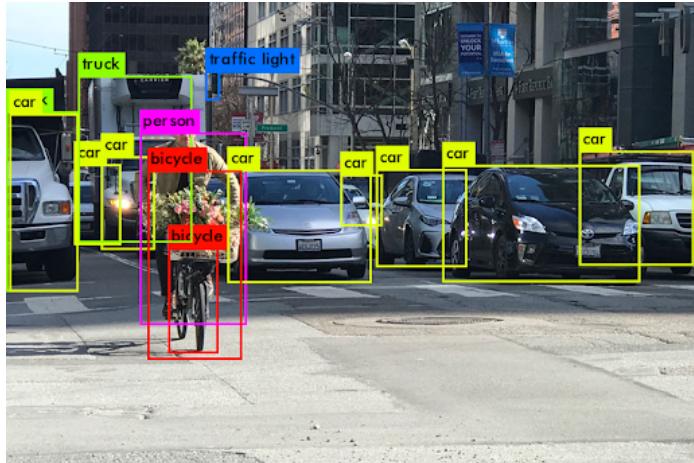
공식사이트 : <https://pjreddie.com/darknet/yolo/>

git repository(darknet):

<https://github.com/pjreddie/darknet>

설명 : 컴퓨터 비전에 관한 오픈소스로 이미지의 물체(object)들을 빠르게 인식하기 위한 딥러닝 기반의 소스이다.

활용 : 이미지를 빠르게 인식하므로 실시간으로 도로위 물체를 인식할 때 사용할 수 있을 것이다.



3) 2)의 정리한 코드 중 하나 실행해서 결과 확인

- 구현 환경 (시스템, lib, docker 등 실행 환경) 및 실행에 관련된 코드 설명 등

2)의 1번 코드인 openCV를 이용한 차선추출 코드를 실행한 결과를 확인했다.

코드의 내용은 2번에서 설명한 내용을 따르며

실행환경은 코드의 git repository의 readme에 따라

python 3.5이상과 Numpy, OpenCV, Matplotlib, MoviePy 라는 모듈이 필요하다.

실행은 이미지의 경우 명령 프롬프트에서 해당 명령어로 실행 가능하며

```
python lane_lines.py -i input_image.jpg -o output_image.jpg -I
```

input_image.jpg에 확인할 이미지 경로를

output_image.jpg에 차선표시된 이미지가 저장될 경로 및 이름을 입력하면 된다.

동영상의 경우 아래 명령어를 이용한다.

```
python lane_lines.py -i challenge.mp4 -o extra.mp4
```

challenge.mp4에 확인할 동영상 경로를

extra.mp4에 차선표시된 동영상의 경로와 이름을 입력하면 된다.

4) 1)~3) 내용 본인 GIT에 공개하기

git 주소(report1 폴더에 과제 관련 코드 있습니다):

https://github.com/kwonbuyeon/2021-1_basicCarAI/tree/master