PilotNet을 이용한 모형 차량(HENES-T870)의 자율 주행 시뮬레이션

유승준○ 유상곤 황성수
한동대학교 전산전자공학부
21600447@handong.ac.kr, 21700457@handong.ac.kr, sshwang@handong.edu

Autonomous Driving Simulation of Model Car(HENES-T870) Using PilotNet

Seungjun Yu○ SangGon Yu SungSoo Hwang
Computer Science and Electrical Engineering, Handong Global Unversity

요 약

End-To-End deep learning은 주행 과정 자체를 학습을 함으로써 새로운 주행 시나리오에 대한 추가적인 알고리즘의 구현 없이 해당 시나리오에 대한 주행 데이터셋을 학습시키면 자율 주행이 가능하다는 장점을 가지고 있다. 이에 본 프로젝트에서는 End-To-End deep learning을 사용하는 PilotNet[1]을 이용하여서 모형 자동차가 자율적으로 특정 장소(400m 달리기 트랙)를 주행하는 실험을 진행하였다. 학습에 필요한 dataset은 PC로 차량을 직접 주행하며 획득하였고, 획득한 dataset의 일부는 학습 후 시뮬레이션에 사용될 test_set으로 사용하였다. 본 논문에서는 PilotNet의 학습과정과 학습된 PilotNet의 시뮬레이션까지의 내용을 다루고 있다.

1. 서 론

PilotNet[1]은 End-To-End deep learning 기반의 CNN 단일 네트워크이다. End-To-End deep learning은 기존의 deep learning 방식과는 다르게 입력에서 출력까지의 과정을 파이프라인 네트워크 없이 한 번에 처리한다는 것을 의미한다. PilotNet은 학습을 위해서 두 가지의 입력 정보(차량 주행 중의 정면 영상과 주행 중의 차량 핸들 각도 값)만을 사용한다.

즉, 이론적으로 PilotNet은 영상 촬영 및 학습의 과정만으로 자율 주행이 가능한 네트워크임을 의미하며, 영상만으로 자율 주행이 가능한 SLAM과 비교하였을 때, 연산에 요구되는 메모리의 양이 적기 때문에, 여러 알고리즘을 동시에 사용하는 자율 주행 프로그램의 특성상, 동일한 메모리의 조건 안에서 PilotNet이 SLAM보다 다른 알고리즘을 처리할 여유가 있다는 장점을 가지게 된다. 따라서 차량의 메인보드의 가격을 절감하거나, 다른 알고리즘을 추가하여 프로그램의 성능을 높일 발판의 역할을 할 것을 기대해본다.

본 연구에서는 모형 차량을 이용하여 PilotNet이 실제로 자율 주행을 구현 할 수 있는지 알아보기 위해PilotNet을 학습시키고 시뮬레이션을 진행하였다. 연구 과정은 '하드웨어 세팅-데이터셋 생성-PilotNet 학습-시뮬레이션 진행'의 순서로 진행되었다.

2. 본문

2.1 하드웨어 구성 및 소프트웨어

본 연구에서 모형 자동차로는 HENES사의 유아용 전동 차량인 HENES-T870을 사용한다. 크기로는 전장 1,400mm, 전폭 780mm, 전고 530mm인 소형 유아용 전동 차량이다. 영상을 얻기 위해 사용한 카메라 장비는 어안렌즈(ELP-USB8MP02G-L180)를 사용하였다. 사용한 어안렌즈의 가로시야각은 180도이며, 최대해상도는 3264(가로)*2448(세로)이다. 카메라는 차량의 전면부에 430mm 높이에 부착되어 있다.

그림1. 실험용 모형 차량과 어안렌즈의 사진

연구에서 차량과 PC가 통신하기 위해서 하드웨어가 추가적으로 필요하다. 실험에서는 nucleo board(NUCLEO-F130RB)를 사용하여 차량을 제어하는 메인보드와 PC 사이의 Serial 통신을 사용하여 두 개체를 연결하였다. PC에서 신호를 보내면 nucleo board에서 차량에 맞는 신호로 변환하는 과정이 필요한데, 이 과정은 nucleo board의 MBED OS에서 사용되는 MBED 코드를 이용하여 수행해주었다. MBED 코드를 통하여 변환된 신호는 차량으로 보내져 주행 명령을 수행하는 방식이다.

[1]※ This work was supported by the National Program for Excellence in Software at Handong Global University (2017-0-00130) funded by the Ministry of Science and ICT.

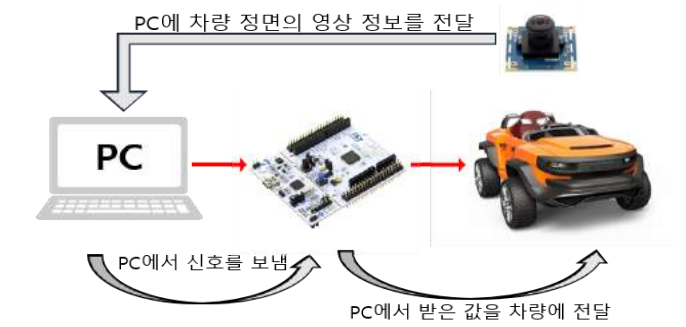

그림2. 차량과 PC의 단방향 연결

다음으로 차량과 PC와의 통신 및 학습을 위하여 사용한 소프트웨어에 대해서도 간략히 설명하겠다. 먼저 키보드 주행을 위한 코드로는 C++언어를 사용하였고 PC와 nucleo board사이의 Serial 통신을 열어주는 코드, keyboard값을 실시간으로 사용자에게 받아 Serial 통신을 통해 board로 넘겨주는 코드, keyboard값을 angle값으로 변환하여 저장하는 코드, 전면 카메라의 image를 저장해주는 코드들로 구성되어있다. 학습 코드와 simulation 코드로는 python 언어로서, Nvidia에서 제공해주는 PilotNet 학습 코드 및 시뮬레이션 코드를 사용하였다. mbed code는 C++언어를 사용하였고, Serial 통신을 통해 넘어온 keyboard값을 HENES-T780차량에 맞는 주행 명령값으로 변경 후, 차량으로 그 값을 넘기는 코드로 구성되어 있다

2.2 학습의 진행

2.2.1 데이터셋 획득 방법

PilotNet 네트워크의 학습을 하기 위한 입력 정보는 주행 시 차량의 정면에 설치된 카메라의 프레임 이미지와 해당 프레임에 매칭되는 주행 각도이다. 여기서, 주행 각도는 헤네스 차량의 앞바퀴 각도이고 정면을 0도로 하여 오른쪽은 양수(+) 각도값을 왼쪽은 음수(-) 각도값을 가진다.

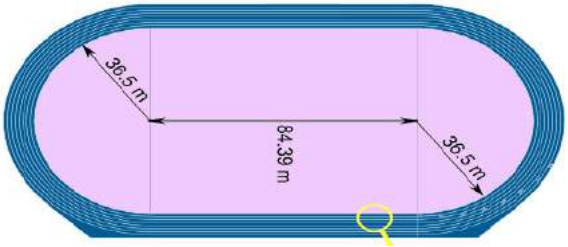

그림3. 400m 규격 트랙

학습을 진행하는 장소는 그림3과 동일한 규격을 가지고 있는 한동대학교 400m 달리기 트랙으로 지정하였다. 일자트랙이 아닌 곡선 트랙이고 라인을 따라 주행하는 것이 아닌 전체 트랙을 하나의 라인으로 간주하고 주행을 실시하였다. 학습에 필요한 주행 데이터셋을 만들 때에는 PC와 간의 Serial 통신을 활용하여 주행을 실시하였다. PC의 키보드 값을 nucleo board의 MBED OS를 통하여 헤네스 차량의 주행 명령값으로 변경 후 차량으로 전달하여 주행을 한다.

2.2.2 데이터셋 구성

End-To-End deep learning은 학습시킨 데이터셋에 대한 주행에 강한 장점을 보이는 만큼 학습시킬 데이터셋의 구성 또한 연구에서 중요하게 다루어야 할 주제이다.

End-To-End deep learning은 데이터셋에 기록된 주행 영상의 과정을 학습하기 때문에, 커브 구간의 비중이 적은 달리기 트랙에서 데이터셋을 생성할 때, PilotNet이 충분한 양의 커브 구간을 학습할 수 있도록 데이터셋을 구성해야 했다.

그래서 데이터셋을 전체 트랙을 주행하는 영상을 기록한 track_set, 트랙의 커브구간만 주행하는 영상을 기록한 curve_set의 구성으로 생성하였다. track_set의 경우 10개 존재하며, 각 데이터셋 당 트랙을 1바퀴 주행하며 기록을 하였으며, 총 약 4만 프레임의 이미지 파일을 저장하고 있다. curve_set의 경우 6개 존재하며, 각 데이터셋 당 커브 구간을 1번 주행하며 기록을 하였고, 약 1만 5천 프레임의 이미지 파일을 저장하고 있다. 학습은 위 track_set과 curve_set을 무작위 배열하여 생성한 약 5만 5천 프레임의 train_set으로 진행하였으며, 학습 이후 모델의 시뮬레이션에 사용될 test_set은 track_set에서 연속된 4500개의 프레임을 무작위로 선별하여 사용하였다.

2.2.3 PilotNet 학습

PilotNet은 총 9개의 레이어로 이루어진 단일 CNN 네트워크이며, 자세한 구조와 학습의 로직은 "*End-to-end learning for lane keeping of self-driving cars*"[1]의 내용을 따르고 있다.

PilotNet 네트워크에 주행 데이터셋을 학습하기 위해서는 데이터셋 내부의 정보를 전처리 하는 과정이 필요하다. 입력정보로 들어가는 프레임의 크기가 320*160이어야 한다, 하지만 데이터셋을 만들 때 생성되는 프레임의 크기는 원본 프레임인 640*480에서 ROI 영역으로서 지정한 640*240이기 때문에 사이즈 변경해 주어야 한다. 그 후, 주행 명령값이 저장된 txt 파일을 프레임이 해당된 경로와 그 프레임의 해당된 주행 명령값이 입력된 csv 파일로 변환 해주어야 한다. 이 때의 데이터셋에 기록된 주행 명령값의 범위는 -20 ~ +20 이지만, PilotNet에서 받아들이는 주행 명령값의 범위는 -1에서 +1사이의 값이어야 하기 때문에 정규화 과정이 진행되어야 한다.

데이터셋에 구성에 맞게 데이터셋들을 획득 후, input data를 PilotNet에 맞게 변환 해 준 뒤 학습을 진행하였다. 이때 epoch 수는 100으로 지정하고, batch size는 12 로 지정하였다. 학습 후 나온 validation loss, training loss값의 그래프는 다음과 같다.

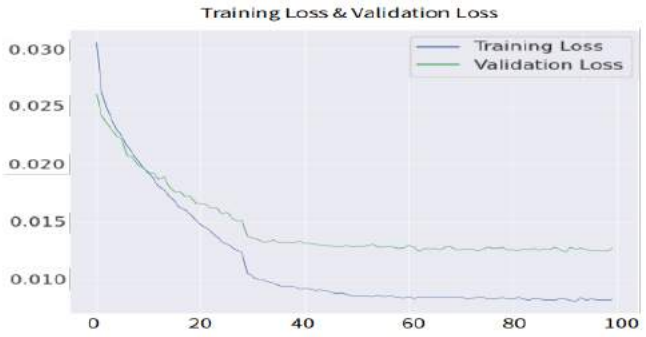

그림4. Validation Loss & Training Loss Graph

그림 4 그래프의 x축은 epoch수, y축은 training loss, validation loss 값을 나타낸다. 이 그래프를 보면 loss값들이 학습이 진행 될 수록 각자 특정 값에 수렴하고 있음을 확인할 수 있다.

3. PilotNet 시뮬레이션 결과

학습이 완료된 PilotNet의 모델에 대한 시뮬레이션 진행은 딥 러닝 손실 함수인 MSE(평균제곱오차)를 통하여 이루어졌다.

$$\mathrm{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

시뮬레이션은 2.2.2에서 언급한 것처럼 track_set에서 test_set을 선별하여 학습된 PilotNet이 산출하는 값과 test_set의 원래 값(Ground Truth)를 비교하여 MSE 값을 확인하는 것으로 진행하였으며, PilotNet의 학습을 진행하는 코드에서 test mode의 명령어를 통하여 진행할 수 있다. 시뮬레이션이 완료가 되면 matplotlib으로 test_set의 원래 값과 학습된 PilotNet이 산출한 값을 비교할 수 있게 보여준다.

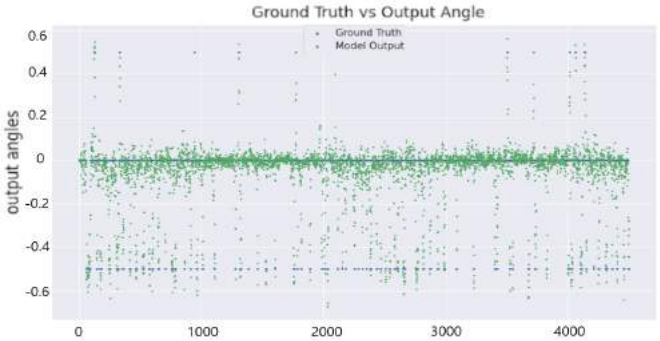

그림5. 실제 값과 모델 예측값(양자화 전)

그림 5를 보면 x축은 프레임의 숫자를 의미하고 y축은 angle값을 나타낸다. 또한 초록점은 모델이 예측한 anlge값이고 파란점은 실제 angle값이다. 그림 5에서 알 수 있듯이 예측값이 실제값에 집중적으로 분포해 있는 것을 알 수 있다. 실제 값들은 주행 각도값을 -1에서 1 사이의 값으로 정규화 과정을 진행한 값이고 주행 각도값이 -20, -10, 0, 10, 20인 만큼 정규화한 값도 -1, -0.5, 0, 0.5, 1의 값만을 가진다. 그에 비해 모델 예측 값들은 -1과 1 사이의 여러 실수 값들을 가지고 이 값을 바로 주행 명령값으로 쓸 수 없기 때문에 정규화한 값을 가지도록 분류하는 양자화 과정을 거쳐야 한다.

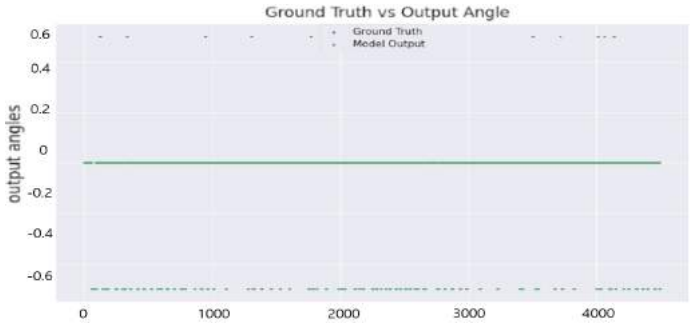

그림6. 실제 값과 모델 예측값(양자화 후)

양자화 과정을 거치게 된다면 그림 6(그림 5와 같은 형식)처럼 모델 예측 값이 실제값처럼 -1, -0.5, 0, 0.5, 1의 값들을 가지게 된다.

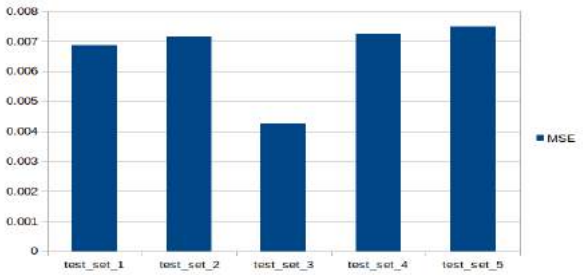

그림7. 시뮬레이션의 MSE 결과

그림7의 x축은 시뮬레이션 번호이고, y축은 각 시뮬레이션에 해당하는 MSE 값을 나타낸다. 시뮬레이션은 총 5번을 진행하였으며, 각 MSE 결과는 그림6에서 확인할 수 있듯이, test_set_3의 결과를 제외하고는 각 값이 비슷한 수준의 결과를 보이고 있다.

4. 결 론

본 논문에서는 End-To-End deep learning 기반의 네트워크인 PilotNet을 활용하여 모형 차량의 자율주행을 구현하려 하였다. 또한, PilotNet의 학습을 진행할 때, 달리기 트랙의 커브 구간에 대한 취약점을 보완하기 위해 데이터셋의 커브 영상 비중을 높여 학습을 진행하였다. 실제 학습의 결과에서 epoch 수가 증가함에 따라 training loss, validation loss값이 줄어들며 특정 값에 수렴하고 있음을 확인할 수 있었다.

또한, 시뮬레이션을 진행하였을 때에도, 전체적인 모델의 예측값들의 흐름이 실제값들의 흐름과 유사한 것을 볼 수 있다. 이러한 결과를 기반으로 실제 주행에서 높은 정확도를 가지고 자율 주행을 수행할 수 있을 것으로 기대된다.

5. 참고 문헌

[1] Bojarski, M., Del Testa, D., Dworakowski, D., Firner, B., Flepp, B., Goyal, P., Jackel, L. D., Monfort, M., Muller, U., Zhang, J., Zhang, X., Zhao, J., & Zieba, K. (2016, April 25). *End to end learning for self-driving cars*. arXiv.org. Retrieved June 7, 2022, from https://arxiv.org/abs/1604.07316