Transactions of KSAE, Vol. 28, No. 6, pp.427-437 (June, 2020)

pISSN 1225-6382 / eISSN 2234-0149
DOI http://dx.doi.org/10.7467/KSAE.2020.28.6.427

가상 3D 라이다 기반 객체 분류 딥러닝 학습 데이터셋 구축 방법에 관한 연구

장 형 준[1)] · 손 원 일[1)] · 안 태 원[1)] · 이 용 기[1)] · 박 기 홍[*2)]

국민대학교 자동차공학전문대학원[1)] · 국민대학교 자동차공학과/자동차공학전문대학원[2)]

A Study on Methods for Constructing Deep Learning Training Datasets for Object Classification Using Virtual 3D Lidar Sensor

Hyeongjun Jang[1)] · Weonil Son[1)] · Taewon Ahn[1)] · Yongki Lee[1)] · Kihong Park[*2)]

[1)]*Graduate School of Automotive Engineering, Kookmin University, Seoul 02707, Korea*
[2)]*Department of Automotive Engineering, Graduate School of Automotive Engineering, Kookmin University, Seoul 02707, Korea*
(Received 2 April 2020 / Revised 9 May 2020 / Accepted 12 May 2020)

Abstract : Deep learning algorithms are widely adopted in autonomous driving due to their strong capabilities of classifying the objects around the vehicle. This paper introduces methodologies for constructing datasets for training deep learning algorithms that receive LiDAR sensor data. The training datasets were built in a virtual environment using 3D LiDAR sensor models and different object models. These datasets were used to train the simple CNN model developed in this study. The performance of the trained CNN model was evaluated using the Waymo open datasets and driving scenarios that include multiple moving objects interfering with one another. The CNN model proved to be as good as the best benchmark models, which in turn assured the validity of the training datasets in this study. The proposed method can achieve significant time and cost savings in the generation of proper training datasets for deep learning algorithms in a variety of autonomous driving complexities.

Key words : Autonomous drivng(자율주행), Lidar sensor(라이다 센서), Deep learning(딥러닝), Training dataset(학습 데이터셋), Object classification(객체 분류)

1. 서 론

자율주행 자동차에는 카메라, 레이더, 라이다, 초음파 등 주변 환경을 인지하는 여러 가지 환경 센서가 장착되어 있다. 이러한 환경 센서를 기반으로 주변 환경의 위험도 높은 장애물들을 검출하고, 우선순위를 선정하여 해당 장애물과의 충돌을 방지하는 시스템들이 활발히 개발중이다.[1,2)] 특히 차량, 보행자, 자전거, 오토바이, 트럭 등 여러 가지 동적 객체의 종류를 구분하여 식별하는 것은 위험도 분석 및 우선 순위 결정에 있어서 가장 중요한 부분 중 하나이다.

최근 컴퓨팅 파워가 높아지고 GPU(Graphics Processing Unit)를 활용한 병렬처리 기법이 확대됨에 따라 기계학습 및 딥러닝 기술이 급속히 발전되고 있으며, 자율주행 자동차 분야에서는 빅데이터를 다뤄야 하는 카메라 센서의 영상 처리에 활발히 적용되고 있다. 특히 딥러닝 구조 중 하나인 CNN(Convolution Neural Network) 연구가 활발하게 이루어지면서 AlexNet,[3)] VGGNet,[4)] ResidualNet[5)] 등 CNN 구조를 활용한 사진 분류 알고리즘이 개발되고 있다.

하지만, 카메라의 영상 데이터만을 이용하여 객체를 검출하는 데에는 크게 두 가지 문제점이 존재한다. 첫째, 영상 정보는 조도, 음영, 날씨 등 주변 환경의 변화에 민감하고 이는 센서 성능에 큰 영향을 끼치게 된다. 둘째, 카메라 센서를 통해 주변 환경의 3차원 공간 정보가 2차원 정보로 변환되기 때문에, 정확한 3차원 공간 정보를 유추해내기 어렵다. 자율주행 자동차는 실시간으로 객체들과의 거리 정보를 바탕으로 운행하기 때문에, 정확한 거리 정보를 얻을 수 없다면 차량 운행이 불가능한 상황까지 이를 수 있다.

이러한 문제점을 해결하기 위해 3D 라이다 센서 데이터를 활용하여 객체 분류와 객체와의 정확한 거리 정보

*Corresponding author, E-mail: kpark@kookmin.ac.kr

를 얻기 위해 연구가 활발히 진행되고 있다.[6-8] 3D 라이다 센서를 통해 얻어지는 포인트 클라우드는 포인트별로 3차원 공간의 (x, y, z) 좌표 정보를 포함하며, 레이저가 객체에 반사되어 돌아올 때의 세기를 나타내는 반사율 정보를 또한 포함한다. 이러한 데이터를 바탕으로 객체와의 정확한 거리 정보를 얻을 수 있으며, 객체의 형상 정보를 바탕으로 객체 분류 또한 가능하게 된다.

이를 바탕으로 3D 포인트 클라우드 정보를 2차원 평면으로 투영시켜 영상 데이터를 만들고, 카메라 영상처리 기법에 사용되는 분류 알고리즘을 적용시켜 객체를 분류할 수 있다. 혹은 3D 포인트 클라우드 정보를 Voxel 정보로 변환하여 처리하는 기법인 VoxelNet[9]이 개발되었다. Voxel은 3D 포인트 클라우드 정보를 3차원 공간상의 격자 형태의 데이터로 변환시키는 과정을 뜻하며, 격자 크기에 따라 분해능이 결정된다. 그러나 3차원 공간 정보를 2차원으로 변환시키지 않고 딥러닝에 적용할 경우, 데이터 처리 시간이 2차원 데이터를 처리할 때보다 수십 배 이상 많이 소요되게 된다.

딥러닝 학습은 크게 지도학습과 비지도학습으로 나뉘게 된다. 비지도학습은 비슷한 데이터들끼리 군집화하여 미래를 예측하는 방법으로, 지도학습의 적절한 특징을 찾아내기 위해 전처리 방법으로 사용된다. 반면 지도학습의 경우 학습 시 필요한 Ground Truth 정보를 함께 제공하여 학습을 진행하게 된다. 학습 데이터셋은 수집된 데이터의 Ground Truth 정보를 판단하는 작업을 사용자가 직접 수행해야 하기 때문에, 작업 시간이 오래 걸리고 작업 오류 또한 발생하기 쉽다.

본 논문에서는 가상 3D 라이다 환경을 기반으로 포인트 클라우드로부터 객체 정보를 추출하기 위한 딥러닝 학습 데이터셋 구축을 위한 방법에 대해 연구하였다. 실제 라이다 센서의 스펙에 맞춰 설계된 가상 라이다 센서를 차량 모델에 장착하고, 가상의 객체 정보를 스캔한 뒤 자동적으로 객체를 라벨링하도록 하였으며, 타당성 검증을 위해 실제 라이다 센서 기반의 공개된 오픈 데이터셋과의 비교검증을 실시하였다. 타당성 검증 및 실제 딥러닝 모델의 구현을 위하여, CNN 입력 데이터 처리 및 CNN 모델을 구성하였으며, 가상 환경에서 검증하기 위한 시나리오 및 시나리오별 검증 결과를 통해 가상 라이다 센서 기반 데이터셋의 실효성을 검증하였다.

2. 라이다 포인트 클라우드 기반 객체 분류 알고리즘

본 논문에서는 가상 3D 라이다 센서 기반 데이터셋 구축을 위해, Fig. 1과 같은 프로세스를 설정하였다. 첫째, 가상 환경을 구축하고 가상 라이다 센서를 모델링한다. 둘째, 객체별 데이터셋을 취득하기 위해 가상 3D 라이다 센서로 각 객체를 스캔하여 딥러닝 모델의 입력 데이터로 사용될 데이터셋을 구축하고, 해당 데이터셋을 통해 전처리과정을 거쳐 CNN 네트워크 모델을 학습시킨다. 셋째, 가상 라이다 센서 기반 데이터셋을 통해 학습된 CNN 모델이 정상작동을 하는지 확인하기 위하여 이미 공개된 데이터셋을 이용하여 비교 검증을 실시하고, 공개된 벤치마크 결과와 비교 분석을 시행하는 절차를 거친다. 이에 대한 자세한 내용은 3장에서 다루고자 한다.

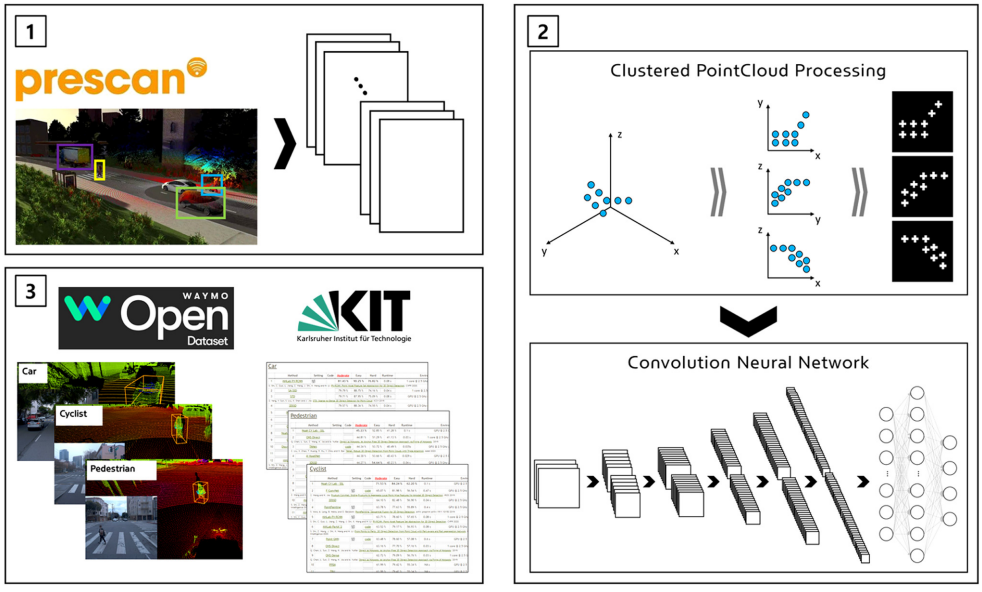


Fig. 1 Construction methods outline for deep learning training dataset outline

본 논문의 CNN 네트워크 모델은 이미지, 비디오, 텍스트, 사운드 데이터의 특징점을 검출하여 분류하는 모델로서 딥러닝에서 많이 사용되는 기법이다. 특히 이미지 패턴을 찾아 분류하는데 용이하며, CNN에 이용되는 각각의 필터는 학습된 가중치를 기반으로 전체 데이터가 아닌 일부분의 데이터만 사용하기 때문에 데이터 변경, 왜곡 등에 따라 분류 성능에 큰 영향을 받지 않아 높은 강인성을 가진다.[10]

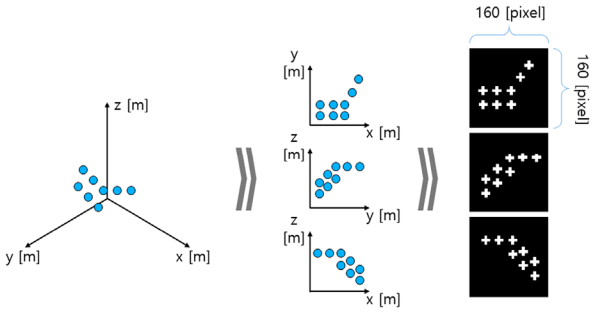


Fig. 2 3D pointcloud plane projection and conversion to grid map

2.1 CNN 입력 데이터 전처리

본 연구에서 CNN 입력 데이터는 Fig. 2와 같이 3D 라이다 센서 데이터에서 검출된 객체 포인트 클라우드를 xy평면, yz평면, xz평면으로 투영시켜 각 좌표 평면별 이미지로 변환된다. 객체 크기의 평준화를 위하여, 8×8×8 m^3 크기의 정사면체 형태 데이터를 (3, 160, 160) 즉 160×160 픽셀 크기의 사진 3장으로 변환시켰다. 입력 데이터 처리 시 각 포인트의 m 단위의 좌표 정보를 Pixel 단위의 좌표 정보로 변환하는 방식을 사용하였으며, 하나의 Pixel 은 8 m를 160 pixel로 나눈 수치인 0.05 m의 분해능을 가지게 된다. 또한, 각 좌표 평면으로 투영된 포인트를 Fig. 2와 같이 사진으로 변환시킬 때, 한 개의 포인트를 한 개의 픽셀로 대조시키지 않고, 그 점을 중심으로 상, 하, 좌, 우 네 방향으로 확장시켜 5개의 픽셀로 대조시킴으로써 많은 데이터를 사용하여 객체 분류 정확도를 높일 수 있도록 하였다.

2.2 CNN 네트워크 구성

본 연구에서 CNN 네트워크 구성은 Fig. 3에서 보는 것과 같이, 한 개의 Input Layer와 다섯 개의 Convolution Layer 그리고 2개의 Fully-Connected Layer로 구성하였다. 입력 데이터의 크기는 (3, 160, 160) 즉 3장의 160×160 픽셀 크기의 사진이며, Convolution Layer를 통해 합성곱 연산 과정[11]을 거쳐 특징값을 추출하여 네트워크 모델을 학습시키게 된다.

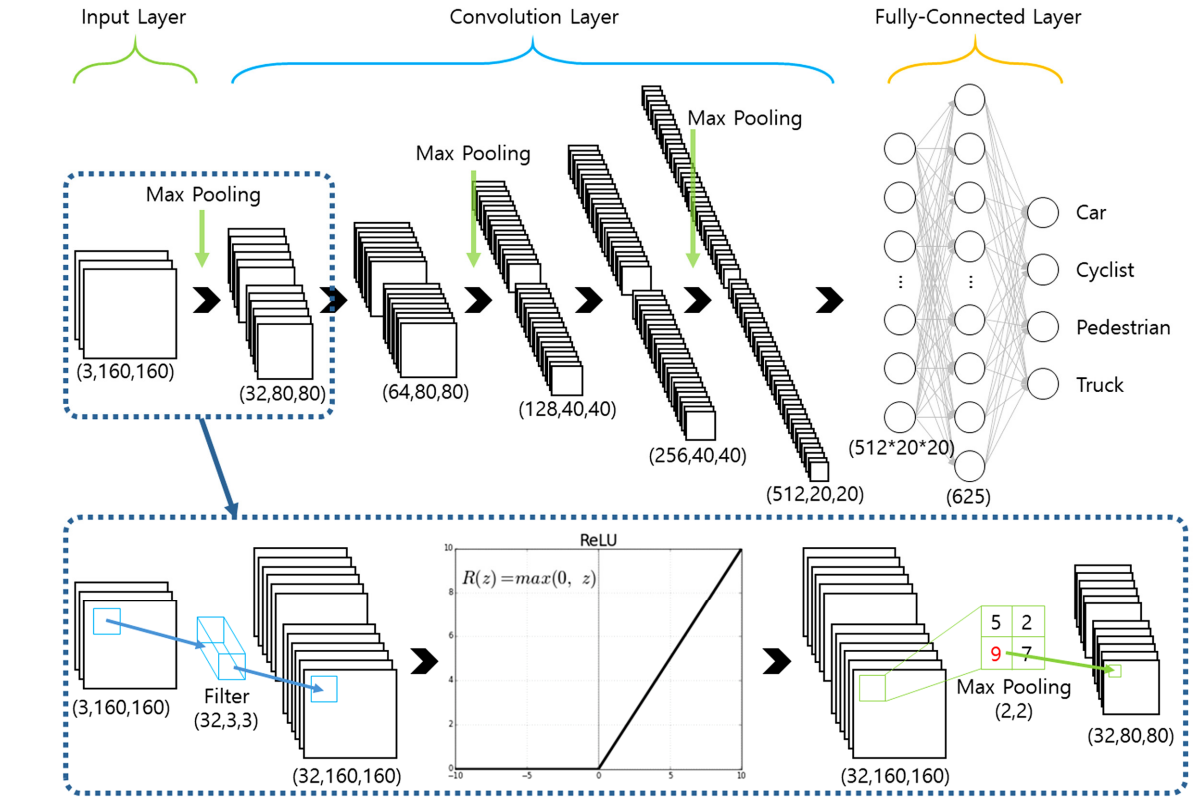


Fig. 3 Convolution neural network

Table 1 Convolution neural network composition

Layer name	Input size	Output size	Activation function
Convolution layer 1	(3, 160, 160)	(32, 160, 160)	ReLU
Max pooling layer 1	(32, 160, 160)	(32, 80, 80)	-
Convolution layer 2	(32, 80, 80)	(64, 80, 80)	ReLU
Convolution layer 3	(64, 80, 80)	(128, 80, 80)	ReLU
Max pooling layer 2	(128, 80, 80)	(128, 40, 40)	-
Convolution layer 4	(128, 40, 40)	(256, 40, 40)	ReLU
Convolution layer 5	(256, 40, 40)	(512, 40, 40)	ReLU
Max pooling layer 3	(512, 40, 40)	(512, 20, 20)	-
Fully-connected layer 1	(1, 512*20*20)	(1, 625)	ReLU
Fully-connected layer 2	(1, 625)	(1, 4)	-

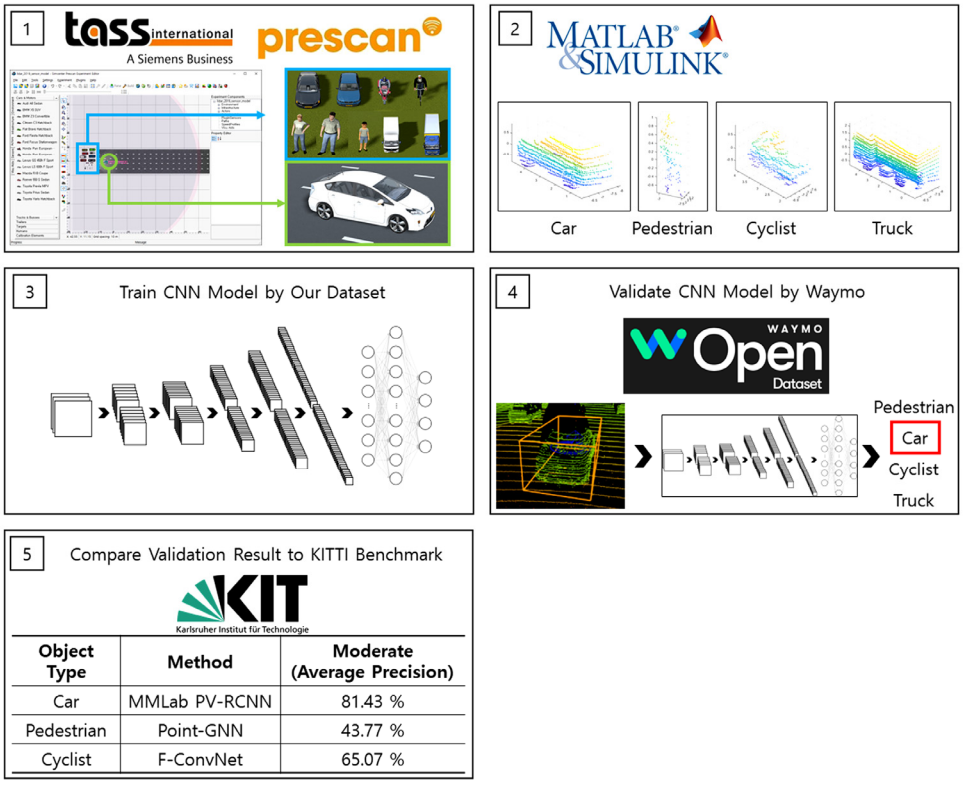
Convolution Layer는 입력 데이터로부터 특징을 추출하는 역할을 수행하며, 특징점 추출을 위한 필터와 필터의 값을 비선형 값으로 바꾸어주는 활성화 함수로 구성되어 있다. 본 논문에서는 활성화 함수로 ReLU 함수를 사용하였다.[12] 활성화 함수는 네트워크 모델의 비선형성을 추가하기 위해 사용되며, ReLU 함수는 입력되는 값이 0보다 작은 경우 0으로 환산하며, 0보다 클 경우 선형함수로 구성되어 있어 구현이 간편하고 계산 효율이 높다는 장점을 가진다.

또한, 모델의 과적합을 방지하기 위해 Pooling Layer는 Max Pooling[13] 방식을 사용하였다. Pooling Layer의 역할은 이미지를 압축시키는 역할을 수행하며 이를 통해 연산 시간을 단축시킬 수 있다. Max Pooling은 Fig. 3과 같이 2 × 2 크기의 행렬의 데이터 중 가장 높은 값을 반환하는 방식을 사용한다.

Table 1은 네트워크의 구성 요소를 나타내며, 각 Layer의 입력 및 출력 크기를 표기하였다.

3. 시뮬레이션 환경 및 검증 결과

Fig. 4는 시뮬레이션 환경 구축 및 자체 데이터셋을 공개된 데이터셋과의 비교 검증을 위한 순서를 나타낸다. 1번과 같이 Siemens사의 PreScan을 사용하여 가상 환경에서 동적 객체 및 센서 모델링을 진행한다. 이후 2번과 같이 Mathworks사의 MATLAB/Slimulink에서 객체별 포인트 클라우드 정보를 수집한 뒤 자체 데이터셋을 구축한다. 이후 3번과 같이 앞서 구성한 CNN 모델을 통해 구축된 자체 데이터셋을 기반으로 학습을 수행한다. 학습된 CNN 모델을 바탕으로 4번과 같이 Waymo사가 제공하는 Open Dataset을 이용하여 CNN 모델을 검증한다. 최종적으로 나온 결과 값을 5번과 같이 KITTI 벤치마크 결과와 비교하였다.



Object Type	Method	Moderate (Average Precision)
Car	MMLab PV-RCNN	81.43 %
Pedestrian	Point-GNN	43.77 %
Cyclist	F-ConvNet	65.07 %

Fig. 4 Flow chart of virtual dataset construction and comparison

3.1 시뮬레이션 환경 구성 및 검증 환경

3.1.1 시뮬레이션 환경 구성

객체 분류 알고리즘 검증을 위해서 Mathworks사의 MATLAB/Slimulink와 Siemens사의 PreScan을 사용하여 가상 환경을 구성하였다.Velodyne사의 Puck[14]과 Puck Hi-Res[15]의 실제 라이다 센서를 스펙에 맞추어 가상 라이다 센서로 모델링하였으며, 이들 센서를 장착할 차량으로는 현대자동차의 소나타 차량을 그 제원을 바탕으로 모델링하였다. Fig. 5는 센서의 장착 위치를 나타낸다. Puck의 경우 차량 루프 정중앙에 장착하였으며, Puck Hi-Res의 경우 전방 범퍼 끝단에 장착하였다.

Fig. 5 Installation of LiDAR sensors on test vehicle

차량과 라이다를 이와 같은 특정한 제품으로 선정한 것은, 본 연구기관에서 실물로 구축 중에 있는 자율주행 자동차의 하드웨어 환경을 그대로 가상 환경에서 모사하기 위함이다. 비록 본 논문의 연구 범위에, 본 연구에서 개발된 알고리즘의 실제 자율주행차량 환경에서의 검증이 포함되지 않았지만, 향후 차량이 완성되면 많은 실도로 주행을 통해 이러한 부분을 진행할 계획이다.

또한, 사용자의 개발 환경에 맞춰 센서를 모델링할 수 있으며, 센서의 장착 위치 및 각도를 임의로 설정이 가능하다는 장점이 있다.

본 연구에서 사용된 두 라이다 센서는, 수직 F.o.V. (Field of View) 각도에 있어서만 차이가 있고 그 외의 스펙은 동일하다. 두 라이다 센서 모두 16 채널의 Layer를 가지며, 수평 F.o.V.는 360 deg, 수평 각도 분해능은 샘플 시간 5 ~ 20 Hz일 때 0.1~0.4 deg이다. 수직 F.o.V.의 경우, Puck이 ±15 deg, Puck Hi-Res이 ±10 deg이며, 두 제품 모두 16채널이고, 수직 각도 분해능은 Puck이 2 deg, Puck Hi-Res이 1.33 deg이다. 따라서, Puck Hi-Res의 경우 Puck보다 객체를 수직으로 좀더 촘촘히 스캔하게 된다. 이를 통해 1초 동안 얻는 포인트의 개수는 라이다 센서 당 288,000 개가 된다. 참고로 이 갯수는 샘플 주기와 무관한데 이는 샘플 주기가 변함에 따라 수평 각도 분해능이 연동되어 변하기 때문이다. 그리고 Puck 및 Puck Hi-Res의 경우 스펙 상 반경 100 m까지 객체를 검출 가능한 것으로 명시되어 있으나 실제 테스트를 통해 60 m 내외에서 대부분의 데이터가 소실되는 것을 알 수 있었다. 이에 따라 두 센서의 모델에서는 최대 검출거리를 65 m로 설정하였다.

본 연구에서는 실제 라이다 센서의 포인트 클라우드 데이터에 존재하는 랜덤한 형태의 거리 노이즈를 반영하기 위해, Fig. 6과 같이 Matlab/Simulink를 이용하여 Gaussian 형태의 노이즈를 생성한 다음 이를 PreScan에서 출력되는 포인트 클라우드 데이터에 주입시키는 과정을 수행하였다. 거리 노이즈의 크기는라이다 센서 스펙과 동일하게 표준 편차 3 cm로 설정하였다.

두 개의 라이다 센서로부터 취득되는 포인트클라우드 데이터는 CNN 모델로 입력되기 이전에 하나로 통합되는 캘리브레이션 과정을 거치게 된다. 두 라이다 센서 중

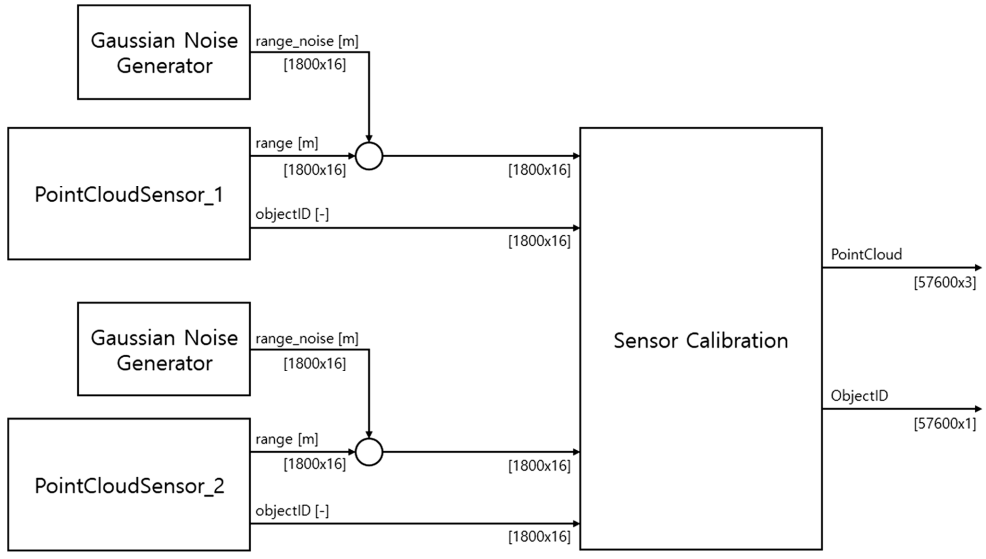


Fig. 6 Virtual LiDAR sensor pointcloud generator

하나를 기준으로 삼고, 두 라이다 센서의 장착 위치에 따른 상대 거리와 상대 각도 정보를 바탕으로 구해지는 이동변환 행렬과 회전변환 행렬을 이용하면, 두 개의 3D 포인트 클라우드 정보를 한 개의 3D 포인트 클라우드 정보로 통합할 수 있다. 본 연구에서는 추후 실차시험을 고려하여 두 개의 라이다 센서만을 이용하였지만, 이러한 통합 과정은 2개보다 더 많은 라이다 센서가 사용되는 경우 또는 해상도가 다른 센서가 사용된 경우에도 동일하게 적용될 수 있다.

라이다 센서 여러 개를 사용하는 것은, 라이다 센서의 해상도가 높아질수록 센서 가격이 급격히 증가하는 문제점을 해결하는 방안이 될 수 있다. 동일한 객체에 대해 보다 세밀한 스캔을 할 수 있으며, 이는 CNN 모델의 학습에 긍정적인 영향을 주며 이는 궁극적으로 CNN 모델의 인식률 향상을 가져다 준다. 특히 본 연구에서는 가상환경에서 데이터셋을 구축함으로써 센서의 갯수와 그들 사이의 상대 위치 및 각도를 매우 쉽게 설정할 수 있다는 장점을 제공한다.

3.1.2 시뮬레이션 환경 구성

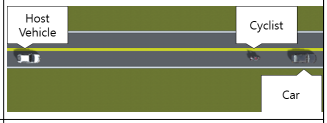
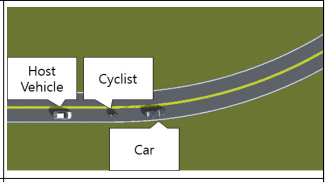
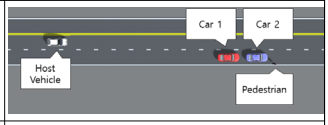
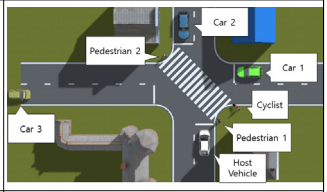
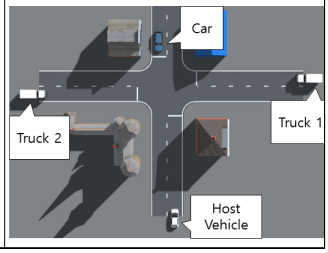
본 연구의 객체 분류 알고리즘을 검증하기 위하여 Table 2에서 보는 것과 같이 총 5가지 시나리오를 구성하였다.

Scenario 1과 Scenario 2는, 직진 및 곡선 주행 상황에서 선행 차량과 선행 오토바이를 구분할 수 있는지 확인하기 위해 구성하였다. 이러한 상황은 일반 주행시 흔히 발생할 수 있는 시나리오이며, 선행 오토바이 앞에 선행 차량이 존재함으로써 오토바이에 의해 가려진 선행 차량의 포인트 클라우드 데이터가 올바르게 분류되는지를 확인한다.

Scenario 3은 EURO N-CAP[16] 시나리오 중 하나로, 차량에 가려져 있던 보행자가 도로를 횡단하기 위해 갑자기 나올 때 보행자 검출이 원활히 이루어지는지를 확인하기 위해 구성하였으며, 포인트 클라우드 데이터가 보행자로 올바르게 분류되는지를 확인한다.

Scenario 4와 Scenario 5는 교차로 상황에서 차량, 보행자, 자전거, 트럭을 구분할 수 있는지 확인하기 위해 구성하였다. 교차로 상황에서는 이러한 객체들이 건물, 신호

Table 2 Test scenario

Scenario number	Road type	Objects	Scenario specification	Scenario image
Scenario 1	Straight road	Car, Cyclist	Host vehicle speed : 20 kph Car speed : 20 kph Cyclist speed : 20 kph	
Scenario 2	Curved road	Car, Cyclist	Host vehicle speed : 20 kph Car speed : 20 kph Cyclist speed : 20 kph Road radius curvature : 100 m	
Scenario 3	Straight road	Car, Pedestrian	Host vehicle speed : 20 kph Car1, Car2 speed : 0 kph Pedestrian speed : 5 kph, Cross road	
Scenario 4	Cross road	Car, Cyclist, Pedestrian	Host vehicle speed : 0 kph Car1, Car2 speed : 0 kph Car3 speed : 18 kph, Turn right Pedestrian1, Pedestrian 2 speed : 3.6 kph, Cross road Cyclist speed : 3.6 kph, Cross road	
Scenario 5	Cross road	Car, Truck	Host vehicle speed : 18 kph Car speed : 0 kph Truck 1 speed : 18 kph, Turn right Truck 2 speed : 18 kph, Turn right	

등과 같은 정적 객체들 속에서 유기적으로 이동하게 되는데 이러한 악의적인 환경에서 정상적으로 객체 분류가 이루어지는지 확인한다.

3.1.3 CNN 학습 데이터셋 추출

CNN 알고리즘은 지도학습으로, 학습 데이터셋 내에 입력 데이터와 그 입력 데이터의 Ground Truth 정보인 라벨 데이터를 함께 입력시켜주어야 한다. 입력 데이터의 경우 수집이 용이한 편이나, 입력 데이터가 어떤 데이터인지를 판단하여 라벨링하는 작업은 대부분 사용자가 수동으로 입력하여야 된다. 사용자가 수동으로 입력을 하게 되면, 작업자 오류가 발생할 수 있고 그에 따라 정확한 입력 데이터 형성이 어려우며 수작업으로 인해 작업 시간이 오래 걸리게 된다.

본 논문에서는 위와 같은 작업자 오류를 최소화시키고, CNN 입력 데이터의 자동 라벨링 방법에 대한 연구를 수행하였다. 본 논문에 사용된 CNN은 크게 차량, 자전거(오토바이), 보행자, 트럭 총 4가지의 객체를 분류하며, 이를 위해 PreScan의 기본적인 Actor 모델을 학습 데이터로 사용하였다. 각 객체는 다시 다음과 같이 세분화하여 총 9개의 객체를 구성하였다. 차량의 경우 세단 차량과 해치백 차량으로 구성하였다. 자전거의 경우 오토바이 및 자전거로 구성하였으며, 보행자의 경우 성인 남성, 성인 여성, 어린이로 구성하였다. 트럭의 경우 1톤 및 2.5톤 탑차 형태의 트럭으로 구성하였다.

각 객체별로 다양한 각도 및 거리를 두기 위해 Fig. 7과 같이 객체를 이동 및 회전시켜 학습 데이터를 수집하였다. 각 객체의 종방향 위치는 최초 위치 5 m로부터 시작해서 5 m 간격으로 65 m까지 총 13개의 위치를 갖도록 구성하였으며, 횡방향 위치는 최초 위치 0 m로부터 시작해서 차선의 표준 폭인 3.6 m 간격으로 ±7.2 m까지 총 5개의 위치를 갖도록 구성하였다. 또한 각 위치에서 객체는 0 deg에서 시작하여 10 deg의 간격으로 회전시킴으로써 총 36개의 회전방향을 갖도록 구성하였다.

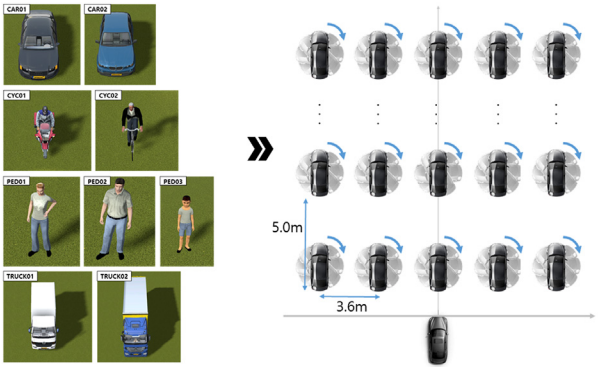


Fig. 7 PreScan actor and extraction object pointcloud method

9개의 객체, 13×5개의 위치, 36개의 회전방향을 조합하여, 총 21,060개의 객체 포인트 클라우드 데이터셋을 구축하였고, 이 중 객체의 크기가 상대적으로 작은 자전거, 보행자의 경우 거리가 멀어 클라우드 포인트 개수가 5개 이하로 떨어지는 경우를 제외함으로써, 최종적으로 총 15,648개의 객체에 대한 학습 데이터셋을 구축하였다. 하지만 이러한 데이터셋으로 학습된 CNN모델을 검증할 때 사용되는 시험 데이터셋에서는 대부분의 객체들이 위와 같은 정형화된 위치와 각도를 벗어나 있게 된다.

Waymo사의 Open Dataset과 같은 경우, 실제 차량과 라이다 센서를 가지고 주행을 하지 않고도 객체 분류 알고리즘을 검증할 수 있는 포인트 클라우드 데이터를 제공한다는 장점을 지닌다. 왜냐하면 실차와 실도로 환경에서 이러한 데이터를 취득하고 Ground Truth 정보를 만드는 작업은 매우 많은 시간과 비용이 소요되기 때문이다. 하지만 사용자가 실제 자율주행차를 구축하는 경우, 차량에 장착된 센서의 종류, 위치, 각도가 오픈 데이터셋 취득시 사용된 센서 환경과 다르기 때문에, 오픈 데이터셋이 제공하는 센서 데이터와 실제 자율주행차에서 얻어지는 데이터가 상이하다는 단점을 지닌다.

본 논문에서는 가상 라이다 센서를 사용하여 이러한 문제점에 대한 해결이 될 수 있는 방법론을 제시하고자 한다. 가상 라이다 센서의 경우 장착 위치 및 각도를 손쉽게 수정할 수 있고 그에 따른 데이터를 얻을 수 있다는 장점이 있다. 또한, 실제 라이다 센서의 경우 각 레이저 빔마다 검출되는 객체의 Ground Truth 정보를 얻을 수 없으나, 가상 라이다 센서의 경우 검출되는 객체에 반사되어 돌아오는 레이저 빔마다 Ground Truth 정보를 얻을 수 있어 학습 데이터의 라벨링 작업을 자동화할 수 있다는 장점을 지닌다.

3.2 검증 결과

3.2.1 CNN 모델 학습 결과

가상 라이다 센서로부터 추출한 학습 데이터셋은 학습 데이터, 검증 데이터, 시험 데이터로 나뉘게 된다. 학습 데이터는 CNN 모델을 학습시키는 데이터이며, 검증 및 시험 데이터는 학습된 CNN 모델의 정확도를 산출하기 위해 사용된다. 데이터의 개수는 학습 데이터 8802개, 검증 데이터 2934개, 시험 데이터 3912개로 이루어져 있다.

학습 데이터와 검증 데이터를 일반적으로 7:3 비율 혹은 9:1로 나누어 교차 검증하는 방식을 Cross-Validation이라고 하며, 이는 CNN 모델의 과적합 방지를 위해 사용된다. 시험 데이터의 수가 천개 단위로 적은 경우 9:1 비율을 사용하나 본 논문에서는 약 1만개의 데이터를 사용

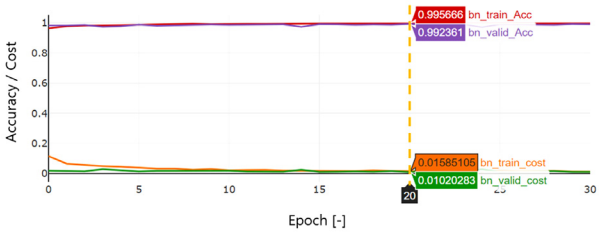


Fig. 8 CNN train-validation accuracy

하므로 7:3 비율로 나누었다.[17] 그리고 시험 데이터의 경우 학습과는 무관하며 CNN 모델의 정확도를 판단하는 척도로 사용되기 때문에 학습 데이터와 분리시켜 관리하도록 하였다.

학습을 위한 프로그래밍 언어는 Python[18]을 사용하였다. 모델을 구성하기 위해 PyTorch 라이브러리[19]를 활용하였으며, 학습률로 0.0003의 값을 사용하였다. 학습률은 매 학습마다 학습 정도를 반영하는 가중치로, 학습률 값이 높을 경우 수렴하지 못하는 상황이 발생할 수 있고, 학습률이 낮을 경우 수렴하는 속도가 매우 느려지는데, 이러한 점을 고려하여 적절한 값으로 선정하였다.

데이터셋을 빠른 속도로 학습시키기 위해 Batch Size를 64개의 크기로 설정하여 전체 데이터셋의 크기를 분할하여 사용하였으며, 매 Epoch마다 데이터셋을 무작위로 섞어서 학습시켰으며, 최종 정확도가 99.5 %를 초과할 때까지 학습을 실행하였다. 그 결과 Fig. 8과 같이 20번의 Epoch만에 학습 정확도 99.5 %, 검증 정확도 99.2 %, 시험 정확도 98.9 %를 얻을 수 있었다. 학습시 사용한 장비로는 AMD 3600X CPU와 Nvidia GTX-1070 GPU가 사용되었으며, 학습 시 소요된 시간은 Epoch마다 대략 1분씩, 20 Epoch 동안 약 20분이 걸렸다.

3.2.2 오픈 데이터셋을 통한 가상 라이다 센서 기반 학습 데이터셋 타당성 검증

본 연구를 통해 도출된 가상 3D 라이다 센서 기반 CNN 학습 데이터셋의 타당성 검증을 위해 최근 공개된 Waymo사의 Open Dataset[20]을 활용하였다. 이 Open Dataset은 약 1 TB 의 크기를 갖는 대용량 데이터로 차량에 장착된 카메라 센서, 라이다 센서 데이터를 10 Hz 단위로 20초 길이의 데이터를 저장하고 있다. 장착된 센서는 1개의 Mid-range 라이다 센서, 4개의 Short-range 라이다 센서, 5개의 카메라 센서(전면, 측면)로 구성되어 있다. 라벨링 데이터의 경우 차량, 보행자, 자전거, 표지판으로 총 4개의 객체 종류를 분류하여 저장되어 있다. 또한, 주행 환경 요인을 고려하여 도심지, 외곽도로, 낮, 밤, 공사 현장, 다양한 날씨 등 여러 환경 조건에서 취득된 데이터로 구성되어 있다.

본 논문에서는 동적 객체만을 고려하여, 차량, 보행자, 자전거, 트럭으로 객체 종류를 분류하였기 때문에, Waymo Dataset과 중첩되는 객체 종류인 차량, 보행자, 자전거를 비교 대상으로 선정하여 타당성 검증을 위한 데이터를 추출하였다. 이후 앞서 도출한 CNN 모델의 입력 데이터로 변환시켜 각 객체별 평균 정확률(Average Precision)을 산출하였다. 사용된 객체의 개수는 차량 203,839개, 보행자 40,948개, 자전거 1,777개로 총 246,564개의 데이터로 구성되어 있으며, 객체별 평균 정확률은 차량의 경우 98.95 %, 보행자의 경우 60.99 %, 자전거의 경우 59.82 %로 도출되었다.

KITTI 벤치마크[21] 결과에 따르면 3D Object 분류 평균 정확률에서 포인트 클라우드 정보를 사용하여 객체 분류를 한 결과, 차량의 경우 MMLab PV-RCNN이 81.43 %, 보행자의 경우 Point-GNN이 43.77 %, 자전거의 경우 F-ConvNet이 65.07 %로 나타났다.

본 논문을 통해 도출된 CNN 모델은 가상 라이다 센서로부터 얻어진 데이터셋을 통해 학습되었으며, Waymo Dataset을 통해 검증한 평균 정확률을 KITTI 벤치마크 결과를 비교하였을 때, 차량의 경우 17.52 % 높으며, 보행자의 경우 17.22 % 높으며, 자전거의 경우 5.25 % 낮은 것으로 확인되었다. 이를 통해 기후와 같은 환경변화가 포함된 Waymo Dataset을 이용하여 CNN 모델을 검증하였을 때 실질적으로 사용하는데 이상이 없는 수준으로 판단되며, 오히려 차량 및 보행자의 경우 KITTI 벤치마크 결과에 비해 더 좋은 성능을 발휘함을 확인할 수 있었다.

3.2.3 검증 시나리오별 결과

앞서 정의한 검증 시나리오에 따라 검출되는 객체 정보를 CNN 모델로 입력하여 객체별 정확률(Precision), 재현율(Recall), 정확도(Accuracy) 및 혼동 행렬(Confusion Matrix)을 도출하였다. 혼동 행렬은 시험용 데이터를 학습 모델에 입력하여 실제 정답과 모델을 통해 예측한 답을 하나의 행렬로 나타내며 이를 통해 모델의 정확률, 재현율, 정확도를 산출할 수 있다.

정확률은 식 (1)과 같이 모델이 참이라고 분류한 것 중에서 실제 참인 것의 비율이며, 재현율은 식 (2)와 같이 실제 참인 것 중에서 모델이 참이라고 예측한 것의 비율이고, 정확도는 식 (3)과 같이 모델이 참이라고 예측한 것 중에서 실제 참인 것과 모델이 거짓이라고 예측한 것 중에서 실제 거짓인 것의 비율을 나타낸다.

Fig. 9는 차량 객체에 대한 혼동 행렬을 나타내며, True Positive, True Negative, False Positive, False Negative의 경우가 혼동 행렬에서 차지하는 영역을 보여준다.

$$Precision = \frac{TP}{TP + FP} \quad (1)$$

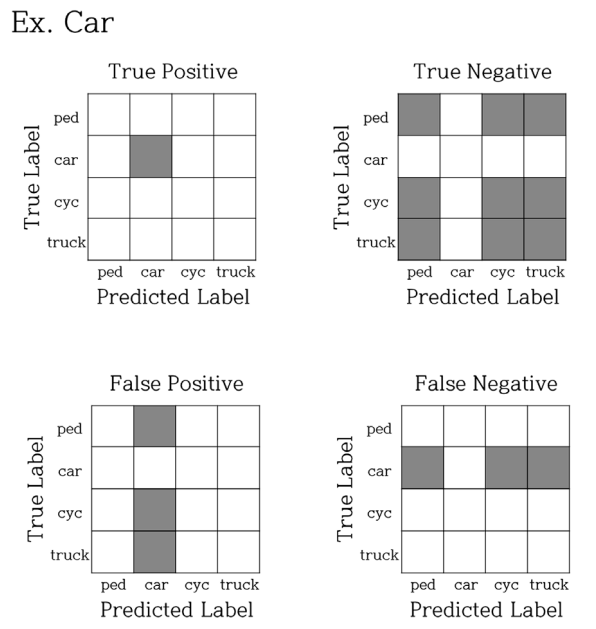


Fig. 9 Example of car TP, TN, FP, FN by confusion matrix

$$Recall = \frac{TP}{TP+FN} \tag{2}$$

$$Accuracy = \frac{TP+TN}{TP+FN+FP+TN} \tag{3}$$

$True Positive(TP)$: 실제 참인 것을 모델이 참이라고 예측한 경우
$True Negative(TN)$: 실제 거짓인 것을 모델이 거짓이라고 예측한 경우
$False Positive(FP)$: 실제 거짓인 것을 모델이 참이라고 예측한 경우
$False Negative(FN)$: 실제 참인 것을 모델이 거짓이라고 예측한 경우

시나리오별 검증 데이터셋을 구성하고, 구성된 검증 데이터셋을 앞서 도출한 CNN 모델의 입력 데이터로 인가하여 각 시나리오별 정확률, 재현율, 정확도와 혼동 행렬을 Table 3과 같이 도출하였다. 객체별로 정리를 하면, 차량의 경우 평균 정확률 96.04 %, 평균 재현율 90.25 %, 평균 정확도 91.82 %이며, 보행자의 경우 평균 정확률 77.55 %, 평균 재현율 74.94 %, 평균 정확도 88.44 %이며, 자전거의 경우 평균 정확률 80.29 %, 평균 재현율 96.45 %, 평균 정확도 89.43 %이며, 트럭의 경우 평균 정확률 100 %, 평균 재현율 93.89 %, 평균 정확도 96.35 %이다.

대체적으로 각 시나리오마다 차량은 좋은 결과를 나타내고 있으나, Scenario 3의 보행자의 경우 정확률이 66.33 %로 다소 떨어짐을 확인할 수 있었다.

Fig. 10은 이를 분석하기 위한 그래프를 보여준다. x축은 프레임 번호로서 시간의 진행을 나타내며 윗 그림은 상대거리, 아래 그림은 CNN 모델에서 예측한 객체 라벨 값을 나타낸다, 약 80번 프레임부터 객체를 인식하고 있으며 객체가 약 20 m 안쪽으로 다가옴에 따라 보행자를 자전거로 반복적으로 오인식하는 것을 볼 수 있다. 이는

Table 3 Test scenario result

Scenario number	Object type	Precision (%)	Recall (%)	Accuracy (%)
Scenario 1	Car	98.75	74.06	86.56
	Cyclist	78.39	94.06	84.06
Scenario 2	Car	97.71	100.00	98.83
	Cyclist	100.00	97.66	98.83
Scenario 3	Pedestrian	66.33	91.55	92.38
	Car	100.00	87.07	88.87
Scenario 4	Pedestrian	100.00	41.73	80.57
	Car	84.17	99.80	90.53
	Cyclist	64.41	100.00	90.77
Scenario 5	Car	96.27	100.00	98.44
	Truck	100.00	93.89	96.35

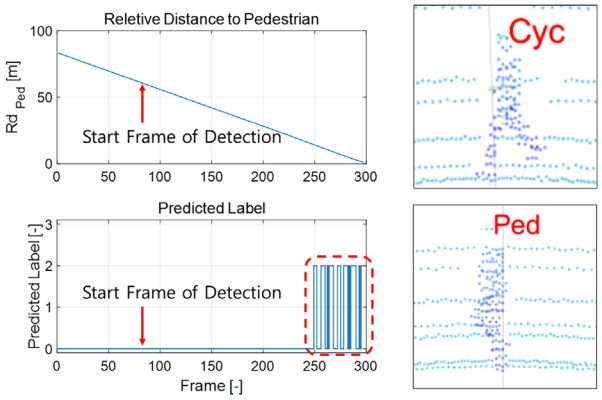


Fig. 10 Pedestrian classification result in Scenario 3

보행자의 발의 위치가 겹쳐있을 때는 보행자로 잘 인식하지만, 걸음을 내디뎌 발이 벌어질 때는 자전거와 모양이 유사하게 되어 오인식하기 때문인 것으로 판단된다. 이러한 문제는 추후 학습에 사용되는 객체의 형상을 좀 더 다양화시키거나 더 정교한 딥러닝 모델을 사용함으로써 개선할 수 있을 것으로 판단된다.

또한, Scenario 4의 보행자의 경우 재현율이 41.73 %로 떨어지는 것을 볼 수 있다. 이는 횡단보도를 지나는 보행자들이 앞서 설명한 바와 같이 걸음에 따라 자전거로 오인식되기 때문인 것으로 판단된다. 이에 따라 자전거의 정확률 또한 64.41 %로 다소 떨어지는 것을 볼 수 있다. 하지만, 64.41 %의 정확률은 KITTI 벤치마크 결과 중 자전거의 정확률 65.07 %와 거의 동일한 수준인 것을 알 수 있다.

본 연구에서는 Fig. 11과 같이 Scenario 4의 보행자 및 일부 차량이 학습 시에 사용되지 않은 객체를 포함하도록 설정하였다. 이는 학습 시 사용하지 않은 객체 데이터로 시험하였을 때에도 CNN 모델이 객체를 올바르게 분류하는지 확인하기 위함이다.

Fig. 11 Objects different from training dataset in Scenario 4

4. 결 론

본 연구를 통해 가상 3D 라이다 센서의 포인트 클라우드를 기반으로 객체 분류용 딥러닝의 학습 데이터셋을 구축하고, 데이터셋의 타당성을 검증하였다. 일반적으로 딥러닝용 학습 데이터셋 구축의 경우, 사용자에 의해 수작업으로 각 객체의 종류를 판단하고 이후 객체의 라벨링 작업을 진행하므로, 작업자 오류와 물리적 시간이 많이 소요된다는 단점을 가지고 있다. 하지만 본 논문에서 제안한 가상의 라이다 센서 포인트 클라우드는 객체의 Ground Truth 정보를 모두 포함하고 있으며, 필요에 의해 센서의 종류, 위치, 각도 등을 자유자재로 조절할 수 있다는 장점을 가진다.

하지만, 가상의 라이다 센서인 만큼 실제 라이다 센서의 계측 데이터와는 노이즈 특성과 반사 특성이 완벽히 일치하지 않는다는 단점이 존재하며, 데이터셋의 타당성 검증이 필수적이다. 따라서 본 논문에서는 Waymo사의 Open Dataset을 활용하여 실제 라이다 센서의 데이터셋과 교차검증을 통해 도출된 수치가 KITTI 벤치마크의 결과와 크게 상이하지 않음을 보였고, 이를 통해 가상 라이다 센서를 이용한 데이터셋의 타당성 검증하였다.

앞서 도출한 가상 라이다 센서 기반 데이터셋을 통해 학습된 CNN 모델이 정상적으로 객체를 분류하는지 확인하기 위해 5가지의 시나리오를 자체 개발하여 검증한 결과, 각 객체별로 정상적으로 객체를 분류하고 높은 수치의 정확도를 나타내는 것을 볼 수 있었다. 결론적으로, 본 연구에서 제시하는 가상 3D 라이다 센서 기반의 딥러닝 학습 데이터셋 구축 방법을 통해, 사용자는 자신의 자율주행 하드웨어와 적합한 인지 알고리즘을 개발하고 검증할 수 있도록 자체 데이터셋을 효율적으로 구축할 수 있을 것으로 기대된다.

후 기

본 연구는 산업통상자원부의 '산업핵심기술개발사업'으로 지원받아 수행되었습니다(10062449, 차량용 전방 스캔 LiDAR 센서 신호처리 원천기술 개발).

References

1) J. Lee, K. Choi, T. Park and S. Kee, "A Study on the Vehicle Detection and Tracking Using Forward Wide Angle Camera," Transactions of KSAE, Vol.26, No.3, pp.368-377, 2018.
2) E. Hyun, Y. Jin, B. Kim and J. Lee, "Development of Human Detection Algorithm for Automotive Radar," Transactions of KSAE, Vol.25, No.1, pp.92-102, 2017.
3) A. Krizhevsky, I. Sutskever and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," Communications of the ACM, Vol.60, Issue 6, pp.84-90, 2017.
4) K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-scale Image Recognition," International Conference on Learning Representations, 2015.
5) K. He, X. Zhang, S. Ren and J. Sun, "Deep Residual Learning for Image Recognition," The IEEE Conference on Computer Vision and Pattern Recognition, pp.770-778, 2016.
6) R. Charles, Y. Li, S. Hao and J. Leonidas, "PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space," Conference on Neural Information Processing Systems, pp.5105-5114, 2017.
7) Y. Wang, T. Shi, P. Yun, L Tai and M. Liu, "PointSeg: Real-Time Semantic Segmantation Based on 3D LiDAR Point Cloud," arXiv preprint arXiv: 1807.06288, 2018.
8) B. Wu, A. Wan, X. Yue and K. Keutzer, "SqueezeSeg: Convolutional Neural Nets with Recurrent CRF for Real-Time Road-Object Segmenatation from 3D LiDAR Point Cloud," IEEE International Conference on Robotics and Automation, pp.1887-1893, 2018.
9) Y. Zhou and O. Tuzel, "VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection," The IEEE Conference on Computer Vision and Pattern Recognition, pp.4490-4499, 2018.
10) S. Chang and N. Morgan, "Robust CNN-based Speech Recognition With Gabor Filter Kernels," Proceedings of Interspeech, pp.905-909, 2014.
11) H. H. Aghdam and E. J. Heravi, Guide to Convolutional Neural Networks : a Practical Application to Traffic-sign Detection and Classification, Springer International Publishing, Cham, Switzerland, pp.85-94, 2017.

12) Y. Zhang, C. Pan, J. Sun and C. Tang, “Multiple Sclerosis Identification by Convolutional Neural Network with Dropout and Parametric ReLU,” Journal of Computational Science, Vol.28, pp.1-10, 2018.
13) D. Scherer, A. Muller and S. Behnke, “Evaluation of Pooling Operations in Convolutional Architectures for Object Recognition,” International Conference on Artificial Neural Networks, pp.92-101, 2010.
14) Velodyne Lidar, https://velodynelidar.com/products/puck/, 2020.
15) Velodyne Lidar, https://velodynelidar.com/products/puck-hi-res/, 2020.
16) Euro NCAP, Test Protocol: AEB VRU Systems Version 3.0.2, 2019.
17) R. Kohani, “A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection,” Proceedings of the 14th International Joint Conference on Artificial Intellegence, Vol.2, pp.1137-1143, 1995.
18) Python, https://www.python.org/, 2020.
19) Facebook, https://pytorch.org/, 2020.
20) Waymo LLC, https://waymo.com/open/data/, 2020.
21) Karlsruhe Institute of Technology and Toyota Technological Institute at Chicago, http://www.cvlibs.net/datasets/kitti/eval_object.php?obj_benchmark=3d, 2020.