연구논문

Journal of Drive and Control, Vol.18 No.4 pp.9-18 Dec. 2021
ISSN 2671-7972(print) ISSN 2671-7980(online)
http://dx.doi.org/10.7839/ksfc.2021.18.4.009

딥러닝 기반 장애물 인식을 위한 가상환경 및 데이터베이스 구축
Development of Virtual Simulator and Database for Deep Learning-based Object Detection

이재인[1] · 곽기성[1] · 김경수[1] · 강원율[2] · 신대영[3] · 황성호[1*]
Jaeln Lee[1], Gisung Gwak[1], KyongSu Kim[1], WonYul Kang[2], DaeYoung Shin[3] and Sung-Ho Hwang[1*]

Received: 30 Sep. 2021, Accepted: 29 Oct. 2021

Key Words : Database(데이터베이스), Object Detection(객체 인식), Image Segmentation(이미지 세그멘테이션), Deep Learning(딥러닝), Relational Database Management System(관계형 데이터베이스 관리 시스템)

Abstract: This study proposes a method for creating learning datasets to recognize obstacles using deep learning algorithms in automated construction machinery or an autonomous vehicle. Recently, many researchers and engineers have developed various recognition algorithms based on deep learning following an increase in computing power. In particular, the image classification technology and image segmentation technology represent deep learning recognition algorithms. They are used to identify obstacles that interfere with the driving situation of an autonomous vehicle. Therefore, various organizations and companies have started distributing open datasets, but there is a remote possibility that they will perfectly match the user's desired environment. In this study, we created an interface of the virtual simulator such that users can easily create their desired training dataset. In addition, the customized dataset was further advanced by using the RDBMS system, and the recognition rate was improved.

1. 서 론

최근 자율주행 자동차, 건설장비 자동화 시스템, 무인 감시 시스템 등에는 카메라, 레이더, 라이다 같은 환경 센서를 통해 주변 환경의 위험도 높은 장애물을 검지하는 연구들이 개발되고있다.[1-5),16-17)]

특히, 카메라, 라이다 센서의 경우는 딥러닝을 활용한 객체 인식을 통해 기존 Rule-based 알고리즘으로 해결하지 못했던 다양한 문제를 해결함으로써 그 활용도가 매우 높아지고 있다. 그 예로 CNN (Convolutional Neural Network)를 활용한 AlexNet, VGGNet, ResidualNet 등의 이미지 분류 네트워크가 개발되었다. 해당 네트워크들은 객체를 분류함에 있어서 높은 성능을 보여주고 있고 이미 다양한 객체 검출 네트워크의 Backbone으로 활용되고 있다. 본 연구에서도 해당 네트워크를 Backbone으로 구성하고 있는 BiSeNet[6)], DeepLabv3+[7)]을 활용하여 Image Segmentation작업을 진행하였다.

또한 기존의 무거웠던 학습용 프레임워크들이 점차 최적화되고 있고, 이에 따라 높은 인식률에 비해 실시간성이 떨어지는 딥러닝 모델을 보완하여 자율주행자동차, 무인이동체, 안전감시 시스템 등 실시간성이 매우 중요한 임베디드 시스템에 사용할 수 있게 되었다.

하지만 딥러닝 모델은 학습에 활용되는 데이터 셋에 따라 인식률이 많이 차이가 나게 된다. 최근 여러 기관 및 업체들이 연구를 목적으로 보유하고 있는

* Corresponding author: hsh0818@skku.edu
1 Department of Mechanical Engineering, Sungkyunkwan University, Suwon 16419, Korea
2 Institute of Vehicle Engineering, Seoul 06640, Korea
3 Institute of Industrial Technology, Cheonan 31056, Korea

데이터 셋을 배포하고 있지만, 오픈 데이터 셋의 경우, 대부분 일반적인 시내 주행환경이 모사 되어있는 학습 데이터이기 때문에, 이를 이용해 학습이 될 경우 학습 데이터 셋에는 존재하지 않는 비정형 도로, 건설기계, 도로 조경 등의 인식률이 떨어진다. 또한 배포되고 있는 오픈 데이터 셋들의 경우 실제 딥러닝 모델을 사용할 시스템의 환경에 맞춰져있는 경우가 매우 드물기 때문에 사용자가 원하는 구도로 학습을 진행하지 못한 경우 인식률이 떨어지는 문제가 발생한다.

따라서 최근 가상환경을 활용한 학습 데이터 셋 제작에 관한 연구가 진행되고 있다. 일반적으로 지도학습용 데이터 셋을 제작할 때는 정답을 미리 판단할 수 있는 사람이 직접 데이터 라벨링 작업을 진행한다. 하지만 사람이 직접 라벨링 하는 경우 Human error를 초래할 수 있는데, 이는 학습 모델의 가중치 갱신에 영향을 미치는 요소이며, 또한 방대한 양의 데이터를 라벨링하기에는 시간과 비용이 매우 많이 들게 된다. 하지만 가상환경을 사용할 경우 시뮬레이션은 시나리오 내에 구성된 장애물 정보를 미리 알고 있기 때문에 이미지의 GT(Ground Truth)를 실시간으로 Labeling 할 수 있게 되고 기존 오픈 데이터 셋의 문제였던 Human error를 고려하지 않아도 된다. 이와 같은 장점을 이용하여 가상 환경센서 및 지도를 구성하여 데이터 셋을 구축하고 그 타당성을 검증하는 연구들이 진행되고 있다.

그 사례[4]로 가상 라이다 센서를 장착한 뒤 실제 라이다의 노이즈 성분을 추가하여 실제 라이다 센서와 비슷한 포인트 데이터를 얻어내 지도학습이 가능한 포인트 데이터로 추출하여 학습 라이다 센서를 활용한 학습용 데이터 셋을 제작하고 검증하였다. 본 연구에서는 가상환경에서 추출되는 이미지를 이용한다. 실제 주행하고 있는 청라지구의 701번 버스 노선과, 굴착기의 작업환경의 모습을 포인트 데이터로 취득하여 가상환경에 구현하고, 구현된 환경에서 추출되는 학습 데이터 셋을 통해 그 타당성을 검증하였다.

본 논문은 위와 같은 가상환경의 장점을 활용해 정확한 정답을 가지는 이미지 데이터 셋 구축 방법을 서술하고자 한다. 이를 통해 학습 데이터 셋을 증량하고, 건설기계, 비정형 도로 와 같은 오픈 데이터 셋에는 분류되지 않은 객체에 대한 다량의 학습 데이터 셋을 얻어낼 수 있다. 이후 제작한 가상환경 기반 데이터 셋의 타당성을 검증하기 위해 배포된 오픈 데이터 셋과 비교 검증을 시행하였다.

2. 가상 환경 구성

2.1 시뮬레이션 구성

딥러닝 학습 이미지 데이터 추출을 위해 가상환경을 구성하였다. 가상환경에 사용될 Map 데이터는 openCRG 및 openDRIVE를 이용하여 제작하였다.

첫 번째 데이터 셋인 시내환경 및 일반 도로환경을 모사한 데이터 셋은 인천 청라지구 701번 버스의 노선을 따라 수집된 포인트 데이터를 이용해 제작하였다. 수집된 데이터를 openDRIVE 포맷으로 정리한 뒤 Fig. 1과 같이 만들고 이를 그래픽화 하여 Fig. 2 같이 변환하였다. 이후 시뮬레이션에 적용하였고, 시내환경을 모사한 가상환경의 Map 작업을 완료하였다.

두 번째 데이터 셋인 건설 환경을 모사한 데이터 셋은 일반 시내환경과는 달리 도로의 고도차이가 매우 심하기 때문에 시내환경과 같이 openDRIVE포맷으로 표현할 수 없다. 따라서 이를 openCRG를 활용하여 표현하였다. openCRG를 통한 처리 방법은 다음과 같다.

Fig.3 과 같이 현장에서 Velodyne사의 라이다를 굴착기에 부착한 뒤 주행함으로써 현장의 포인트 데이터를 축적 하였다. 축적된 데이터를 통해 Fig.4과 같이 현장 모습이 담긴 포인트 데이터로 저장하였다. 이후 해당 데이터에서 CRG파일을 제작하기 위한 기준선을 임의로 정한 뒤 기준점을 중심으로 도로의 기울기 및 기준선 Heading angle을 이용하여 도로를 생성한다. Fig. 5와 같이 도로는 직교하는 점에 의해 표현될 수 있고, 기준선은 x(u), y(u)에 의해 표현된다. 위와 같은 방법으로 취득된 비정형 도로의 Point Cloud 정보의 중앙에 기준선을 그어 양옆으로 고도정보를 얻었다. 이를 그래픽 파일로 생성하였고, 이를 시뮬레이션에 적용하였다.

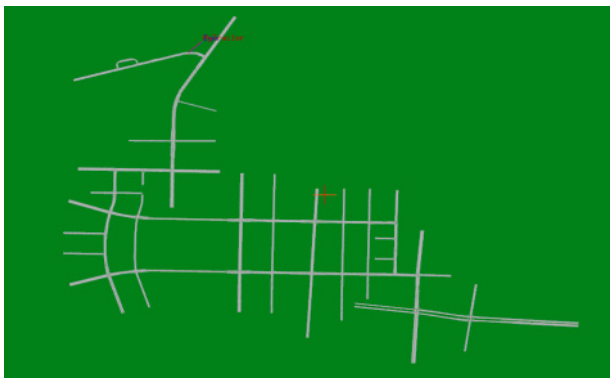

Fig. 1 Road dataset openDRIVE format

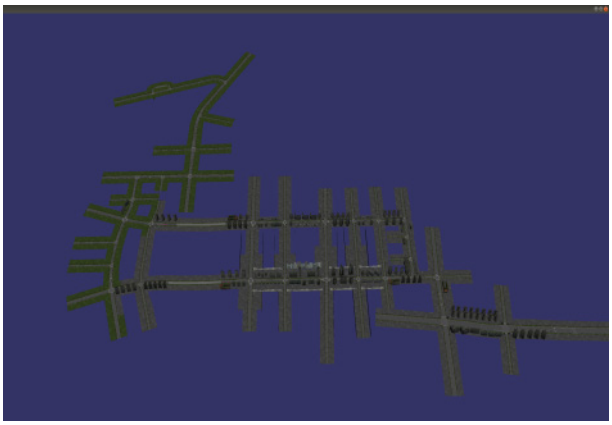

Fig. 2 Road dataset graphic

Fig. 3 Data acquisition enviroment setting

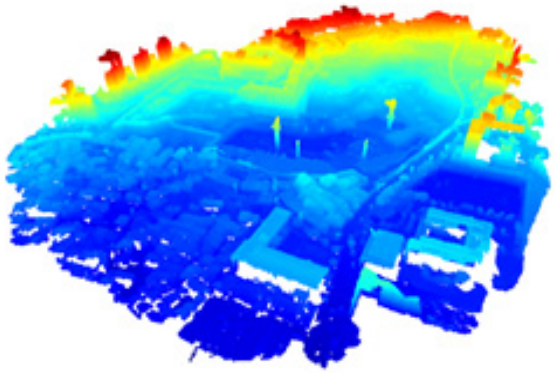

Fig. 4 Point cloud data

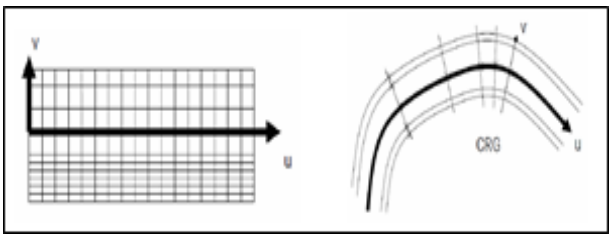

Fig. 5 openCRG road representation

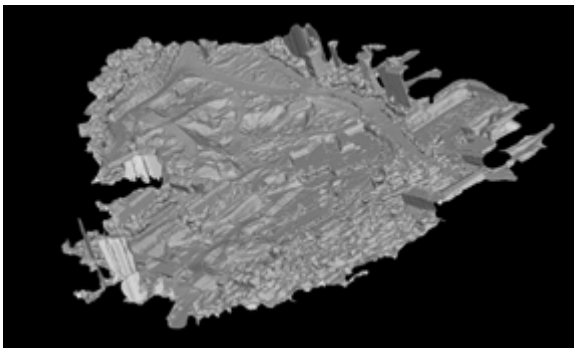

Fig. 6 Point cloud data to 3D graphics

2.2 시뮬레이션 통신 환경 구성

구성한 시뮬레이션에서 이미지를 추출하고 저장할 수 있도록 통신 환경을 구성하였다. 통신 인터페이스는 ROS2(Robot Operating System)를 이용하여 구성하였다. 구성은 Fig. 7과 같다. 차량시뮬레이션에서는 시나리오에 따라 Ego 차량이 경로를 따라 움직이며 시나리오 상황을 보고 있는 가상의 카메라에서 이미지를 만든다. 제작된 인터페이스에서는 총 2가지의 이미지가 송출되는데, 하나의 카메라는 시나리오 상황에 후처리가 없는 시뮬레이션 이미지를 그대로 송출하고, 나머지 한 대의 카메라에서는 사용자가 미리 정리한 GT Label 값을 적용한 이미지가 송출된다. 사용자는 분류가 필요한 항목에 1~254까지의 라벨값을 부여할 수 있게 하였고, 이 외 인식에 필요하지 않은 픽셀값은 255의 값을 부여하여 모델이 해당 픽셀값에 의한 가중치 갱신을 하지 않도록 설정하였다.

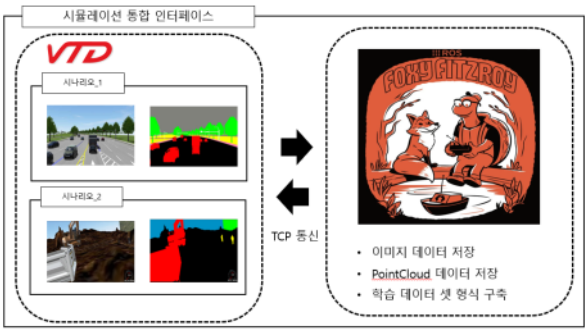

Fig. 7 ROS2 and simulation interface

2.3 데이터 셋 분류기 구성

저장된 이미지와 시뮬레이션 정보를 통해 데이터를 분류할 수 있는 데이터 셋 분류 프로그램을 제작하였다. 본 논문에서 제작한 통신 인터페이스에서는 이미지와 PCD(Point Cloud Data) 그리고 시뮬레이션 과정 중에 생기는 시뮬레이션 정보를 저장하는데, 이

Table 1 Database table

Time	Weather	Velocity[m/s]	Data
Day	Normal	10	0.png
Day	Rain	15	1.png
Night	Snow	20	2.png
Night	Normal	15	3.png

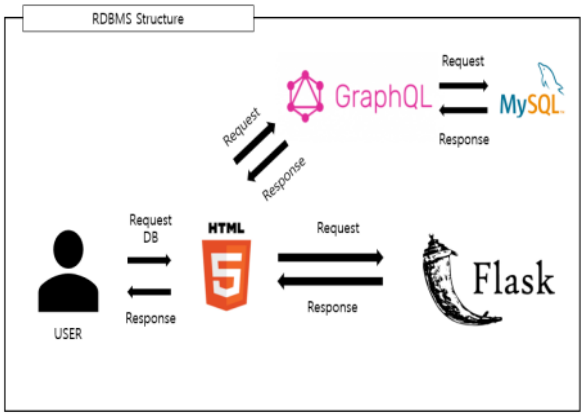

Fig. 8 RDBMS structure

때 저장하는 정보에는 날씨, 시간, 장애물 종류를 동시에 저장하게 된다. 이를 이용하여 이미지 데이터를 같이 나온 정보를 묶어 SQL문으로 탐색할 수 있도록 구조화 하였다. Table 1은 구조화된 테이블의 예시이다.

테이블 예시는 저장된 시뮬레이션 정보와 이미지 파일을 매칭시켜 구조화 한 예시이다. 시뮬레이션 정보는 크게 4개로 구분되고 각각 시간, 날씨, 속도, 이미지 경로 가 있다. 시간 열에는 주간, 야간 속성이 존재하고, 날씨 열에는 맑음, 비, 눈 속성이 존재한다. 속도 정보는 정수 형태로 저장하게 되고 이미지는 SQL에 직접 넣는 것이 아닌 경로를 저장하여 향후 쿼리에 의해 탐색된 이미지를 모아 zip파일로 묶어 사용자의 컴퓨터에 저장된다.

이 같은 환경을 구성하기 위해 web 서버는 Synology 사의 DS420을 이용하였고, 시뮬레이션에서 나오는 모든 데이터를 서버에 업로드 하였다.

Fig. 8 은 데이터 셋 분류 프로그램의 간략한 인터페이스이다. Python Flask와 HTML을 통해 사용자와 서버간 상태를 나타내는 페이지를 제작하였다. 이후 DB에 저장되어 있는 정보에 효율적인 접근이 가능하게 해주는 프레임 워크인 GrapthQL을 사용하여 DB와 사용자간 SQL정보를 처리하였다. 해당 논문에서 사용된 DB 관리 시스템은 MySQL을 이용하여 시뮬레이션 정보를 저장 및 관리하였다.

3. 시나리오, 데이터 셋 제작 및 검증 결과

3.1 학습용 데이터 셋 제작

통합 인터페이스 환경과 RDBMS 환경을 구성한 뒤 Image Segmentation을 위한 학습용 데이터 셋을 제작하였다. 학습용 데이터 셋은 앞서 언급한 2개의 대분류로 먼저 분리된다. 그 중 첫 번째인 일반 주행 환경을 모사한 데이터 셋의 경우 Table 2와 같이 3개의 시나리오로 나누었다. 첫 번째 시나리오의 경우는 인천 청라지구 역에서 차량이 출발하여 도심지에 들

Table 2 City driving scenarios

Scenario number	Road type	Objects	Scenario specification	Scenario image
Scenario 1	Intersection, Normal	Car Tree Pavement Grass	Ego vehicle : Autonomous mode Vehicle : Autonomous mode Road Landscaping	
Scenario 2	Intersection, Normal	Car Pedestrian Pavement Building Tree	Ego vehicle : Autonomous mode Vehicle: Autonomous mode Pedestrian: Stop	
Scenario 3	Intersection, Normal	Car Tree Pavement Building	Ego vehicle : Autonomous mode Vehicle: Autonomous mode	

어가기 전에 마주하는 건물이 없는 지역이다. 따라서 건물과 같은 이미지 보다 도로조경 및 나무, 간단하게 구현된 인도를 적용시켰다. 차량은 차량시뮬레이션에 존재하는 자율주행 모드로 제어 하였으며, 이외의 차량 모두 시뮬레이션에서 자율주행모드를 사용하였다.

Scenario 1번은 도심지에 진입하기 전 도로조경 및 건물이 있지 않은 영역으로 구성하였다. 이 데이터 셋은 선행 차량과 교차로, 나무, 도로조경 등 도심지에서는 보기 힘든 객체들로 구성하였다. 이를 통해 사용자의 시스템이 도심지 이외의 환경에서 선행차량과 도로, 인도 등을 구분할 수 있게 하였다.

Scenario 2는 Ego 차량이 도심지에 들어온 환경이다. 잦은 교차로와 많은 차량들 그리고 버스 정류장과 사람들을 해당 시나리오에 적용하였다. 교통섬처럼 만들어져있는 버스 정류장을 지나게 되는 게 특징이고, 2번 시나리오를 통해 도로 위에 존재하는 교통섬 및 보행자를 구분할 수 있게 하였다. Scenario 2번 또한 Ego 차량과 주위 선행 차량은 모두 시뮬레이터 기본 자율주행 상태로 설정하였다. 그리고 정류장에 있는 사람은 모두 정지해있는 상태로 적용하였다.

Scenario 3은 가정 역에 다시 청라 역으로 상행하는 구역이다. 주변 건물들이 매우 밀집되어 있고 큰 대로가 있는 것이 특징이다. 위 환경을 통해 건물이 밀집되어있는 곳에서 차량 및 도로를 구분할 수 있게 하였다. Scenario 3에서도 Ego 차량과 주위 선행 차량을 모두 자율주행 상태로 설정하고 데이터를 수집하였다.

두 번째 데이터 셋으로는 굴착기 및 건설환경에서 장애물 인식을 위한 시나리오를 구성하였다. Table 3은 건설환경 시나리오 세부 내용이다.

Scenario 4는 앞서 제작한 openCRG맵을 시뮬레이션에 적용한 후, 주변 환경의 고도를 표현함으로써, 장애물의 위치에 높이가 반영되었다. 환경 또한 일반 도로가 아닌 건설환경에 자주 보이는 흙바닥을 표현하였다. 이를 통해 두 번째 데이터 셋인 건설환경 인식용 데이터 셋을 제작하였다.

3.2 데이터 셋 학습 결과

본 연구의 학습용 데이터 셋의 타당성을 검증하기 위해 총 2가지의 학습모델을 이용하였다. 첫 번째는 Image Segmentation 모델 중 정확도가 타 모델에 비해 높은 Deeplabv3+ 모델을 사용하였고, 이후 Deeplabv3+보다 정확도는 떨어지나 보다 빠른 속도로 이미지를 분류하는 BiSeNet을 사용하였다. 학습을 위한 프로그래밍 언어는 Python[13]을 사용하였고, 딥러닝 데이터 모델은 PyTorch 라이브러리를 활용하였다. Deeplabv3+ 학습에 사용한 학습률은 0.01로 지정하였다. 학습률 값은 학습 정도를 반영하는 가중치인데, 너무 높거나 낮으면 Loss가 수렴하지 않거나 학습 속도가 너무 느려진다. 이를 반영하여 적절한 값을 선정하였다. 본 논문에서 사용하는 PC의 성능과 그래픽카드 메모리 용량을 고려하여 Batch Size를 4개의 크기로 지정해 데이터 셋 크기를 분할하여 사용하였으며, 총 300 Epoch로 설정하여 학습을 진행하였다.

Fig. 9은 KITTI 데이터 원본 이미지이고, Fig. 10는 Deeplabv3+ 모델을 Cityscapes[7] 데이터 셋에서 학습시킨 뒤 KITTI 데이터(Fig. 9)를 예측한 이미지이다.

Fig. 9 KITTI Dataset Test Image

Table 3 Construction site scenario

Scenario number	Road type	Objects	Scenario specification	Scenario image
Scenario 4	Construction site	Car Person	Ego vehicle : Autonomous mode object : Autonomous mode	

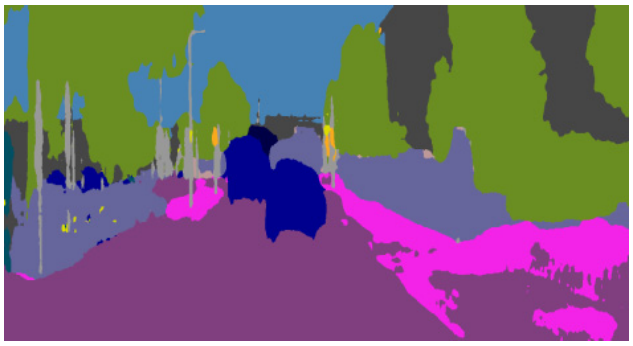

Fig. 10 Predict KITTI by Deeplabv3+ (Cityscapes)

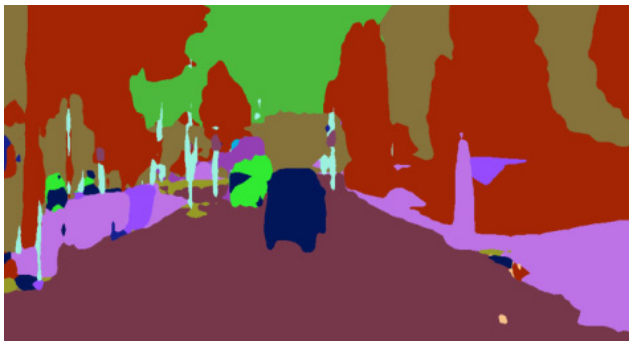

Fig. 11 Predict KITTI by BiSeNet (Cityscapes)

예측 결과 도로 및 자동차, 버스 등은 잘 구분 하였지만 인도와 도로 영역에서 노이즈가 많이 발생하였다.

다음으로는 BiSeNet 모델을 Cityscapes 데이터 셋으로 학습을 진행하고, KITTI 데이터를 예측하였다. BiSeNet 학습 파라미터로는 위와 동일하게 학습률을 0.01로 지정하였고, Batch size는 4로 설정, 300번의 Epoch로 설정하고 학습하였다. Fig. 11은 Fig. 8을 BiSeNet 모델로 예측한 결과이다.

Image Segmentation 모델에서 발생하는 노이즈 원인은 모델의 성능, 훈련 방법, 데이터 셋 부족 등 다양한 원인이 있다. 그 중 본 논문에서는 가상환경으로 제작한 데이터 셋으로 학습 데이터 셋을 수정 혹은 증량하여 문제를 해결하고자 한다. 데이터 셋을 수정 및 증량하기 위해선 가상환경에서 나오는 GT 이미지의 라벨값과 증량을 목표로 하는 데이터 셋의 형식을 동일하게 맞춰야한다. 따라서 가상환경에서 추출되는 데이터를 Cityscapes 라벨값과 동일하게 추출되도록 후처리를 진행하였다.

Table 4는 Cityscapes 데이터에서의 GT image의 라벨 값과 시뮬레이션에서 추출되고 있는 GT image를 객체별로 정리한 표이다.

Table 4 Cityscapes & VTD object labels

Object	Cityscapes Label	VTD Label
Unlabeled	0	0
Ego vehicle	1	0
Rectification Border	2	0
Out of ROI	3	0
Static	4	0
Dynamic	5	0
Ground	6	0
Road	7	7
Sidewalk	8	8
Parking	9	0
Rail Track	10	0
Building	11	0
Wall	12	0
Fence	13	0
Guardrail	14	0
Bridge	15	0
Tunnel	16	0
Pole	17	0
Pole Group	18	0
Traffic Light	19	0
Traffic Sign	20	0
Vegetation	21	21
Terrian	22	22
Sky	23	23
Person	24	24
Rider	25	0
Car	26	26
Truck	27	26
Bus	28	26
Caravan	29	0
Trailer	30	0
Train	31	0
Motorcycle	32	0
Bicycle	33	0

표와 같이 라벨값을 모두 맞춰준 결과 학습모델의 파라미터를 동일하게 설정할 수 있었다. 이후 위에서 정의했던 시나리오 1, 2, 3번을 시내주행 데이터 셋에 묶고 학습을 진행했으며, 학습 class는 8개로 설정하고 학습을 진행하였다. 학습 클래스의 개수를 8개로 지정한 이유는 현재까지 시뮬레이션에서 구분 가능한 라벨의 개수가 총 8가지로 제한되어있기 때문

이며, 선정된 8개의 클래스는 Table 4의 VTD Label에서 0이 아닌 값을 가지는 객체이다.

따라서 Label가 묶이지 않은 Unlabeled 데이터를 하나의 클래스로 묶고 나머지 7개의 클래스를 합쳐 8개의 클래스로 학습을 진행하였다.

나머지 학습 파라미터는 이전 학습과 동일하게 설정하였다. Fig. 12은 본 연구를 통해 제작한 가상환경에서 추출된 학습데이터만을 이용하여 학습한 뒤 Fig. 8 이미지를 예측한 결과이다.

Cityscapes 데이터 셋에서 학습된 Deeplabv3+와 비교하면, 보다 정교하게 도로와 인도를 구분하였지만, VTD 데이터의 경우 부여된 라벨값이 Cityscapes 데이터에 비해 다양하지 않아 정확하게 Class를 부여하지 못한(검은색) 영역이 존재한다. 현재 차량시뮬레이터에는 건물, 펜스, 울타리 등은 라벨링 할 수 없어 분류되어 있지 않다. 하지만 Cityscapes 데이터 셋에는 세부 분류가 되어있기 때문에, 이를 보완하기 위해 Cityscapes 데이터 셋과 가상환경 데이터를 섞어서 학습하였다. 다시 제작한 학습 데이터 셋은 Cityscapes 50% 와 VTD 데이터 50%을 섞어 제작하였으며 VTD로는 분류하지 못한 건물, 벽 등의 class를 보완할 수 있도록 학습 class의 개수를 기존 8개에서 10개로 늘린 뒤 학습하였다. Fig. 13는 학습 결과이다.

Fig. 12 Predict KITTI by Deeplabv3+ (VTD)

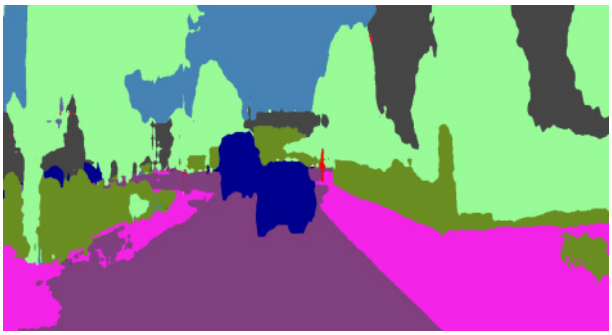

Fig. 13 Predict by Deeplabv3+(Cityscapes + VTD)

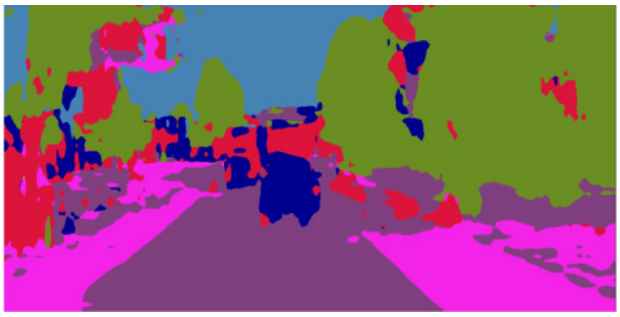

Fig. 14 Predict by BiSeNet (VTD)

Fig. 15 Predict by BiSeNet (Cityscapes + VTD)

학습결과 VTD만으로 학습했을 때 보다 벽, 건물, 그림자에 가려진 포장도로 등을 정확하게 분류하였고, 차량, 도로, 인도에 노이즈가 줄어들었음을 확인하였다. 위 과정을 BiSeNet에도 동일하게 적용한 뒤 학습 결과를 확인하였다.

Fig.14은 VTD 데이터 셋으로 BiSeNet을 학습한 결과이다. Cityscapes 데이터 셋으로만 학습했을 때보다 인도와 도로의 구분은 선명하게 되었지만, 자동차, 건물, 나무를 나타내는 객체에 노이즈가 많이 생겼다.

Fig. 15는 Cityscapes 30%, VTD 70% 비율로 섞어 BiSeNet을 학습한 결과이다.

두 개의 모델을 Cityscapes, VTD, Cityscapes + VTD 3개의 데이터 셋으로 학습한 결과를 비교해보았다. 본 연구에서 제안된 가상환경 기반 시내주행 데이터 셋으로 학습된 모델은 도로와 인도의 구분에 강인한 특징을 가졌다. 하지만 이외의 장애물에 노이즈가 많이 발생하였다. 해당 문제의 원인은 가상환경에서 현재 보내주는 데이터 라벨의 종류가 Cityscapes 종류에 비해 매우 적고, 가상환경으로 만들어진 이미지의 특성인 그래픽화된 이미지가 실제 카메라 이미지와는 차이가 있기 때문이다. 따라서 이를 보완하기 위해 가상환경 기반 데이터 셋과, 실제 이미지 기반 데이터 셋을 50% 비율로 섞은 뒤 학습을 진행하였고, 그 결과 Fig.13와 Fig.15와 같이 두 모델 전체적으로 노

이즈가 성분이 줄어들었으며, 도로와 인도간의 구분 성능이 향상함을 볼 수 있었다. 이를 통해 가상환경 기반 시내주행 데이터 셋의 검증을 완료하였다. 향후 연구에서는 시내주행 데이터 셋에 현재 적용되어있는 8개의 class에 신호등, 건물, 가로등, 표지판 등을 포함 시켜 데이터 셋을 제작함으로써 학습 모델의 학습 성능을 향상 시킬 계획이다.

다음은 두 번째 건설환경 데이터 셋을 검증하기 위해 Cityscapes, VTD, Cityscapes + VTD 데이터 셋으로 분류한 뒤 Deeplabv3+을 학습시켰고 각 학습결과를 확인하였다. Fig. 16는 테스트 데이터 셋인 MOCS(Moving Objects in Construction Sites)의 샘플 데이터이다. MOCS 데이터 셋은 건설환경에 존재하는 굴착기, 롤러, 로더, 타워크레인 등 건설환경에 존재하는 이미지와 라벨값을 모아놓은 데이터이다.

Cityscapes 데이터 셋으로 학습한 결과 Fig. 17과 같은 예측결과가 나왔다.

예측 결과 굴착기의 형태는 잡아내었지만, 동일한 class로 분류하지 못하였고, 여러 class의 노이즈가 섞여있는 결과를 도출하였다. 따라서 건설환경을 모사한 두 번째 데이터 셋을 이용해서 학습을 진행해 보았다.

Fig. 18은 시뮬레이션에서 추출된 데이터로 예측한 결과이다.

Cityscapes로 학습한 모델에 비해 굴착기 내부에 노이즈가 많이 섞여있지 않다. 하지만 전체적인 모습을 완벽하게 하나의 class로 분류하지 못하였고, 또한 Cityscapes로 학습한 모델에 비해 지면을 확실하게 잡아내지 못하였다. 따라서 위와 같은 문제를 해결하고자, 두 데이터를 40% 비율로 섞어서 데이터 셋을 다시 제작하였고, 굴착기 및 건설기계의 인식률을 높이기 위해 앞서 제작한 RDBMS 프로그램을 이용해서 굴착기를 포함하고 있는 가상환경 기반 이미지를 분류한 뒤 나머지 20% 비율을 굴착기가 포함된 이미지 데이터 셋을 추가해 학습을 진행하였다.

Fig.19은 Citysacpes + VTD + RDBMS VTD 데이터 셋으로 학습된 모델의 예측 결과이다. 예측 결과 앞서 보았던 두 개의 학습 모델보다 굴착기의 형상을 명확하게 잡아내었다. 하지만 Fig.16에 굴착기 앞에 있는 자갈더미 형상은 거의 인식하지 못하였다. 그 이유는 VTD 데이터 셋에 현재 추출할 수 있는 학습

Fig. 16 MOCS sample data

Fig. 17 Predict by Deeplabv3+ (Cityscapes)

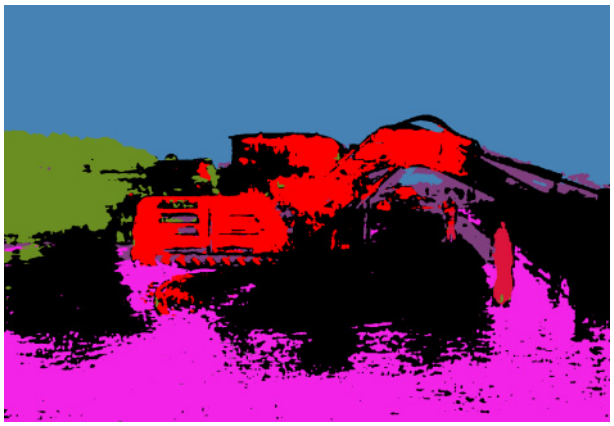

Fig. 18 Predict by Deeplabv3+ (VTD)

Fig. 19 Predict by Deeplabv3+ (Cityscapes + VTD)

라벨값이 매우 제한적이기 때문에 해당 자갈더미와 같은 형상을 라벨링한 정답 이미지가 없기 때문이다. 또한 본 연구에서 VTD 데이터 셋과 같이 학습을 진행한 Cityscapes 데이터 셋 또한 해당 자갈더미와 유사한 객체를 따로 분류한 클래스가 없기 때문에 해당 장애물에 대한 인식률이 떨어졌다. 향후 VTD에서 학습 클래스를 더욱 세분화 시킬수 있다면, 해당 장애물과 같은 형상의 정답 데이터를 제작 및 추가학습을 진행할수 있을 것으로 판단된다.

가상환경 기반 건설환경 데이터 셋의 검증을 진행하였다. 추가학습 혹은 특정 객체의 인식률을 높여보기 위해 제작한 RDBMS 프로그램으로 굴착기 이미지를 보충해 보았고, 그 결과 굴착기의 형상 및 인식 성능이 향상되었다. 이를 통해 오픈 데이터 셋에 많이 존재하지 않는 굴착기 및 건설기계를 인식하기 위해 가상환경 기반 데이터를 활용할 수 있음을 검증하였다.

4 결 론

본 연구를 통해 가상환경 기반 장애물 인식용 학습 데이터 셋을 구성하고 그 타당성을 검증하였다. 대다수 사용자들은 일반적으로 소수의 라벨링된 오픈 데이터 셋을 활용하거나, 사용자의 환경과 다른 환경으로 구성된 학습 데이터를 사용하게 된다. 그러나 학습 데이터에는 없는 객체거나 혹은 그 양이 매우 제한적이고 다양하지 않기 때문에 학습을 다양하게 진행할 수 없는 한계점이 있다.

본 연구에서 제작한 가상환경 기반 데이터 셋을 활용할 경우, 환경센서의 설치위치 혹은 사양을 쉽게 조절할 수 있으며, 실제 환경에서 얻기 어려운 시나리오를 적용하여 데이터를 추출할 수 있기 때문에 오픈 데이터 셋보다 더 다양한 상황을 구성할 수 있다는 장점을 가진다. 하지만, 가상환경의 그래픽적인 한계로 인해 실제 카메라에서 추출되는 이미지와는 다른 점이 존재하고, 따라서 제작된 데이터 셋의 타당성 검증이 필수적이다.

본 연구에서는 KITTI 데이터 셋과 MOCS 데이터 셋을 검증 데이터 셋으로 활용하였고, 그 결과 Cityscapes 데이터 셋을 활용하여 얻어진 학습결과와 크게 상이 하지 않음을 보였다. 또한, 일반 오픈 데이터 셋에 가상환경 기반 데이터 셋을 추가하여 학습을 진행했을 때 인식 성능이 향상됨을 확인하였다.

결론적으로 본 연구를 통해 사용자는 학습하고자 하는 데이터 셋을 쉽게 제작해 시스템에 활용될 딥러닝 모델을 검증할 수 있을 것이며, 실제로는 얻기 어려운 사고 상황을 구현해 데이터를 추출하여 안전 감시 시스템에 활용할 수 있는 의미 있는 데이터를 효율적으로 구축할 수 있을 것으로 기대된다.

후 기

본 연구는 국토교통부/국토교통과학기술진흥원 교통물류연구사업의 연구비지원(21TLRP-C152478-03)과 2020년도 산업통상자원부 및 산업기술평가관리원(KEIT) 연구비 지원에 의한 연구입니다.(과제번호 : 20010725)’

이해관계(CONFLICT OF INTEREST)

저자는 이 논문과 관련하여 이해관계 충돌의 여지가 없음을 명시합니다.

References

1) A. Krizhevsky, I. Sutskever and G. E. Hinton, “ImageNet Classification with Deep Convolutional Neural Networks”, Communication of ACM, Vol.60, Issue 6, pp.84-90, 2017.
2) K. Simonyan and A. Zisserman, “Very Deep Convolutional Networks for Large-scale Image Recognition”, International Conference on Learning Representations, 2015.
3) 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.
4) H. J. Jang, et al., “A Study on the Construction of Deep Learning Dataset based on Virtual Lidar Sensor Point Cloud”, Kookmin University, 2019.
5) S. J. Yoon et al., “Development of Autonomous Vehicle Learning Data Generation System”, The Journal of The Korea Institute of Intelligent Transportation Systems, Vol.19 No.5 pp 162~177.
6) C. Yu, et al., “BiSeNet: Bilateral Segmentation Network for Real-time Semantic Segmentation”, arXiv: 1808.00897, 2018.

7) L. C. Chen et al., “DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected Crfs”, arXiv:1606.00915, 2016.
8) MOCS: A Dataset and Benchmark for Detecting Moving Objects in Construction Sites
9) KITTI Sematic Segmentation Evaluation, http://www.cvlibs.net/datasets/kitti, 2021.
10) Cityscapes Semantic Understanding of Urban Street Scenes, http://www.cityscapes-dataset.com/, 2021.
11) Y. Zhang et al., “Multiple Sclerosis Identification by Convolutional Neural Network with Dropout and Parametric ReLU”, Journal of computational Science, Vol.28, pp.1-10, 2018.
12) Pascal VOC Dataset Mirror, http://www.pjreddie.com/projects/pascal-voc-dataset-mirror/, 2021.
13) Python, https://www.python.org/, 2021.
14) F. Chollet, “Xception: Deep Learning with Depthwise Separable Convolutions.”, arXiv:1610.02357, 2016.
15) V. Badrinarayanan, A. Kendall, and R.Cipolla “Segnet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation.”, arXiv:1511.00561, 2015.
16) D. Yu et al., "Hybrid Control Strategy for Autonomous Driving System using HD Map Information", Journal of Drive and Control, Vol.17, No.4, pp.80-86, 2020.
17) Y.-H. Im et al., "Real-Time Simulation of an Excavator Considering the Functional Valves of the MCV", Journal of Drive and Control, Vol.16, No.4, pp.38-47, 2019.