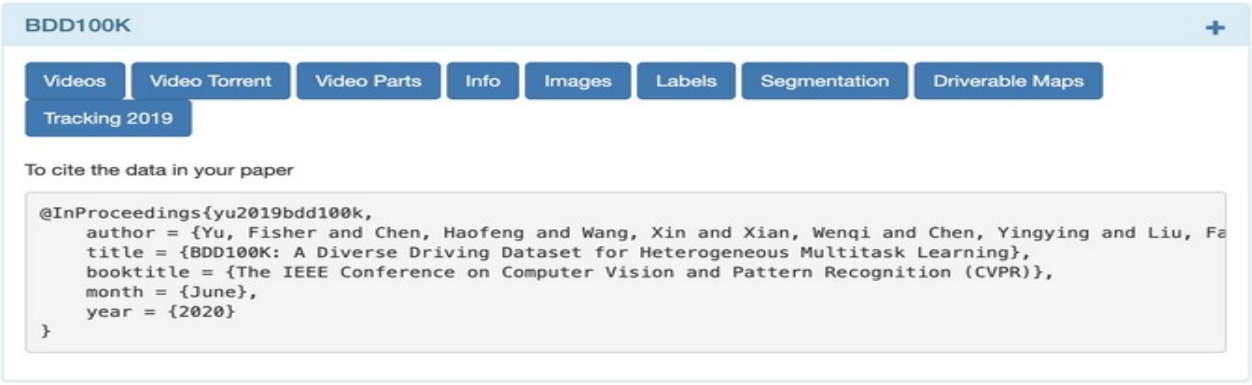
1.Datasets For Autonomous Driving

1.Berkeley DeepDrive

UC Berkeley 의 Berkeley DeepDrive 데이터 세트는 이미지 수준 태깅, 개체 경계 상자, 운전 가능 영역, 차선 표시 및 전체 프레임 인스턴스 분할을 포함한 다양한 종류의 주석이 포함 된 10 만 개 이상의 비디오 시퀀스로 구성됩니다.

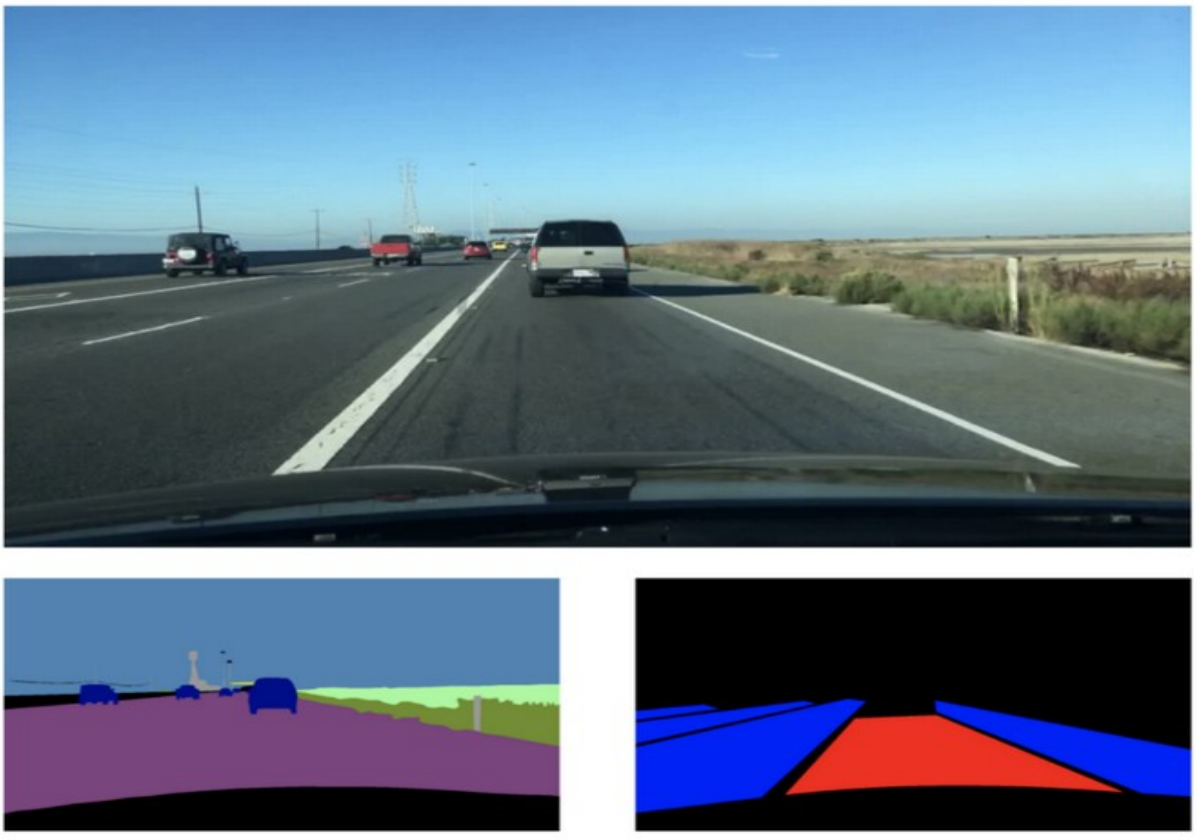
BDD 100K 포털에 로그인하면 다음과 같은 페이지에 액세스 할 수 있습니다.



버튼을 클릭하면 관련 데이터를 다운받을 수 있습니다. 처음 3 개의 폴더 Image, Segmentation 및 Driverable Maps 에는 이미지와 세분화 이미지가 포함되어 있습니다.세분화는 맵을 생성하여 각 픽셀에 클래스를 할당하는 것이 목표 인 알고리즘 유형입니다.

따라서 레이블은 각 픽셀이 클래스를 나타내는 하나의 색상으로 구성된 이미지입니다.

`Image>Label becomes Image>Image.`



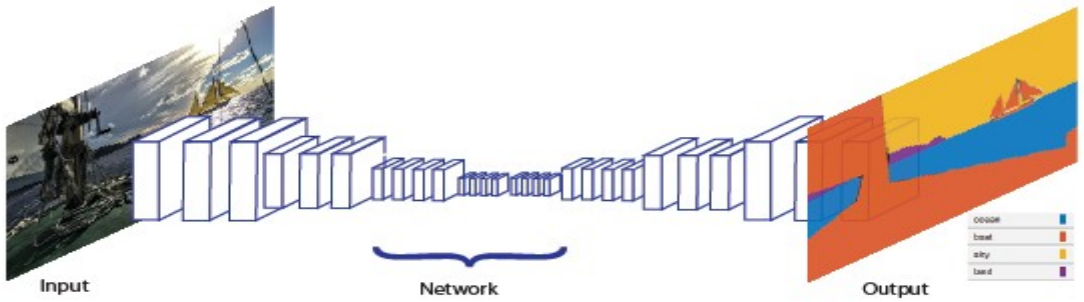
Image, Segmentation, and Driveable Area labels from BDD100K

이미지는 1280x720 RGB 이미지입니다.이미지 파일의 이름은 모든 라벨 이미지에서 찾을 수있는 특정 번호입니다.
시맨틱 세분화에서 각 픽셀은 하나의 객체입니다.여기서 세분화 유형은 풀 프레임 세분화이므로 도로는 보라색, 하늘은 파란색, 자동차는 마린.

Driverable Maps는 차선을 나타냅니다.빨간색 픽셀은 운전해도 괜찮은 자아 차선을 나타냅니다. 파란색 픽셀은 인접한 차선을 나타냅니다.

이미지 데이터 사용

이러한 유형의 데이터를 사용하려면 의미 론적 분할 알고리즘이 필요합니다. 이는 일반적으로 인코더 / 디코더 아키텍처에 의해 수행됩니다.



Semantic Segmentation Network

JSON 파일의 이해

레이블 버튼을 클릭하면 두 개의 JSON 파일이 다운로드됩니다. 하나는 학습용이고 다른 하나는 검증용입니다.
JSON 파일은 특정 형식의 정보에 액세스 할 수있는 파일입니다.
실제 컴퓨터 비전 엔지니어가 되려면 JSON 파일의 정보에 액세스 할 수 있어야합니다.

```
“name”: “0a0a0b1a-7c39d841.jpg”,
 “attributes”: {
 “weather”: “clear”,
 “scene”: “highway”,
 “timeofday”: “daytime” >
 },
```

첫 번째 부분에서는 정보가 일반적으로 장면에 대한 것임을 알 수 있습니다.

다음은 이러한 요소에 대한 데이터 세트 분포입니다.

Distribution

다음으로 장애물 레이블에 액세스합니다.


```
"timestamp": 10000,
 "labels": [
 {
 "category": "car",
 "attributes": {
 "occluded": true,
 "truncated": false,
 "trafficLightColor": "none"
 },
 "manualShape": true,
 "manualAttributes": true,
 "box2d": {
 "x1": 555.647397,
 "y1": 304.228432,
 "x2": 574.015906,
 "y2": 316.474104
 },
 "id": 109344
 },
```


두 번째 부분은 장애물 감지에 많은 도움이 될 것입니다.
각 장애물은 경계 상자로 완전히 설명됩니다.
모든 장애물에는 동일한 정보가 있습니다.

레이블 지정 도구를 사용하여 경계 상자를 수동으로 그리고 파일에 정보를 추가했습니다.

(Image from BDD 100K Paper)

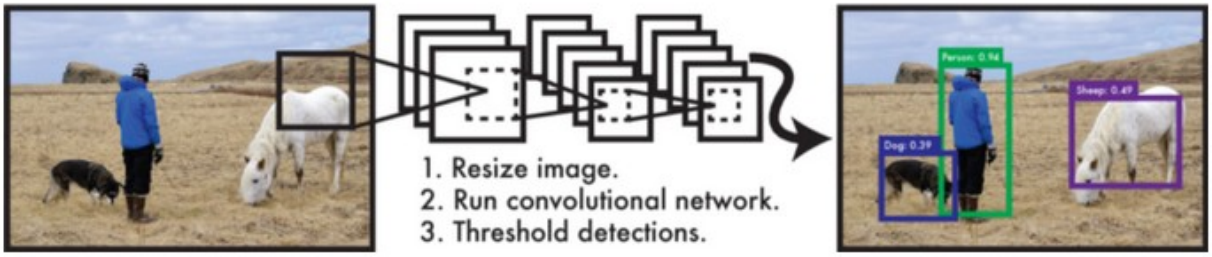
보시다시피 JSON 의 모든 정보는 레이블 지정 도구를 사용하여 제공 할 수 있습니다.

다음은이 데이터 세트의 장애물 분포입니다.

Distribution

Labels 사용

이러한 유형의 정보를 사용하려면 YOLO, SSD 또는 FASTER RCNN 과 같은 장애물 감지 알고리즘이 필요합니다.



YOLO

Driverable Areas

```
{ “category”: “drivable area”,
 “attributes”: {
 “areaType”: “direct”
 },
 “manualShape”: true,
 “manualAttributes”: true,
 “poly2d”: [
 {
 “vertices”: [
 [
 344.484243,
 615.550512
 ],
 … (similar content) …
 [
 344.484243,
 615.550512
 ]
 ],
 “types”: “LLLLCCC”, > Type of the lane
 “closed”: true
 }
 ],
 “id”: 109351
 },
```

각 레인에 대해 이와 같은 하나의 단락이 파일에 존재합니다.
여기서도 도로를 정확하게 표현하기 위해 차선 지점을 클릭합니다.
JSON 파일에 표시되는 각 지점은 차선 가장자리에있는 지점입니다.

(a) Polygon with dispersed clicks.

이는 구동 가능 영역 감지에 사용되지만 전체 프레임 시맨틱 분할에는 사용되지 않습니다.

전체 프레임 시맨틱 분할을위한 레이블을 원하는 경우 이미지를 참조해야합니다.

Driverable Areas 사용

Driverable areas의 경우 가장 좋은 방법은 이전에 보인 것처럼 의미 론적 분할을 이용하는 것입니다. 이 JSON 정보를 사용하는 경우 차선 곡선 방정식을 예측할 수 있지만 데이터 유형에 대한 강력한 모델 정의가 필요합니다.

2.Landmarks

Google은 인간이 만든 자연 명소를 인식하기 위해이 데이터 세트를 오픈 소스로 제공했습니다.이 데이터 세트는 2018 년 랜드 마크 인식 및 랜드 마크 검색 Kaggle 챌린지의 일부로 출시됩니다.Google Landmarks 데이터 세트 v2는 5 백만 개 이상의 이미지 및 20 만 개 이상의 고유 한 인스턴스 라벨으로 이루어져 있다
.데이터 세트는 인식과 검색이라는 두 가지 다른 컴퓨터 비전 작업을 평가하기 위해 두 세트의 이미지로 나뉩니다. 데이터는 원래 [1]에서 설명되었으며 Google Landmark Recognition Challenge 및 Google Landmark Retrieval Challenge의 일부로 게시되었습니다.

데이터 세트에는 온라인에서 공개적으로 사용할 수있는 이미지 URL이 포함되어 있습니다.데이터 세트에는 테스트 이미지, 학습 이미지 및 색인 이미지가 포함됩니다. 테스트 이미지는 두 작업 모두에서 사용됩니다. 인식 작업의 경우 각 테스트 이미지에 대해 랜드 마크 라벨이 예측 될 수 있습니다. 검색 작업을 위해 각 테스트 이미지에 대해 관련 인덱스 이미지가 검색 될 수 있습니다. 학습 이미지는 랜드 마크 라벨과 연결되며 인식 및 검색 문제에 대한 모델 학습에 사용할 수 있습니다 (학습 이미지의 지리적 분포 시각화).인덱스 이미지는 검색 작업에 사용되며 이미지를 검색해야하는 집합을 구성합니다.

데이터 셋 구성

훈련 및 인덱스 세트는 알고리즘을 사용하여 지리적 위치 및 시각적 유사성과 관련하여 사진을 클러스터링하여 구성되었습니다.훈련 이미지 간의 일치는 로컬 특성 일치를 사용하여 설정되었습니다. 랜드 마크 당 여러 클러스터가있을 수 있으며 일반적으로 랜드 마크의 다른 뷰 또는 다른 부분에 해당합니다. 편견을 피하기 위해 Ground Truth 생성에 컴퓨터 비전 알고리즘을 사용하지 않았습니다. 대신, 인간 어노 테이터를 사용하여 테스트 이미지와 랜드 마크 간의 지상 진실 대응을 설정했습니다.

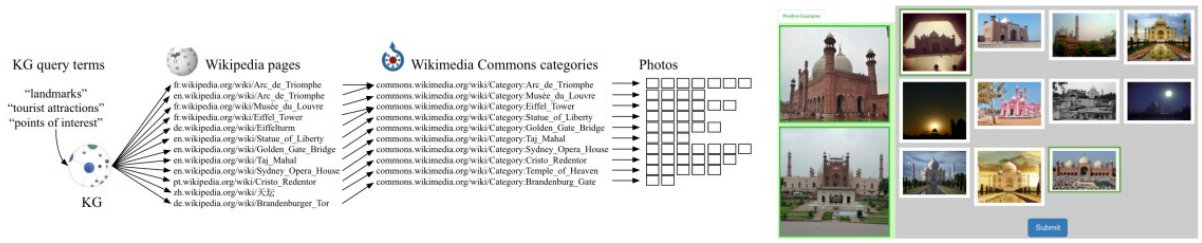


Figure 6: Left: pipeline for mining images from Wikimedia Commons. Right: the user interface of the re-annotation tool.

3.Waymo Open Dataset

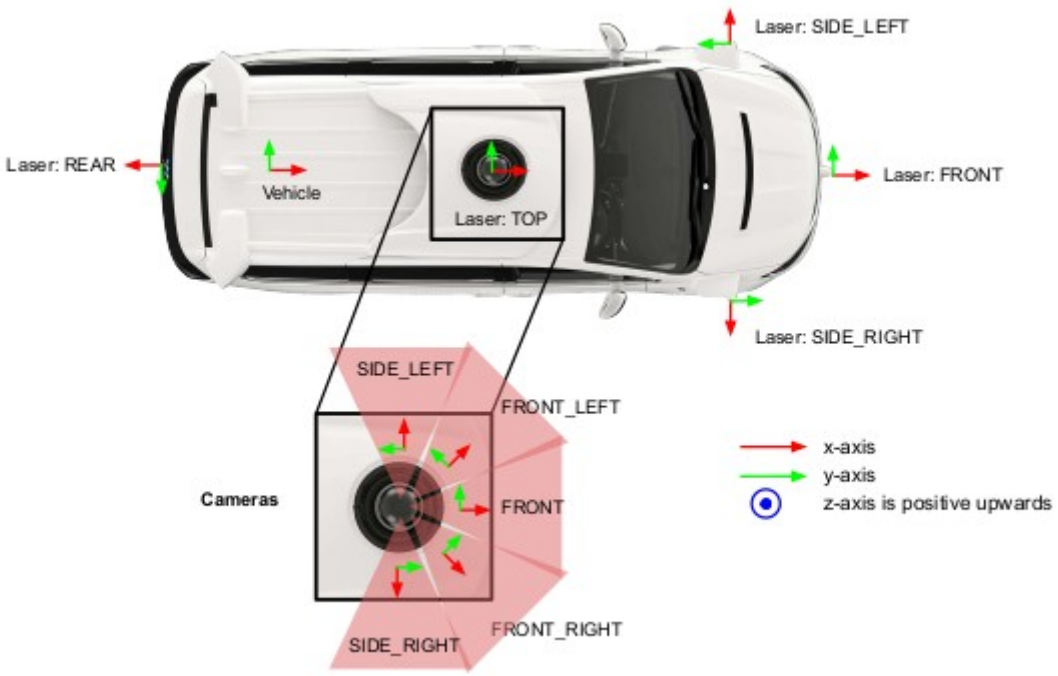
Waymo Open Dataset 은 다양한 조건에서 Waymo Driver 가 운영하는 자율 주행 차량이 수집 한 고해상도 센서 데이터로 구성됩니다.제안된 데이터 세트에는 많은 수의 상위 항목이 포함되어 있다. 품질, 수동으로 주석을 단 3D 접지 실측 경계 상자 LiDAR 데이터 및 2D 고정 경계 상자용 카메라 이미지용입니다.모든 접지 진실 상자에는 트랙이 포함됩니다.데이터 세트에는 약 1200 만 LiDAR 가 포함되어 있습니다.상자 주석 및 약 1200 만 개의 카메라 상자 주석 약 113k LiDAR 개체 트랙을 생성하고 약 250k 카메라 이미지 트랙.Waymo 는 데이터 셋의 모든 센서 데이터를여러 개의 고전압으로 구성된 산업 강도 센서 제품군 해상도 카메라 및 다중 고품질 LiDAR 센서로 찍었습니다.

센서 사양

데이터 수집은 5 개의 LiDAR 센서와 5 개의 고해상도 핀홀 카메라를 사용하여 수행되었습니다.LiDAR 데이터의 범위를 제한하고 각 레이저 펄스의 처음 두 반환에 대한 데이터를 제공합니다.

	TOP	F,SL,SR,R
VFOV	[-17.6°, +2.4°]	[-90°, 30°]
Range (restricted)	75 meters	20 meters
Returns/shot	2	2

카메라 이미지는 롤링 셔터 스캔으로 캡처되며, 여기서 정확한 스캔 모드는 장면마다 다를 수 있습니다. 모든 카메라 영상이 다운샘플링되고 원시 영상에서 잘라집니다.



작업

1.객체 탐지

주어진 프레임의 경우, 3D 감지 작업에는 차량, 보행자, 표지판 및 자전거 이용자를 위한 3D 수직 상자 예측이 포함된다.탐지 방법은 LiDAR 및 카메라 센서의 데이터를 사용할 수 있으며, 이전 프레임의 센서 입력을 활용할 수도 있습니다.3D 감지 작업과 달리 2D 카메라 이미지 감지 작업은 입력 데이터를 LiDAR 데이터를 제외한 카메라 이미지로 제한합니다.

2.객체 추적

다중 개체 추적은 시간이 지남에 따라 장면에서 개체의 ID, 위치 및 선택적으로 속성을 정확하게 추적합니다.Waymo 의 데이터 세트는 10Hz 로 샘플링된 데이터를 생성하는 여러 센서와 각각 20 초 길이의 시퀀스로 구성된다.또한 데이터 세트의 모든 개체에는 각 시퀀스에 걸쳐 일관된 고유한 식별자가 주석이 달려 있다.

2.Open sources for Autonomous Driving

1.AirSim

에어심(AirSim)은 언리얼 엔진을 기반으로 제작된 드론, 자동차 등을 위한 시뮬레이터이다.오픈 소스, 크로스 플랫폼이며, 물리적이고 시각적으로 현실적인 시뮬레이션을 위해 PX4 와 아두 파일럿과 같은 인기 있는 비행 컨트롤러와 PX4 와 같은 하드웨어 인 루프(hardware-in-loop)를 사용하는 소프트웨어 인더루프 시뮬레이션을 지원한다.the-loop) 시뮬레이션을 지원한다.

AirSim 의 목표는 AI 연구를 위한 플랫폼으로 에어심(AirSim)을 개발하여 자율주행차에 대한 딥러닝, 컴퓨터 비전, 강화학습 알고리즘을 실험하는 것이다. 이를 위해 AirSim 은 API 를 공개하여 플랫폼 독립적인 방식으로 데이터를 검색하고 차량을 제어한다.

사용 방법

수동 드라이브

아래와 같이 리모컨(RC)이 있는 경우 시뮬레이터에서 드론을 수동으로 제어할 수 있습니다. 자동차의 경우 화살표 키를 사용하여 수동으로 주행할 수 있습니다.

교육 데이터 수집

딥 러닝을 위해 AirSim 에서 교육 데이터를 생성할 수 있는 방법은 두 가지가 있습니다. 가장 쉬운 방법은 오른쪽 아래 모서리에 있는 레코드 버튼을 누르기만 하면 됩니다. 각 프레임에 대한 포즈 및 이미지 쓰기가 시작됩니다. 데이터 로깅 코드는 매우 간단하며 마음껏 수정할 수 있습니다.

코드 구성

코드의 대부분은 AirLib 에 있습니다. 이 라이브러리는 C++11 컴파일러로 컴파일할 수 있는 자체 포함 라이브러리입니다.AirLib 은 다음 구성 요소로 이루어집니다.

1. 물리학 엔진: 이것은 헤더 전용 물리 엔진입니다.다양한 차량을 구현하기 위해 빠르고 확장 가능하도록 설계되었습니다.

2. 센서 모델: 이것은 기압계, IMU, GPS 및 자기계용 헤더 전용 모델입니다.

3. 차량 모델: 이 모델은 차량 구성 및 모델에 대한 헤더 전용 모델입니다. 현재 X 구성에서 MultiRotor 와 PX4 QuadRotor 구성을 구현했습니다.

4. 제어 라이브러리: AirLib 의 이 부분은 API 를 위한 추상적인 베이스 클래스와 MavLink 와 같은 특정 차량 플랫폼을 위한 구체적인 구현을 제공한다. 또한 RPC 클라이언트 및 서버에 대한 클래스도 있습니다.

Unreal/Plugins/AirSim

이것은 언리얼 엔진에 의존하는 프로젝트의 유일한 부분입니다. 우리는 다른 플랫폼(예: Unity)을 위한 시뮬레이터를 구현할 수 있도록 분리해 두었다. Unreal 코드는 Blueprint 를 포함한 UO 객체 기반 클래스를 활용합니다. SimMode_ 클래스: 우리는 드론이 없는 순수한 컴퓨터 비전 모드와 같은 다양한 시뮬레이터 모드를 지원하고자 합니다. SimMode 클래스는 다양한 모드를 구현하는 데 도움이 됩니다.2 Vehicle PawnBase: 이 클래스는 모든 차량 전당포 시각화의 기본 클래스입니다. 차량 기준: 이 클래스는 렌더링 구성 요소(즉, 언리얼 폰), 물리 구성 요소(즉, MultiRotor) 및 컨트롤러(즉, MavLink)의 조합을 구현하기 위한 추상 인터페이스를 제공합니다.도우미).

2.Gym-Duckietown

Gym-Duckietown 은 Python/OpenGL(Pyglet)으로 쓰여진 Ducietown Universe 의 시뮬레이터이다. 그것은 여러분의 에이전트인 덕키봇을 덕키타운의 한 예 안에 위치시킵니다: 회전하는 도로, 교차로, 장애물, 덕키 보행자와 다른 덕키봇이 있는 순환 도로입니다.

Gym-Duckie 타운은 빠르고, 개방적이며, 사용자 정의가 가능합니다. 차선 추적 시뮬레이터에서 시작된 것은 기계 학습, 강화 학습, 모방 학습 또는 심지어 고전적인 로봇 알고리듬까지 훈련하고 테스트하는 데 사용할 수 있는 완전한 기능을 갖춘 자율 주행 시뮬레이터로 진화했다.Gym-Duckiown 은 단순한 차선 추적에서 동적 장애물이 있는 전체 도시 내비게이션에 이르기까지 광범위한 작업을 제공한다. 또한 Gym-Duckiown 은 도메인 무작위화, 정확한 카메라 왜곡, 차동 물리학을 포함하여 알고리즘을 실제 로봇으로 가져올 수 있는 기능, 포장지 및 도구를 제공합니다.

등록된 체육관 환경이 여러 개 있으며, 각각 다른 지도 파일에 해당됩니다.

`Duckietown-straight_road-v0`

- `Duckietown-4way-v0`
- `Duckietown-udem1-v0`
- `Duckietown-small_loop-v0`
- `Duckietown-small_loop_cw-v0`
- `Duckietown-zigzag_dists-v0`
- `Duckietown-loop_obstacles-v0` (static obstacles in the road)

- `Duckietown-loop_pedestrians-v0` (moving obstacles in the road)

MultiMap-v0 환경은 기본적으로 사용 가능한 모든 맵 파일을 자동으로 순환하는 시뮬레이터용 래퍼입니다.이를 통해 다양한 시나리오에 대한 교육이 보다 강력한 정책/모델을 만들 것이라는 생각으로 다양한 맵에서 동시에 훈련할 수 있다.Gym-duckietown 은 실제 Dukiebots 에 동반되는 시뮬레이터로, 실제 로봇에서 코드를 실행할 수 있게 해줍니다.NAT 은 훈련된 정책을 시뮬레이션에서 실제 환경으로 전송하는 데 도움이 되는 도메인 임의화 API 를 제공합니다. 도메인 전송 방법을 사용하지 않으면 학습된 모델이 시뮬레이터의 다양한 측면에 오버핏되어 실제 세계로 전송되지 않을 수 있습니다.

Gym-Duckietown 실행

1.설치 요구 사항:

Python 3.6+

- OpenAI gym
- NumPy
- Pyglet
- PyYAML
- cloudpickle
- PyTorch

Pip3 을 사용하여 PyTorch 를 제외한 모든 종속성을 설치할 수 있습니다.

```
git clone https://github.com/duckietown/gym-duckietown.git
cd gym-duckietown
pip3 install -e .
```

Conda 를 사용한 설치(대체 방법)

```
git clone https://github.com/duckietown/gym-duckietown.git
cd gym-duckietown
conda env create -f environment.yaml
```

Docker Image

Docker Hub 에는 PyTorch 가 설치되어 있는 사전 제작된 Docker 이미지가 있습니다.시작하려면 Docker Hub 에서 Ducker Down/Gym-ducketown 이미지를 꺼내고 용기에서 셸을 엽니다.

```
nvidia-docker pull duckietown/gym-duckietown && \
nvidia-docker run -it duckietown/gym-duckietown bash
```

그런 다음 가상 디스플레이를 생성합니다.

```
Xvfb :0 -screen 0 1024x768x24 -ac +extension GLX +render -noreset &> xvfb.log &
export DISPLAY=:0
```

이제 RL 을 사용하여 정책 교육을 시작할 준비가 되었습니다.

```
python3 pytorch_rl/main.py \
        --algo a2c \
        --env-name Duckietown-loop_obstacles-v0 \
        --lr 0.0002 \
        --max-grad-norm 0.5 \
        --no-vis \
        --num-steps 20
```

테스트

시뮬레이션이나 실제 로봇을 수동으로 제어할 수 있는 간단한 UI 응용 프로그램이 있습니다.시뮬레이션이나 실제 로봇을 수동으로 제어할 수 있는 간단한 UI 응용 프로그램이 있습니다. manual_control.py 애플리케이션은 Gym 환경을 시작하고 카메라 이미지를 표시하고 작업(키보드 명령)을 시뮬레이터 또는 로봇으로 다시 전송합니다. --map-name 인수를 사용하여 로드할 맵 파일을 지정할 수 있습니다.

```
./
```

강화 학습

강화 학습 에이전트를 교육하려면 /pytorch_rl 에 제공된 코드를 사용할 수 있습니다. A2C 또는 ACKTR 알고리즘을 사용하는 것이 좋습니다. 교육을 시작하기 위한 샘플 명령은 다음과 같습니다.

```
python3 pytorch_rl/main.py --no-vis --env-name Duckietown-small_loop-v0 --algo a2c --lr 0.0002 --max-
```

그런 다음 교육 결과를 시각화하기 위해 다음 명령을 실행할 수 있습니다. 교육 프로세스가 실행 중인 동안 이 작업을 수행할 수 있습니다. 또한 SSH 를 통해 이 기능을 실행하는 경우 디스플레이를 얻으려면 X 전달을 활성화해야 합니다.

```
python3 pytorch_rl/enjoy.py --env-name Duckietown-small_loop-v0 --num-stack 1 --load-dir trained_mode
```

모방 학습

실험 디렉토리에는 합성 데모 데이터 세트를 자동으로 생성하는 스크립트가 있다. 힐 클라이밍을 사용하여 획득한 보상을 최적화하고 JSON 파일을 출력합니다.

```
experiments/gen_demos.py --map-name loop_obstacles
```

그런 다음 다음을 통해 모방 학습 모델(인터넷) 교육을 시작할 수 있습니다.

```
experiments/train_imitation.py --map-name loop_obstacles
```

마지막으로, 학습된 모델이 수행하는 작업을 시각화할 수 있습니다.

```
experiments/control_imitation.py --map-name loop_obstacles
```