



저작자표시 2.0 대한민국

이용자는 아래의 조건을 따르는 경우에 한하여 자유롭게

- 이 저작물을 복제, 배포, 전송, 전시, 공연 및 방송할 수 있습니다.
- 이차적 저작물을 작성할 수 있습니다.
- 이 저작물을 영리 목적으로 이용할 수 있습니다.

다음과 같은 조건을 따라야 합니다:



저작자표시. 귀하는 원 저작자를 표시하여야 합니다.

- 귀하는, 이 저작물의 재이용이나 배포의 경우, 이 저작물에 적용된 이용허락조건을 명확하게 나타내어야 합니다.
- 저작권자로부터 별도의 허가를 받으면 이러한 조건들은 적용되지 않습니다.

저작권법에 따른 이용자의 권리는 위의 내용에 의하여 영향을 받지 않습니다.

이것은 [이용허락규약\(Legal Code\)](#)을 이해하기 쉽게 요약한 것입니다.

[Disclaimer](#)



석사학위논문

라이다 포인트 클라우드 형상 특성 및 동적 특성을 활용한 객체 인식 딥러닝 모델 개발

3D Object Detection via Object Structural and Dynamic Feature
Extraction of LiDAR Point-Cloud



국민대학교 자동차공학전문대학원

자동차IT융합전공

김 태 산

2022

라이다 포인트 클라우드 형상 특성 및 동적 특성을 활용한 객체 인식 딥러닝 모델 개발

3D Object Detection via Object Structural and Dynamic Feature
Extraction of LiDAR Point-Cloud



국민대학교 자동차공학전문대학원

자동차IT융합전공

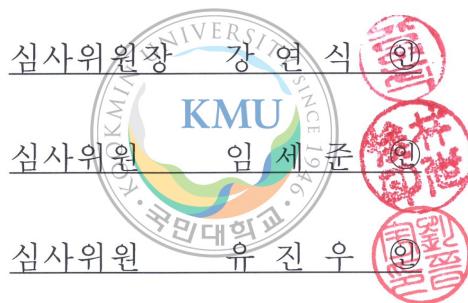
김 태 산

2022

김 태 산의

석사학위 청구논문을 인준함

2023년 1월



국민대학교 자동차공학전문대학원

목 차

그림 목차	iv
표 목차	vi
국문 요약	vii
제 1 장 서론	1
1.1 연구 배경	1
1.1.1 객체 인지 관련 기술	3
1.1.2 자율 주행을 위한 센서 시스템	4
1.2 연구 방향	7
1.3 기존 연구의 한계점	7
1.4 연구 목표	8
1.5 논문 구성	9
제 2 장 연구 동향	10
2.1 인공지능 및 딥러닝 기술 발전	10
2.1.1 하드웨어 (GPU)의 발전	10
2.1.2 빅 데이터	11
2.1.3 알고리즘의 발전	12
2.2 객체 인식 기술	16
2.2.1 2D 객체 인식 기술	17
2.2.2 3D 객체 인식 기술	18
2.2.3 다중 Frame 기반 3D 객체 인식 기술	21
2.3 Attention 메커니즘	22

2.4	시계열 데이터 예측을 위한 딥러닝 네트워크	23
제 3 장	포인트 클라우드 시계열 데이터 기반 3D 객체 인식 기술 개발	25
3.1	문제 정의	26
3.2	포인트 클라우드 시계열 데이터 구성	26
3.3	객체 형상 특성 추출기	27
3.3.1	PointPillars	27
3.3.2	CBAM (Convolutional Block Attention Module)	29
3.4	객체 동적 특성 추출기	31
3.5	객체 인식 Head와 Loss (손실) 함수	32
3.6	연산 속도 개선을 위한 Queue 구조 설계	34
제 4 장	객체 인식 실험 결과 및 분석	37
4.1	학습 방법	37
4.2	실험 설정	38
4.2.1	데이터셋	38
4.2.2	파라미터 설정 및 실험 환경	41
4.3	평가지표	42
4.4	정량적 결과	44
4.5	정성적 결과	47
4.6	Ablation Study	50
4.6.1	CBAM Attention 메커니즘 적용	50
4.6.2	ConvLSTM을 통한 객체 동적 특성 추출	52
4.6.3	객체 형상 특성 및 동적 특성의 병합 (Concatenation)	53
4.6.4	Queue 자료구조를 통한 연산 속도 개선	54

제 5 장 결론	55
5.1 향후 과제	56
참고 문헌	57
Abstract	70
감사의 글	73



그림 목차

그림 1 SAE J3016 Levels of Driving Automation [4]	2
그림 2 An example of a small size and hard case object	8
그림 3 An example of 3D object labels of KITTI dataset [22]	11
그림 4 Sensor setup for nuScenes data collection platform [23]	12
그림 5 Sigmoid Activation Function	13
그림 6 Sigmoid Activation Function	14
그림 7 2D Object Detection Results of Faster R-CNN [6]	18
그림 8 2D Object Detection Results of YOLO [38]	19
그림 9 3D Object Detection Results of VoxelNet [50]	20
그림 10 The entire pipeline of the proposed 3D object detection method	25
그림 11 An Example of a Point Cloud Sequence ($n = 3, d = 10$)	27
그림 12 Network Overview of PointPillars [51]	28
그림 13 The detailed architecture of CBAM	31
그림 14 Distribution of classes in KITTI Object Detection dataset	40
그림 15 Distribution of classes in KITTI Object Tracking dataset	40
그림 16 The prediction result of hard case car objects. From the left to the right direction, images represent a corresponding camera image, the prediction by PointPillars [51], and the prediction by our model, respectively.	47
그림 17 The prediction result of false positive car objects. From the left to the right direction, images represent a corresponding camera image, the prediction by PointPillars [51], and the prediction by our model, respectively.	48

그림 18 The prediction result of a small size object (pedestrian). From the left to the right direction, images represent a corresponding camera image, the prediction by PointPillars [51], and the prediction by our model, respectively. 49

그림 19 The prediction result of a false positive small size object (cyclist). From the left to the right direction, images represent a corresponding camera image, the prediction by PointPillars [51], and the prediction by our model, respectively. 50



표 목차

표 1 Results of BEV AP on test dataset (KITTI tracking dataset). n denotes the number of frames of point cloud sequence.	46
표 2 Results of 3D AP on test dataset (KITTI tracking dataset). n denotes the number of frames of point cloud sequence.	46
표 3 Comparison of the attention effect of CBAM [82] on PointPillars [51]. n is set to 1.	51
표 4 Comparison of the attention effect of CBAM [82] on our model. n is set to 4.	52
표 5 Comparison of the effect of the dynamic feature extraction via ConvLSTM [78]. In the case of the model without ConvLSTM, the n structural features are simply concatenated and directly fed to the Detection Head. n is set to 4.	52
표 6 Comparison of the effect of concatenation of structural and dynamic features. In the case of Dynamic Only, the structural features of current step t are not used. n is set to 4.	53
표 7 Comparison of the inference speed between models with/without using the queue.	54

국문 요약

자율주행에 대한 관심이 최근 몇 년간 전세계를 걸쳐 폭발적으로 증가하며 기존의 전통적인 자동차 OEM과 거대 IT 기업 등 다양한 기업에서 자율주행 기술을 활발히 연구하고 있다. 또한, 미국 자동차공학회(SAE, Society of Automotive Engineers)에서는 자율주행 기술 수준에 따라 자율주행 단계를 6단계로 정의하며 자율주행 기술 개발에 따른 혼란 방지 및 안전 보장을 위한 기준을 세웠다. 이처럼 다양한 기업 및 조직에서 자율구행 기술 구현을 위해 노력하고 있으며, 보다 안전하고 효율적인 자율주행 차량을 상용화하기 위해서는 여러 분야에서의 기술 발전이 여전히 필요한 상태이다. 본 논문에서는 이러한 흐름에 발맞추어 자율주행 기술, 구체적으로는 자율주행에 있어 필수적인 기술인 객체 인식 기술을 라이다 포인트 클라우드와 딥러닝을 기반으로 연구하였으며 이에 대해 기술하였다.

최근 자율주행 분야에서는 객체의 종류, 위치 및 크기를 정확하게 판단하여 안전한 자율주행이 가능하도록 2차원이 아닌 3차원 공간 상에서의 객체 인식에 대한 연구가 활발히 진행되고 있다. 현재 활용되고 있는 대부분의 딥러닝 기반 3차원 객체 인식 모델들은 단일 Frame의 포인트 클라우드를 입력 데이터로 사용한다. 그러나 단일 Frame의 포인트 클라우드만으로는 보행자, 또는 거리가 멀리 떨어져 있어 크기가 작은 객체, 다른 사물에 가려져 있는 Hard 케이스의 객체 등과 같이 라이다의 레이저가 도달하기 힘든 객체를 인식하기가 매우 어렵다. 또한, 도로 상의 객체는 객체 고유의 동적 특성을 가지고 이동하기 때문에 이를 활용한다면 객체 인식 성능이 향상될 것으로 예상되나, 단일 Frame의 데이터만으로는 이를 활용하기 어렵다.

따라서 본 논문에서는 위와 같은 한계점을 극복하고자 먼저 객체 인식 모델에 입력되는 데이터를 시계열화하였다. 현시점은 기준으로 총 n 개의 다중 Frame

의 포인트 클라우드를 하나의 시계열 입력 데이터로 구성하였다. 이후 시계열 포인트 클라우드를 본 논문에서 제시한 객체 형상 특성 추출기에 순차 입력하여 각 Frame의 포인트 클라우드로부터 객체 형상 특성을 추출한다. 추출된 시계열 객체 형상 특성을 객체 동적 특성 추출기에 입력하여 객체 고유의 동적 특성을 추출하고, 마지막으로 현시점에서의 객체의 형상 특성과 동적 특성을 병합하여 객체 인식 헤드에 입력함으로써 최종 객체 정보를 출력하도록 설계하였다.

이를 통해 모든 평가지표에 대해 전반적으로 객체 인식 성능이 향상되는 것을 확인하였으며, 특히, 크기가 작은 객체나 Hard 케이스 객체에 대한 인식 성능이 크게 향상되었다. 모델의 성능 평가 결과는 객체 인식 분야의 대표적인 데이터 셋인 KITTI 데이터셋을 바탕으로 도출함으로 모델의 성능을 입증하였다.

주제어 : Autonomous Driving (자율주행), LiDAR Point-Cloud (라이다 포인트 클라우드), Deep Learning (딥러닝), 3D Object Detection (3차원 객체 인식), Time-series Data (시계열 데이터)



제 1 장 서론

1.1 연구 배경

자율주행 자동차에 대한 관심이 최근 몇 년간 전세계를 걸쳐 폭발적으로 증가하며 현대자동차, 메르세데스 벤츠와 같은 기존의 전통적인 자동차 OEM은 물론 구글, 테슬라와 같은 IT 산업을 기반으로 하는 거대 기업들도 앞다투어 자율주행 기술을 개발하고 있다 [1]. 하드웨어 기술을 바탕으로 하는 기존의 자동차 제조사가 아닌, IT 산업 기반의 기업들이 소프트웨어 기술력을 바탕으로 자율주행 기술 개발에 있어 선두를 달리고 있다는 점이 눈에 띠는데, 특히, 테슬라와 구글의 자회사인 웨이모가 자율주행 산업에서 확실한 두각을 나타내고 있다. 두 기업은 자율주행 실현이라는 목표는 동일하지만, 지향하는 바는 완전히 다르다. 테슬라는 자율주행을 위한 인지 센서로 카메라를 핵심으로 하며 고객들로부터 수집되는 방대한 양의 데이터와 인공지능 기술을 바탕으로 자율주행 기술을 구현한다. 반면, 웨이모는 카메라 외에도 라이다 (LiDAR), 레이더 (Radar) 등 다양한 자율주행 센서를 풀전하여 자율주행 기술을 구현하며, 최근에는 완전 무인 자율주행을 위해 C-ITS 및 고정밀 지도와 관련된 기술도 개발 중이다. 두 기업의 지향점을 바탕으로 자율주행 산업에서는 두 진영을 이루어 기술 개발이 진행되고 있으며 다양한 기업들이 기존의 첨단 운전자 보조 시스템 (ADAS, Advanced Driving Assistant System)을 뛰어넘는 기술 개발에 박차를 가하고 있다 [2].

한편, ADAS 및 자율주행 기술이 상용화 됨에 따라 해당 기능을 사용하여 주행하는 중에 사고가 발생하는 사례가 증가하고 있다. 테슬라는 카메라 센서를 오토파일럿이라 불리는 ADAS를 제공하는데, 운전자가 오토파일럿을 동작시킨 상태에서 차량의 센서가 정지해있는 전방 차량을 검지하지 못하여 충돌한

사례가 있다 [3]. 이러한 경우 사고에 대한 책임 소지가 불분명해지며 자율주행 차량 보급이 증가할수록 비슷한 사고 사례가 증가할 수 있을 것으로 보인다.

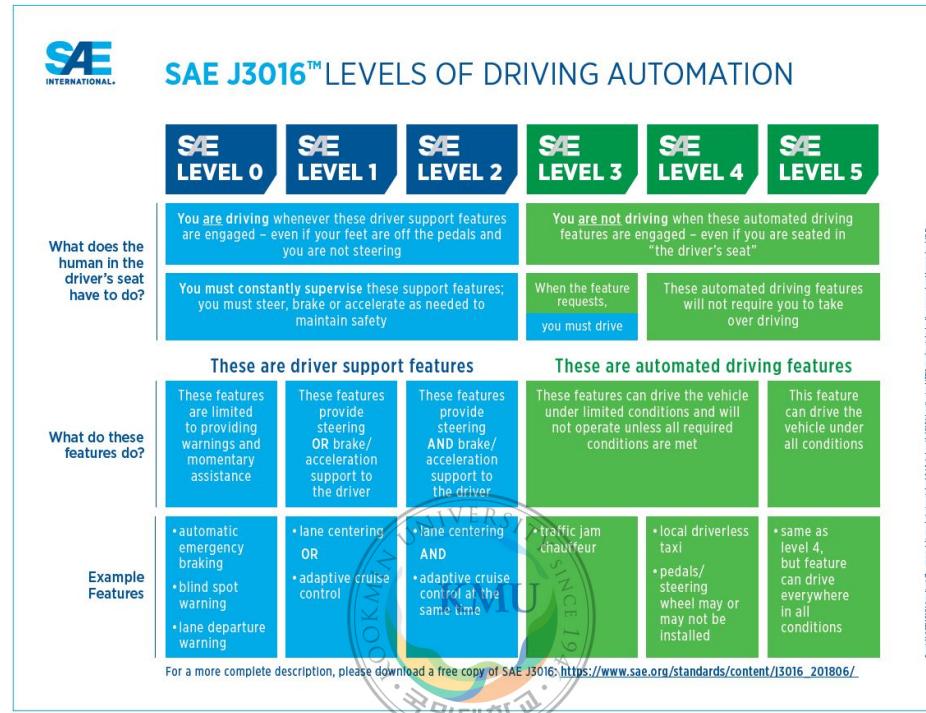


그림 1 SAE J3016 Levels of Driving Automation [4]

이에 미국 자동차공학회 (SAE, Society of Automotive Engineers)는 [그림 1]과 같이 자율주행 기술 수준에 따라 자율주행의 단계를 총 6단계로 구분하여 정의하였으며 [4] 현재 국제적으로 통용되고 있다. 현 시점에서 상용화되어 있는 대부분의 자율주행 기술은 자율주행 2단계에 해당하며 이러한 경우 아무리 기술의 수준이 높다고 하더라도 운전자가 항상 주행 상황을 주시하며 인지하고 있어야 하며, 사고 발생 시에도 사고에 대한 책임은 운전자에게 있다. 최근 자율주행 기술을 구현하고 있는 기업들은 2단계를 넘어서 3단계 이상의 기술을 상용화하여 운전자가 부분적으로 또는 완전히 주행 상황을 인지하지 않아도 주행 가능한 기술을 개발하고 있으며 자율주행 3단계 이상부터는 사고에 대한 책임

소지도 차량 제조사가 물을 가능성이 높다. SAE는 이와 같이 자율주행 단계를 6단계로 구분하여 명확하게 범위를 결정지음으로써 서로 다른 자율주행 기술 수준에 따른 혼란을 방지하고 책임 소지에 대한 기준을 마련하고자 하였다.

이처럼 다양한 기업 및 조직에서 자율구행 기술 구현을 위해 노력하고 있으며, 보다 안전하고 효율적인 자율주행 차량을 상용화하기 위해서는 여러 분야에서의 기술 발전이 여전히 필요한 상태이다. 본 논문에서는 이러한 흐름에 발맞추어 자율주행 기술, 구체적으로는 자율주행에 있어 필수적인 기술인 객체 인지 기술에 대한 연구를 기술하고자 한다.

1.1.1 객체 인지 관련 기술

자율주행의 필수 기술인 인지 (Perception) 기술은 크게 주변 객체의 종류 및 위치를 정확히 판단하는 객체 인식 (Object Detection) 기술과 인식된 객체를 단순히 한 Frame에서만 인식하는 것이 아닌 여러 Frame에서 동일 객체가 존재 할 시 이를 추적하는 객체 추적 (Object Tracking) 기술로 구성된다. 이에 대한 세부적인 설명은 아래와 같다.

(a) 객체 인식 (Object Detection)

객체 인식 기술은 자율주행 차량에 장착되어 있는 카메라, 라이다 (LiDAR), 레이더(Radar) 등의 다양한 자율주행 센서로부터 수집되는 데이터를 기반으로 차량 주행 경로 중에 있는 주변 차량, 보행자, 장애물 등을 인식하여 그 종류와 위치를 예측하는 기술이다. 최근 딥러닝 기술이 크게 발전함에 따라 객체 인식 성능도 크게 향상되었으며, 대부분의 자율주행 기술 개발 기업들이 딥러닝 기반의 객체 인식 기술을 자율주행 시스템에 도입하며 딥러닝 기반의 객체 인식 기술은 자율주행 구현에 있어 핵심 기술로 자리잡고 있다. 딥러닝 기술을 적용하기 위해서 카메라

이미지 기반의 객체 인식 [5], [6], 라이다 포인트 클라우드(Point-Cloud) 기반의 객체 인식 [7], [8] 또는 두 센서를 퓨전한 객체 인식 [9], [10] 기술들이 주를 이루고 있다. 객체 인식 기술의 경우 다양한 연구에서 팔목할 만한 성능을 보이고 있으며, 최근에는 악천후 조건, 빛 반사 등으로 인한 인식 성능 저하 상황에서의 인식 성능을 향상시키기 위한 연구가 활발히 진행되고 있다.

(b) 객체 추적 (Object Tracking)

객체 추적 기술은 자율주행 차량이 객체 인식 기술을 통해 인식한 객체 정보를 단순히 하나의 Frame에서만 인식하는 것이 아니라 인식된 객체들이 어느 방향으로 이동하고 있으며, 다음 Frame에서는 어디에 위치할 것인지를 추적해가는 기술이다. 이를 위해서는 현재 입력된 객체 정보를 기반으로 객체의 위치와 속도를 추정하고 미래 거동을 예측할 수 있어야 한다. 일반적으로 객체 추적 기술에서는 칼만 필터 [11]를 기반으로 하는 EKF (Extended Kalman Filter) [12], UKF (Unscented Kalman Filter) [13] 가 주로 사용된다. 최근에는 딥러닝 기술을 사용하여 객체를 추적하는 연구[14]도 활발히 진행되고 있다.

본 논문에서는 위와 같은 인지 기술 중 객체의 종류와 위치를 판단하는 객체 인식 기술을 개발하였다.

1.1.2 자율 주행을 위한 센서 시스템

자율주행차량이 객체 인식하기 위해서는 크게 카메라, 레이더, 라이다 등의 자율주행 센서를 주로 사용한다. 각각의 센서는 센서마다의 장단점이 존재하며 최근 동향으로는 각 센서의 장점은 극대화하고 단점은 상쇄시키기 위해 다양한 센서를 퓨전하여 사용하는 추세이다. 각 센서에 대한 설명은 아래와 같다.

(a) 카메라 (Camera)

카메라는 가장 대표적이면서 보편적인 자율주행 센서로, 이미 다양한 차량에 장착되어 ADAS를 위한 인지 센서로 사용되고 있다. 차량 및 보행자 인식, 차선 및 표지판 인식과 같은 ADAS 및 자율주행을 위한 용도뿐 아니라 360도 서라운드 모니터링 기능과 같이 운전자의 편의를 제공하는 기술 등 다양한 용도로 활용되고 있다. 카메라는 수동 센서 (Passive Sensor) 중 하나로, 물체에서 반사되는 반사광을 수용하여 2차원의 이미지로 변환함으로써 이미지를 생성하는 방식을 사용한다. 인간의 시각과 비슷한 특징을 가지기 때문에 이해가 쉽고 비용 측면에서도 장점을 가지기 때문에 다양한 기업에서 단순히 객체 인식만 하는 것이 아니라 고정밀지도 생성, 차량 위치 인식 등 다양한 기능 수행을 위해 카메라에 대한 의존도를 높이고 있다. 그러나 야간 주행, 눈, 비 등에 의한 악천후 조건, 직사광선 등의 원인으로 안정적인 객체 인식을 위한 충분한 양의 빛이 들어오지 못하는 상황은 언제든 발생할 수 있으며, 이 때 카메라 기반의 객체 인식 성능은 급격히 감소하게 된다. 따라서 이러한 경우를 대비하여 빛 반사 조건에 강건한 다른 종류의 자율주행 센서가 필요하다.

(b) 레이더 (Radar)

레이더는 카메라와 함께 가장 보편적이면서도 저렴한 가격으로 인해 오래 전부터 많은 차량에 장착되어 ACC (Adaptive Cruise Control), AEB (Autonomous Emergency Braking) 등의 일반적인 ADAS의 기본 센서로 사용되어 왔다. 레이더는 전자기파를 방출하여 물체의 표면에 반사되어 돌아오는 전자기파를 수용하여 객체를 감지하는 대표적인 능동 센서 (Active Sensor)이다. 전자기파가 방출되어 다시 돌아오기까지의 시간 (ToF, Time of Flight)을 기반으로 레이더는 물체까지의 거리를 정확하게

측정할 수 있다. 또한, 도플러 효과 (Doppler Effect)를 바탕으로 방출되는 전자기파와 수용되는 전자기파의 주파수 변화를 측정하여 동적 객체의 속도 측정도 가능하다. 레이더는 긴 파장의 전자기파를 사용하기 때문에 감지 가능 거리가 가장 길고 대부분의 기상 조건에 강건하다는 장점을 가지나, 공간 분해능이 매우 낮기 때문에 횡방향으로 나열되어 있는 여러 객체들을 분간하거나 감지된 객체의 형상을 판단하는 것은 어렵다.

(c) 라이다 (LiDAR)

라이다는 자율주행 기술이 주목을 받기 시작하면서 크게 각광받고 있는 능동 센서 중 하나이다. 라이다는 매우 짧은 파장의 레이저를 방출하고, 해당 빛이 돌아오는 데까지의 소요 시간 (ToF) 측정을 통해 물체까지의 거리를 매우 정밀하게 측정할 수 있다. 현재 자율주행 차량 개발에 있어 주로 사용되는 라이다 타입은 **스캐닝** (Scanning) 타입이다. 스캐닝 라이다는 차량의 루프에 장착되어 Transmitter가 빠르게 360도로 회전하면서 레이저를 방출하며 초당 수천 개의 빛을 받아 포인트 클라우드 데이터를 생성한다. 이러한 방식으로 라이다는 주변 환경에 대한 3차원의 고정밀 포인트 클라우드 맵을 생성할 수 있으며 이로부터 차량, 보행자, 사이클리스트 등의 객체를 정밀하게 인식할 수 있다. 라이다의 가장 큰 단점은 세 가지 센서 중 가장 가격이 높다는 것이다. 최근 Ouster 사의 라이다 등 비교적 저렴한 가격대의 라이다가 출시되고 있지만 여전히 천만원을 웃도는 가격이며 이러한 가격대의 센서가 현재 상용화되고 있는 차량에 장착되기는 불가능에 가깝다. 또한, 라이다는 빛을 방출해야 하므로 주변이 가려지지 않아야 하기 때문에 차량에 장착되어 패키징을 하기가 어렵다는 단점도 존재한다.

본 연구에서는 다양한 자율주행 센서 중에서 라이다 (LiDAR)를 활용한 객체

인식 기술에 대해 연구하였다. 객체의 3차원 정보를 포함하고 있는 라이다 포인트 클라우드를 활용하여 3차원 공간 상에서의 객체의 종류 및 위치를 인식하는 객체 인식 모델을 개발하였다.

1.2 연구 방향

본 논문에서는 위와 같이 현재 활발히 연구되고 있는 자율주행의 다양한 요소 기술 중 딥러닝 기반의 객체 인식 기술 개발을 목표로 한다. 특히, 실제 자율주행 상황에서 주변 객체의 종류, 위치 및 크기를 정확하게 판단하여 안전한 자율주행이 가능하도록 2차원이 아닌 3차원 공간 상에서의 객체 인식 기술을 개발하였다. 또한, 라이다 센서를 기반으로 포인트 클라우드를 딥러닝 모델의 입력 데이터로 활용하여 포인트 클라우드 상 객체의 종류 및 3차원 공간 상에서의 위치 및 크기를 예측하는 기술을 개발하였다.

1.3 기존 연구의 한계점

객체 인식 기술은 2차원 평면 상에서의 객체 인식으로부터 시작되었지만, 자율주행 상황에서는 3차원 공간 상에서 객체를 인식하는 것이 매우 필수적이다. 자율주행 차량이 주변 상황을 인지하고 주변 물체와 충돌하지 않으며 안전하게 주행할 수 있는 경로를 생성하기 위해서는 객체의 종류 및 위치와 더불어 높이, 폭, 길이와 같은 3차원 공간 상에서의 크기를 정밀하게 측정할 수 있어야 하기 때문이다. 따라서 자율주행 차량에 적용되기 위한 3차원 공간 상에서의 객체 인식 기술에 대한 연구가 최근 활발히 진행되고 있다 [15]–[17].

이러한 대부분의 딥러닝 기반 3차원 객체 인식 모델들은 단일 Frame의 데이터를 입력 데이터로 사용한다. 그러나 이러한 경우 [그림 2]와 같이 보행자, 사이클리스트 등의 크기가 작은 객체나 다른 사물에 의해 가려지거나 센서의

시야각을 벗어나 잘린 Hard 케이스의 객체는 인식하기가 매우 어렵다.

또한, 자율주행 상황을 가정해보면 차량이 주행하는 동안 모든 자율주행 센서로부터의 데이터는 시간 순으로 입력된다. 따라서 현재 시점의 Frame에 존재하는 객체는 가까운 과거 시점의 Frame 상에도 존재할 가능성이 높으며, 현재 시점에서 인식하기 어려운 객체를 좌거 시점의 동일한 객체 정보로부터 보완받아 인식률을 높일 수 있다. 그러나 단일 Frame의 데이터만 사용할 경우 이렇게 과거로부터의 객체 정보를 사용할 수 없다는 한계점이 존재한다.

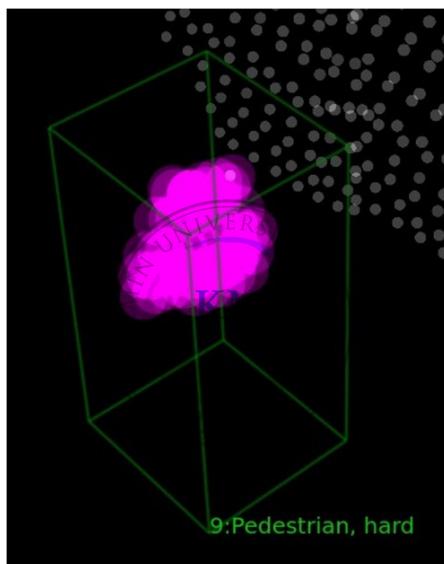


그림 2 An example of a small size and hard case object

1.4 연구 목표

본 논문에서는 객체 인식 모델에 입력되는 데이터를 시계열 화합으로써 위와 같은 한계점을 극복하고자 한다. 시계열 데이터를 활용하여 현시점에서 인식이 어려운 객체를 과거 시점의 객체 정보로부터 보완 받아 객체 인식 정확도를 높이는 것을 목표로 한다. 또한, 시계열 데이터로부터 객체의 시공간적 특성 추출을

통해 객체 고유의 동적 특성도 객체 인식에 활용하였다.

구체적으로는 먼저 현재 시점은 기준으로 과거 시점까지 총 n개의 다음 Frame의 포인트 클라우드를 하나의 시계열 입력 데이터로 구성한다. 이후 시계열 포인트 클라우드를 형상 특성 추출기 (Structural Feature Extractor)에 순차 입력하여 각 Frame의 포인트 클라우드로부터 객체의 형상 특성을 추출한다. 그리고 추출된 시계열 객체 형상 특성을 동적 특성 추출기 (Dynamic Feature Extractor)에 입력하여 객체가 시간의 흐름에 따른 이동에 의한 동적 특성을 추출하고, 마지막으로 이러한 특성을 디텍션 헤드 (Detection Head)에 입력하여 최종 객체의 정보를 출력한다.

이를 통해 본 연구에서는 전반적인 객체 인식 성능 향상과 동시에 기존의 단일 Frame만을 사용하였을 때 발생하는 한계점인 크기가 작은 객체나 Hard 케이스의 객체 인식 성능 향상을 목표로 한다.

1.5 논문 구성

본 논문은 다음과 같이 구성된다. 먼저 섹션 2에서는 객체 인식 기술과 본 논문에서 사용하고자 하는 기술에 대한 동향에 대해 기술한다. 섹션 3에서는 본 연구에서 해결하고자 하는 문제를 정의하고, 이를 위해 개발한 객체 형상 특성 추출기 및 동적 특성 추출기에 대해 설명한다. 또한, 연산 속도를 개선하기 위한 방안을 제시하며 결과를 제시하기 위한 실험 설정에 관해 기술한다. 섹션 4는 본 연구에서 개발한 모델에 대한 실험 결과를 나타낸다. 정량적 결과와 정성적 결과를 각각 나타내었으며, 개발한 모델의 성능을 뒷받침하기 위한 Ablation Study 결과도 기술하였다. 마지막으로 섹션 5에서는 본 연구에 대한 결론을 기술하고 본 연구에서 더 나아가기 위한 향후 과제에 대해 제시한다.

제 2 장 연구 동향

2.1 인공지능 및 딥러닝 기술 발전

최근 객체 인식 기술의 성능이 크게 향상될 수 있었던 것은 인공지능, 특히 그 중에서도 딥러닝 기술이 발전하면서부터이다. 딥러닝은 머신러닝의 한 분야로써 어떠한 과업을 해결할 때 기존의 규칙 기반 알고리즘과 달리 모델 기반의 알고리즘을 활용하여 모델이 특정 과업을 수행하기 위해 데이터를 바탕으로 직접 학습하는 과정을 거치는 기술이다. 학습이 진행되는 동안 최적화 알고리즘을 통해 딥러닝 모델은 주어진 과업을 수행하기 위한 최적의 파라미터를 탐색하게 된다.

딥러닝 기술이 급격하게 성장하게 된 요인은 크게 세 가지로 볼 수 있는데, 1. 하드웨어 (GPU)의 발전, 2. 빅 데이터, 3. 알고리즘의 발전이 그것이다.

2.1.1 하드웨어 (GPU)의 발전

GPGPU는 General-Purpose computing on Graphics Processing Units의 약어로, 컴퓨터 그래픽스의 용도로만 사용되던 그래픽 처리 장치 (GPU)를 응용 프로그램을 위한 연산 장치로 사용하게 되면서 생긴 용어이다. GPU의 경우 그래픽 송출을 위해 일반적으로 수천 개의 Core를 가지기 때문에 이러한 수많은 Core를 활용하여 연산을 병렬화할 경우 특정 과업의 연산 속도를 크게 향상시킬 수 있다. 대부분의 연산 과정이 많은 수의 행렬 곱으로 이루어진 딥러닝 알고리즘 또한 GPU Core 기반 병렬 처리가 가능하며 이를 통해 딥러닝 알고리즘의 학습 속도를 극대화할 수 있었고 다양한 딥러닝 모델이 등장하게 되었다.

2.1.2 빅 데이터

딥러닝 기술 성장의 또 다른 요인은 빅 데이터의 등장이다. 딥러닝은 현재 주어진 데이터를 기반으로 전체 데이터의 분포를 학습하는 방식으로 진행되기 때문에 전체 데이터를 표현하기 위한 대량의 데이터가 필수적이다. 이를 위해 오픈소스로 공개된 가장 대표적인 데이터셋은 ImageNet 데이터셋이다[18]. ImageNet 데이터셋은 1000개의 클래스에 속하는 약 1400만 개의 이미지로 구성되어 있는 데이터셋이며, 해당 데이터셋으로 이미지의 클래스를 분류하는 챌린지 [19]가 2017년까지 매년 개최되었다. 해당 챌린지에서 딥러닝 모델의 대표적인 모델인 VGGNet [20], ResNet [21]이 등이 최초로 발표되며 딥러닝 기술 개발에 촉진제 역할을 하였다.

자율주행 분야에서는 차량이 주행하면서 각종 자율주행 센서로부터 수집된 데이터를 기반으로 배포된 데이터셋이 주를 이룬다. 가장 대표적으로 KITTI 데이터셋 [22], nuScenes 데이터셋 [23], Waymo 데이터셋 [24] 등이 있다.

KITTI 데이터셋은 세 가지 데이터셋 중 가장 역사가 깊은 데이터셋으로 객체 인식 (Object Detection, [그림 3]), 객체 추적 (Object Tracking) 및 위치 인식 (Localization) 등을 위한 데이터가 수집되어 있다. 카메라 이미지, 라이다 포인트 클라우드, GPU 및 IMU 센서 기반 위치 데이터로 구성되어 있으며 약 이십만 개 이상의 객체가 레이블링되어 있다.

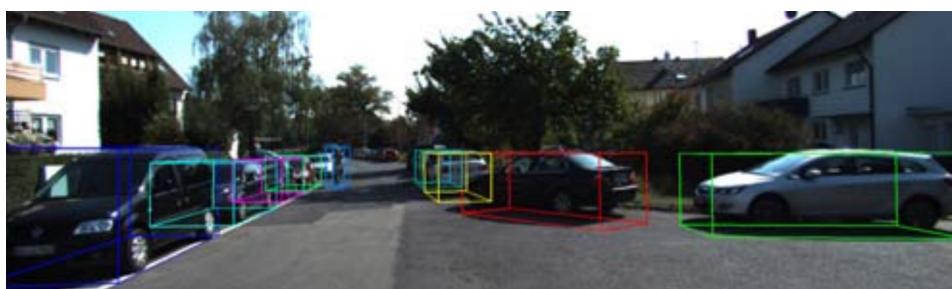


그림 3 An example of 3D object labels of KITTI dataset [22]

nuScenes 데이터셋은 자율주행에서 사용되는 모든 센서의 데이터가 수집되어 있는 최초의 데이터셋으로, [그림 4]과 같이 6대의 카메라, 5대의 레이더 및 1대의 라이다로부터 수집된 데이터가 포함되어 있다. 데이터셋은 23개의 클래스를 포함하는 약 20초 분량의 장면이 1000개로 구성되어 있으며 KITTI 데이터셋보다 약 100배 많은 수의 데이터가 존재한다.

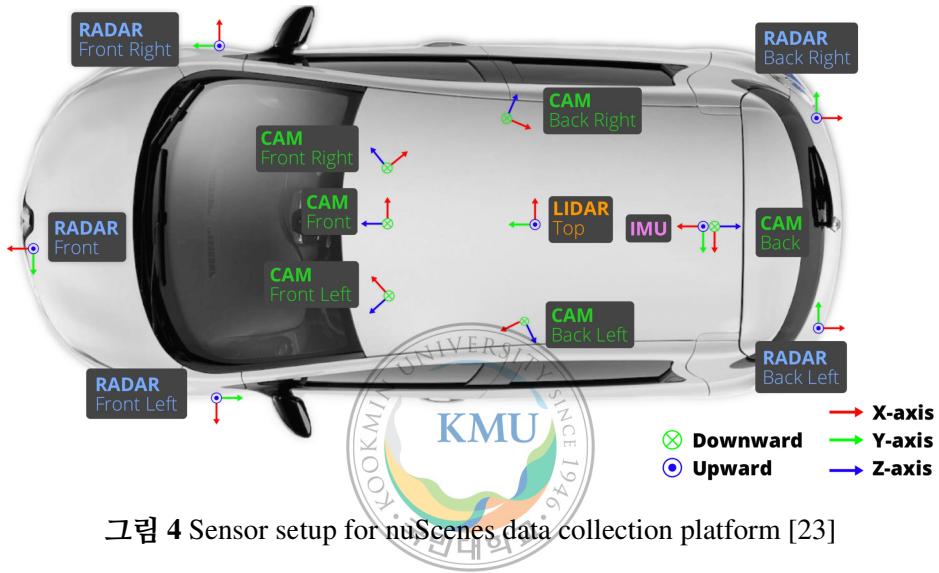


그림 4 Sensor setup for nuScenes data collection platform [23]

Waymo 데이터셋은 세 가지 데이터셋 중 데이터의 수가 가장 많으며, 20초 분량의 장면이 1150개로 구성되어 있다. KITTI 데이터셋과 동일하게 카메라와 라이다로부터 데이터가 수집되었고, 도심지 및 교외지 등 다양한 주행 환경에서 수집된 데이터가 존재한다.

본 논문에서는 위와 같은 다양한 데이터셋 중 KITTI 데이터셋과 nuScenes 데이터셋을 활용하여 개발된 모델의 성능을 평가하였다.

2.1.3 알고리즘의 발전

마지막으로 다양한 딥러닝 알고리즘의 등장과 함께 딥러닝 기술이 급격히 성장하게 되었다. 딥러닝 기술에서 사용되는 심층 신경망(Deep Neural Networks,

DNN)은 1990년 대에도 존재하였다. 그러나 당시에 심층 신경망을 학습시키던 알고리즘의 경우 심층 신경망의 깊이가 깊어질 경우 신경망의 하위 층으로 갈수록 그레디언트의 값이 점점 작아지는 그레디언트 소실(Vanishing Gradient) 문제가 발생하며 모델의 성능이 개선되지 못했고, 이러한 이유로 2000년 대 초까지 딥러닝 기술이 각광 받지 못했다 [25]. 그레디언트가 소실되는 가장 대표적인 원인 중 하나는 시그모이드 함수가 당시 심층 신경망의 활성화 함수로 사용되었기 때문이다. [그림 5]와 같이 시그모이드 함수는 입력의 절댓값이 커질수록 0 또는 1로 수렴하며 기울기가 0에 매우 가까워진다. 따라서 시그모이드 함수로 인해 신경망의 그레디언트가 0에 가까운 값을 가지며 그레디언트가 소실되는 문제가 발생한다.

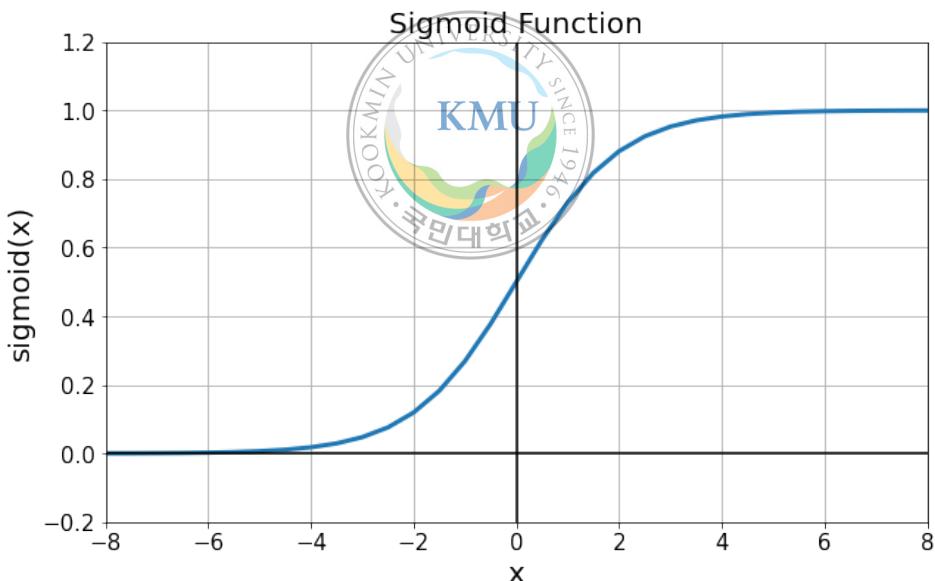


그림 5 Sigmoid Activation Function

이렇게 값이 증가함에 따라 그레디언트가 소실되는 문제를 해소하기 위해 등장한 활성화 함수가 ReLU (Rectified Linear Unit) 함수이다. ReLU 함수는 [그림 6]와 같이 함수의 입력값이 양수인 경우 값이 증가하더라도 기울기는 항상 양의

상수값을 가지기 때문에 그레디언트가 소실되는 문제를 해소할 수 있다. 따라서 ReLU 함수를 활성화 함수로 사용함으로써 깊은 층의 심층 신경망이 안정적으로 학습할 수 있게 되어 다양한 딥러닝 알고리즘이 연구될 수 있었다. 이후 ReLU 함수가 함수 입력값이 음수인 경우 그레디언트가 0이 된다는 문제점을 해결하기 위해 LeakyReLU [26], ELU [27], SELU [28] 등 ReLU 함수 기반의 다양한 활성화 함수가 제시되며 현재 많은 연구에서 활용되고 있다. 본 논문에서도 위와 같은 이점이 있는 ReLU 함수를 모델의 활성화 함수로 사용하였다.

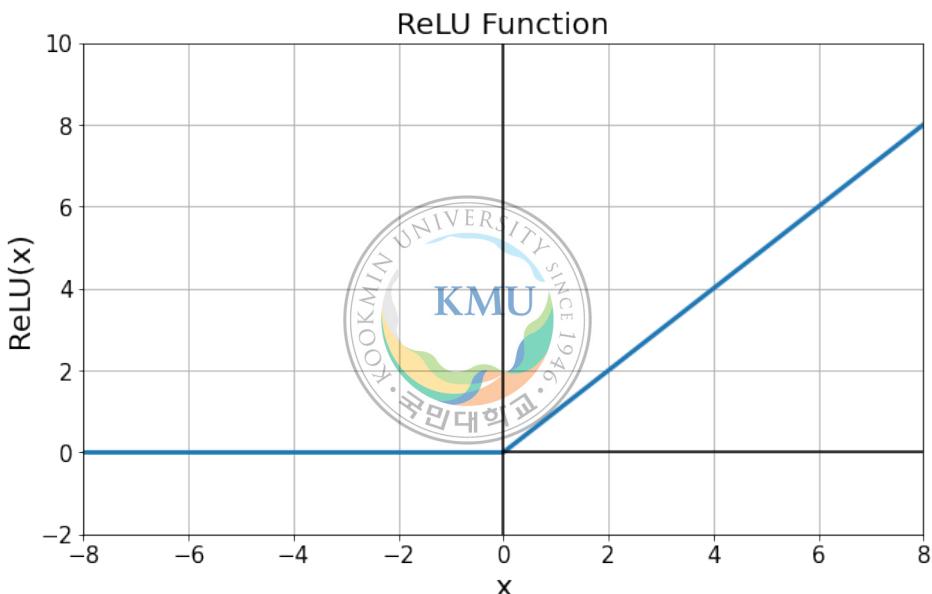


그림 6 Sigmoid Activation Function

또한, 딥러닝 모델의 최적의 파라미터를 탐색하기 위한 다양한 최적화 알고리즘도 등장하였다. 일반적으로 딥러닝 모델의 파라미터를 최적화할 때에는 경사 하강법 (Gradient Descent, GD)을 바탕으로 한다. 경사 하강법은 모델의 예측 값과 실제 정답값의 차이를 나타내는 손실 함수를 최소화하는 방향으로 반복해서 모델의 파라미터를 조정해나가는 방법이다 [25]. 손실 함수로부터 모델의 파라미터에 대한 그레디언트 (경사)를 계산하고, 그레디언트가 감소하는 방향

(경사가 하강하는 방향)으로 모델의 파라미터를 업데이트한다. 이 때 학습률 (Learning Rate)을 하이퍼파라미터로 하여 그레디언트와 곱함으로써 파라미터가 업데이트되는 정도를 조절한다. 딥러닝 기술에서 이러한 방식으로 모델의 파라미터를 업데이트하는 객체를 옵티마이저라 하는데, 경사 하강법을 기반으로 다양한 옵티마이저가 개발되었다.

그 중에서 먼저 모멘텀 최적화 [29] 옵티마이저는 현재 상태의 그레디언트에 추가적으로 이전 그레디언트들을 누적한 값을 나타내는 모멘텀을 사용하여 파라미터를 업데이트하는 방식을 사용한다. 이러한 경우 모멘텀이 관성처럼 작용하여 파라미터 수렴 속도를 높여주고, 파라미터가 지역 최적점에 빠지는 것을 어느 수준 방지해줄 수 있다.

AdaGrad [30] 알고리즘은 적응적 학습률 (Adaptive Learning Rate)을 도입하여 학습이 진행되는 동안 학습률을 점진적으로 감소시키는 방법을 사용한다. 특히, 그라디언트의 제곱을 누적시킨 값을 기준으로 학습률을 감소시키므로 경사가 완만한 차원보다 경사가 급한 차원의 학습률이 보다 빠르게 감소하게 되고 이를 통해 단순히 경사가 급한 방향이 아닌 전역 최적점 방향으로 학습이 진행되도록 유도한다.

RMSProp [31]은 AdaGrad 옵티마이저의 가장 큰 문제인 학습률이 너무 빠르게 감소하여 파라미터가 전역 최적점에 도달하기도 전에 학습이 더 이상 진행되지 않는다는 것을 보완하기 위한 옵티마이저이다. 이를 위해 RMSProp은 감쇠율을 적용하여 학습 과정 전체의 그레디언트가 아닌 최근의 학습 과정에서 계산된 그레디언트들만을 누적시키는 방식을 사용한다.

마지막으로 Adam [32] 옵티마이저는 모멘텀 최적화와 RMSProp 옵티마이저를 하나의 옵티마이저로 합친 알고리즘으로, 모멘텀을 통해 관성적으로 모델의 파라미터를 업데이트하며 옵티마이저의 학습률을 감소시켜 파라미터가 전역

최적점 방향으로 수렴할 수 있도록 한다. 이러한 장점으로 인해 최근 개발된 딥러닝 모델들은 주로 Adam 옵티마이저를 많이 사용하는 경향을 보이며, 본 논문에서도 Adam 옵티마이저를 통해 개발한 모델의 파라미터를 최적화시켰다.

위와 같은 알고리즘의 발전 덕분에 고도화된 심층 신경망의 학습이 가능해지며 다양한 딥러닝 모델들이 등장하게 되었다. 대표적으로 MLP (Multi-Layer Perceptron), CNN (Convolutional Neural Networks) [33], RNN (Recurrent Neural Networks) [34] 등이 있다. MLP는 선형 연산과 비선형 연산이 결합되어 있는 퍼셉트론을 다층 구조로 쌓은 신경망으로, 딥러닝 모델에서 고전적이면서도 현재 까지 많이 사용되고 있다. CNN의 경우 MLP와 비슷한 과정의 연산을 입력 데이터 전체에 걸쳐 연산하는 것이 아닌, 국부적으로 순회하며 (Convolve) 연산하는 신경망으로, 이미지, 영상과 같은 특정 정보가 국소적으로 존재하는 데이터를 학습하는데 주로 사용된다. RNN은 순환 신경망으로써 데이터를 순차적으로 입력 받아 학습하며 메모리 셀을 통해 이전에 입력된 데이터에 대한 정보를 기억하고 이를 다음 입력 데이터의 예측에 보완하는 형태로 동작한다. 과거의 정보를 기억하고 이를 현시점의 예측에 활용할 수 있다는 점으로 인해 RNN은 시계열 데이터 학습에 주로 사용된다. 이러한 딥러닝 모델들은 현재 많은 연구에서 활용되고 있으며 본 연구에서도 해당 모델들을 기반으로 연구를 진행하였다.

2.2 객체 인식 기술

객체 인식 기술은 이미지나 포인트 클라우드와 같은 센서 데이터로부터 알고리즘을 통해 해당 데이터에 존재하는 객체들을 인식하는 기술이다. 객체를 인식할 때에는 객체의 종류를 나타내는 클래스와 객체의 데이터 상의 위치 정보에 대한 값을 예측하게 된다. 평면 상에서 객체의 위치를 파악하는 2D 객체 인식 기술과 공간 상에서 객체의 위치를 파악하는 3D 객체 인식 기술로 나눌 수

있으며, 최근에는 자율주행 기술 분야에서는 3D 객체 인식 기술에 대한 연구가 활발히 진행되고 있다.

2.2.1 2D 객체 인식 기술

2D 객체 인식 기술은 평면 상에서 객체를 인식하며 데이터가 입력되었을 때, 객체의 클래스와 객체의 위치 정보로 객체 중심점의 x, y 좌표와 객체의 높이 h 및 폭 w 를 예측하게 된다. 2차원 평면 상의 객체를 인식하는 기술이기 때문에 주로 카메라 이미지를 입력 데이터로 사용한다. 최근에는 딥러닝을 기반으로 하는 객체 인식 모델이 주를 이루고 있고, 이러한 모델들은 2-stage 방식과 1-stage 방식의 알고리즘으로 구분할 수 있다.

2-stage 방식의 객체 인식 모델은 먼저 첫번째 stage에서 객체가 존재할 확률이 높은 구역인 ROI (Region Of Interest)를 추려내고 두번째 stage에서 추려진 ROI 내에서 객체의 정확한 위치를 예측하는 방법으로 객체 인식을 수행한다. R-CNN 계열의 객체 인식 모델들 [6], [35]–[37]이 대표적이다.

1-stage 방식의 객체 인식 모델들은 2-stage 방식에서 두 stage로 나뉘어 예측이 진행되면서 발생하는 오버로드를 줄이기 위해 제안된 방식이며 ROI를 추리는 stage 없이 하나의 stage만으로 객체 인식을 수행한다. 이를 통해 2-stage를 사용할 때보다 훨씬 더 빠른 연산 속도를 보임과 동시에 객체 인식에 대해서도 준수한 성능을 보였다. 자율주행 분야에서는 모델의 정확도뿐 아니라 모델 추론의 실시간 성 또한 보장되어야 하므로 2-stage보다 1-stage 방식의 딥러닝 모델이 선호된다. 1-stage 방식의 객체 인식 모델에는 대표적으로 YOLO 계열의 모델 [5], [38], [39]과 SSD 계열의 모델 [40]이 있다.

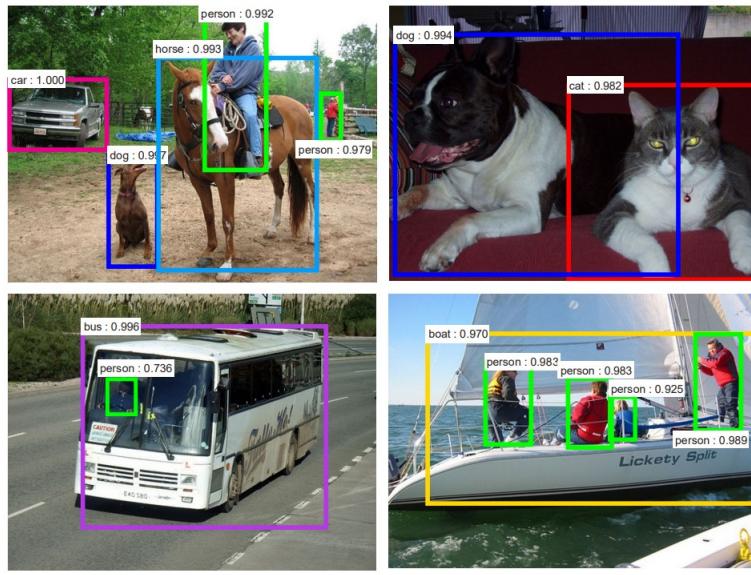


그림 7 2D Object Detection Results of Faster R-CNN [6]

2.2.2 3D 객체 인식 기술

3D 객체 인식 기술은 평면 상이 아닌 공간 상에서 객체의 위치를 예측하는 기술로, 객체의 클래스와 더불어 객체의 중심좌표 x, y, z , 객체의 높이 h , 폭 w , 길이 l 및 객체의 회전각 θ 등 객체의 3차원 상 위치 정보를 출력하게 된다. 3차원 상의 객체 정보를 인식해야 하므로 입력 데이터로 카메라 이미지뿐 아니라 3차원 공간 정보를 담고 있는 라이다 포인트 클라우드도 자주 사용되며, 최근에는 각종 센서 데이터를 퓨전하여 객체 인식을 수행하는 연구도 활발하다. 본 논문에서는 라이다 포인트 클라우드를 입력으로 하는 딥러닝 기반의 3D 객체 인식 모델을 개발하였다.

포인트 클라우드 상에서 3차원 객체 인식을 수행하는 연구는 최근 자율주행 기술 개발이 각광 받음에 따라 매우 활발히 진행되고 있다. 이러한 연구는 포인트 클라우드로부터 특성을 뽑아내는 방식에 따라 분류될 수 있다.

먼저 포인트 기반의 방식을 사용하는 모델 [7], [41]–[48]은 포인트 클라우드

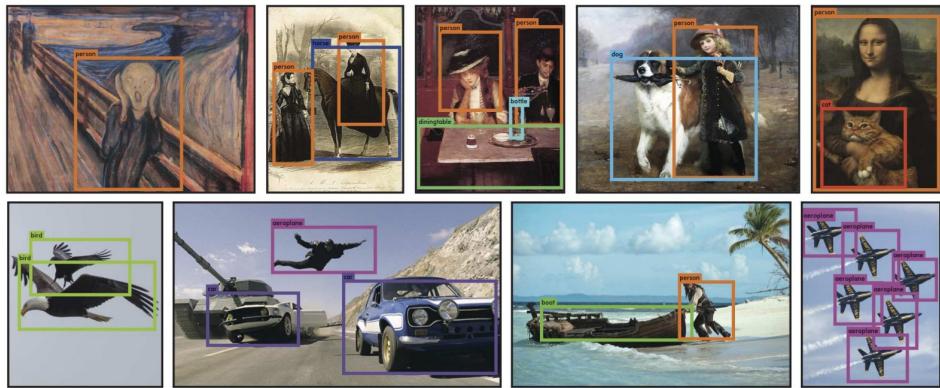


그림 8 2D Object Detection Results of YOLO [38]

원시 데이터를 추가적인 데이터 처리 없이 있는 그대로 모델에 입력한다. 이러한 경우 포인트 클라우드의 모든 정보를 손실 없이 사용할 수 있다는 장점이 있지만, 포인트의 수가 너무 많아 연산량이 크게 증가한다는 문제가 발생한다.

포인트 기반의 모델과 다르게 Voxel 기반의 방식을 사용하는 모델들 [49]–[60]은 먼저 3차원 공간의 그리드 상에서 Voxel이나 Pillar (기둥)과 같은 부피 요소를 정의한다. 그리고 포인트 클라우드의 각 포인트를 각자의 상대적 위치에 따라 각자 속하는 부피 요소에 할당하여 포인트 클라우드를 Voxel 형태로 재정의한 후 이러한 Voxel 요소를 모델의 입력으로 사용한다. Voxel 기반의 방식을 사용하면, 포인트 전체를 사용하는 것보다 모델에 입력되는 전체 요소의 수가 줄게 되므로 메모리나 연산량 측면에서 큰 이점이 있다.

다음으로 Frustum 기반의 방식으로 특성을 추출하는 모델들이 있다 [61]–[66]. 이러한 모델들은 2차원 상에서 제안된 객체 인식 결과를 활용하여 포인트 클라우드를 군집화한 후 최종적으로 3차원 객체 인식을 수행한다. 2 차원 상에서 객체 인식 결과는 포인트 클라우드가 아닌 카메라 이미지로부터 인식된 결과를 사용한다. 카메라로부터 인식된 2차원의 객체 정보를 카메라와 라이다의 Calibration을 활용하여 Frustum이라 불리는 요소로 3차원 공간 상에

투영시킨다. 그리고 최종적으로 Frustum을 기반으로 포인트 클라우드 상의 3 차원 객체 인식 결과를 도출하게 된다.

마지막으로 포인트 클라우드의 특성을 Front View (FV, 정면도) [67], [68]나 Bird Eye View (BEV, 조감도) [10], [49], [52], [69]–[72]와 같이 특정한 시야에서 추출하는 방식이 있다. 이러한 방식을 사용하면, 포인트 클라우드를 3차원이 아닌 2차원 상에서 바라보기 때문에 차원이 감소되어 연산량 측면에서 이점이 있고, 2차원 특성을 추출하여 CNN 모델에 곧바로 입력할 수 있다는 장점이 있다. 그러나 포인트 클라우드를 특정한 시야에서 표현하기 위해 수작업을 통한 인코딩 작업이 필요하고 이 과정에서 포인트 클라우드의 정보 손실을 야기할 수 있다는 단점이 있다.

본 논문에서는 메모리 및 연산량의 이점을 활용하면서도 포인트 클라우드로부터 특성을 효율적으로 추출할 수 있는 Voxel 기반의 방식을 활용하여 연구를 진행하였다.

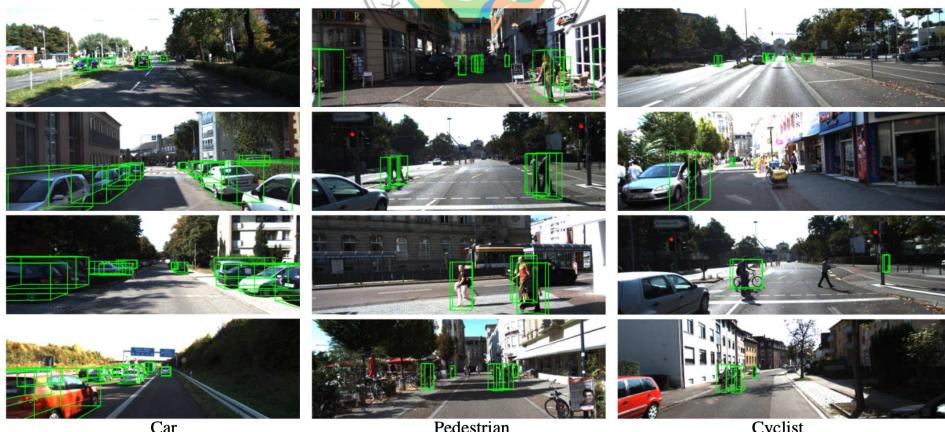


그림 9 3D Object Detection Results of VoxelNet [50]

2.2.3 다중 Frame 기반 3D 객체 인식 기술

위에서 지금까지 소개한 3D 객체 인식 모델은 단일 Frame의 포인트 클라우드를 입력으로 하는 모델이다. 최근에는 단일 Frame이 아닌 다중 Frame의 포인트 클라우드를 하나의 시계열 데이터로 간주하여 3D 객체 인식에 활용하는 연구가 증가하고 있다.

[73]은 포인트 클라우드 시계열 데이터를 BEV 이미지로 변환하고 이를 병합 (Concatenate)한 후 공간 축과 시간 축 방향으로 3D CNN 모델에 입력함으로써 객체 인식을 수행한다.

[74]는 먼저 포인트 클라우드의 시공간적 특성을 추출하기 위해 각각의 포인트 클라우드를 그래프 기반 Message Passing 네트워크에 입력하여 인코딩 한다. 이후 포인트 클라우드 시계열 데이터의 시공간적 정보를 축약하기 위해 Attention 기반의 Spatio-Temporal Transformer GRU 모델을 사용한다.

[75]에서는 오프라인 객체 인식 파이프라인을 제안하였고 10초 이상의 긴 포인트 클라우드 시계열 데이터를 활용하고자 하였다. 제안된 파이프라인은 상대적으로 긴 시계열 데이터를 사용하므로 실시간 성을 보장하기 어려우며 오프라인 상태에서 활용하기 적합하다. 따라서 해당 연구에서는 시계열 데이터로부터 다양한 시야의 객체 정보를 활용함으로써 3차원 객체 인식을 위한 정확한 레이블링 자동화 (Auto Labeling)를 목적으로 한다고 제안한다.

[76]는 U-Net 구조의 3D CNN 활용하여 각 Frame의 포인트 클라우드로부터 공간적 특성을 추출한다. 그리고 추출된 시계열 공간적 특성을 LSTM 모듈에 입력하여 3D 객체 인식을 수행한다. 이는 RNN 계열의 모델을 활용한다는 점에서 본 연구에서 제안한 모델과 유사점이 존재한다.

[77]은 시계열 데이터로부터 객체를 인식하기 위해 ConvLSTM [78] 모델을 활용한다. 해당 연구는 본 연구와 비슷하게 포인트 클라우드의 특성을 추출하

기 위해 PointPillars [51]를 차용하였고, PointPillars의 Detection Head 이전에 ConvLSTM을 삽입하여 시계열 특성을 추출한다. [77]이 본 연구와 비슷한 구조의 모델을 제안하였지만, 객체 인식을 위해 시계열 특성을 활용하는 방법에서 명확한 차이를 보인다. [77]는 ConvLSTM에서 추출된 특성만을 사용한 반면에, 본 연구에서는 객체의 형상 특성과 동적 특성을 분리하여 학습하기 위해 ConvLSTM에서 추출된 특성과 PointPillars에서 추출된 특성을 모두 사용한다. 이렇게 함으로써 객체 인식 성능이 [77]보다 본 논문에서의 모델이 좋음을 확인하였으며 이를 4에서 보였다.

2.3 Attention 메커니즘

Attention 메커니즘은 본래 자연어 처리 (Natural Language Processing, NLP) 분야에서 처음으로 등장하였다. [79]은 기존의 기계 번역에서 주로 사용하였던 LSTM [80]에 Attention 메커니즘을 적용하였고, 이를 통해 문장을 번역할 때 매 단어마다 전체 문장 중 어떤 단어가 가장 영향을 많이 미치는지에 대한 확률값 계산을 통해 번역 성능을 높였다.

Attention 메커니즘의 가장 대표적인 연구는 Transformer [81]이다. Transformer는 기본적으로 Self-Attention 메커니즘을 기반으로 하며, 이를 통해 기존의 기계 번역에서 주로 사용되는 LSTM을 대체하며 LSTM의 데이터 순차 입력에 의한 연산 속도 저하 문제를 해결할 수 있었다. Self-Attention 메커니즘은 입력 데이터를 통해 어떠한 과업을 수행할 때 하나의 데이터 내에서 어떤 부분이 가장 영향을 많이 주는지를 Key, Query, Value의 관계를 통해 연산하고 이를 통해 Attention Map을 생성한다. Attention Map을 통해 과업을 수행하기 위해 중요한 부분을 강조하는 방식으로 Attention 메커니즘을 적용한다. 그러나 이러한 Attention Map은 Transformer의 단점이 되기도

하는데, Key, Query, Value를 통한 Attention Map 생성 과정에서 메모리가 많이 필요하고, 문장이 길어짐에 따라 이는 더욱 늘어나게 된다. 이로 인해 특성 맵의 크기가 큰 이미지나 포인트 클라우드 데이터에는 Self-Attention 메커니즘을 적용하기가 어렵고, 특히나 자율주행 차량은 일반적으로 임베디드 시스템 내에서 동작하기 때문에 많은 메모리를 필요로 하는 모델은 사용하기 어렵다.

따라서 본 연구에서는 이를 고려하여 Transformer가 아닌 효율적인 Attention 메커니즘을 적용하고자 하였고, Pooling 함수를 기본으로 하여 단순 연산을 통해 입력 특성에 Attention 메커니즘을 적용하는 Attention 모듈인 CBAM [82] 모델을 활용하였다. CBAM은 채널 방향 (Channel Wise)과 공간 방향 (Spatial Wise)으로 최대 Pooling 함수와 평균 Pooling 함수를 사용하여 입력된 특성 벡터로부터 무엇 ("What")과 어디 ("Where")를 강조해야 할지를 연산한다. 기본 연산을 Pooling 함수로 하기 때문에 CBAM은 연산량과 모델의 파라미터를 크게 증가시키지 않은 채 Attention 메커니즘을 적용할 수 있다는 장점이 있으며 본 논문에서도 이를 활용하여 효율적인 객체 인식 알고리즘을 개발하였다.

2.4 시계열 데이터 예측을 위한 딥러닝 네트워크

LSTM (Long Short Term Memory) [80]은 RNN이 장기 의존성을 보존하기 못해 길이가 긴 시계열 데이터를 학습하기 어렵다는 단점을 해소하기 위해 제안된 모델이다. LSTM이 제안된 이후 LSTM의 방법론을 차용하여 다양한 도메인에 적용하기 위한 모델들이 제시되었다 [78], [83]–[86]. 이 중에서 ConvLSTM [78]은 기본적으로 FC (Fully Connected)-LSTM [83]에서 input-state 간, state-state 간의 변환 과정에서 컨볼루션 연산을 통해 연산이 이루어지도록 한 모델이다. 이를 통해 영상과 같이 시간적 의존성뿐 아니라 공간적 의존성까지 지니는 4

차원의 데이터로부터 시공간적 특성을 학습하는 것이 가능하다. ConvLSTM은 CNN과 LSTM이 결합되어 구성된 모델이기 때문에 컴퓨터 비전에서도 주로 Video Segmentation [87], Action Proposal [88], Sequence Prediction [89]과 같은 영상 분석 분야에서 많이 사용된다. 또한, 포인트 클라우드 시계열 데이터를 활용하는 최근 연구 [8], [76], [77], [90]에서도 ConvLSTM을 기반으로 여러 방법론을 제시하였다. 본 논문에서도 이러한 ConvLSTM의 시공간적 특성을 동시에 추출할 수 있다는 장점을 활용하여 객체의 시계열 형상 특성으로부터 객체의 동적 특성을 추출하는 데 사용하였다.



제 3 장 포인트 클라우드 시계열 데이터 기반 3D 객체

인식 기술 개발

본 장에서는 객체 인식을 위한 파이프라인에 관한 설명을 기술하였다. 먼저 해결하고자 하는 문제를 정의하고 이를 위해 입력 데이터 구성 방법에 대해 설명하였다. 이후 시계열 데이터로부터 객체의 형상 특성과 동적 특성을 추출하는 방법론에 대해 상세히 기술하였으며 시계열 데이터 사용 시 필연적으로 발생하는 순차 입력을 줄여 연산 속도를 개선하기 위한 Queue 구조 설계에 대해서도 설명하였다.

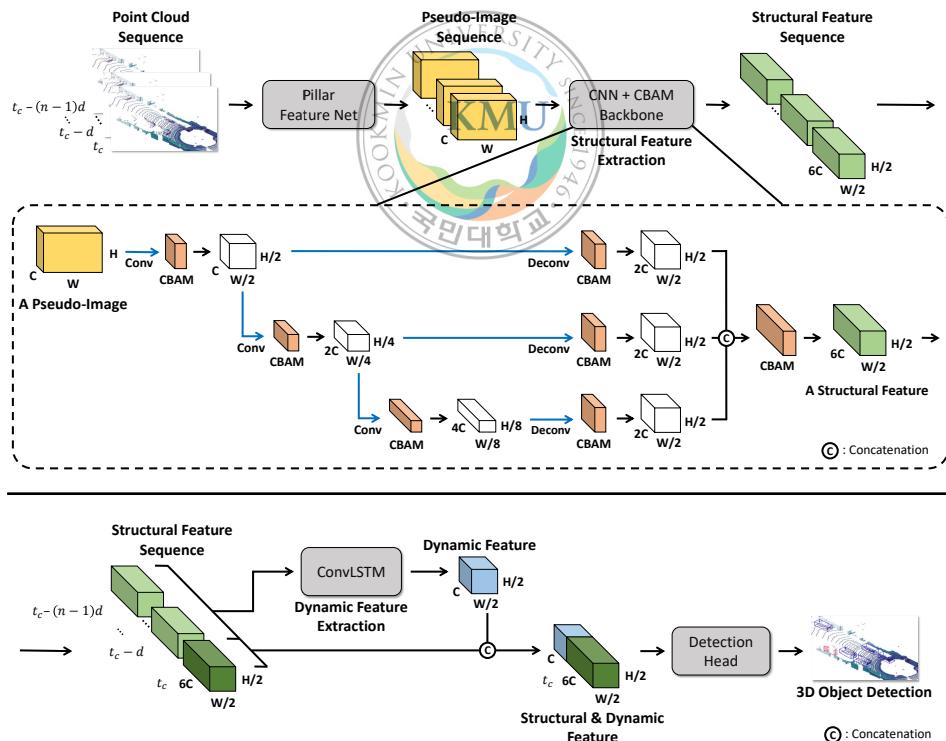


그림 10 The entire pipeline of the proposed 3D object detection method

3.1 문제 정의

본 연구에서 해결하고자 하는 문제는 기본적으로 3차원 객체 인식 기술이다. 그러나 이를 위해 객체 인식 모델의 입력 데이터로 단일 Frame의 포인트 클라우드가 아닌 다중 Frame의 포인트 클라우드 시계열 데이터를 사용하였다. 구체적으로는 먼저 입력 데이터 기준 현재 시점을 t_c , 입력 포인트 클라우드의 Frame 수를 n , 입력 포인트 클라우드의 Frame 사이 간격을 d 로 정의하였다. 위와 같이 정의된 파라미터를 기준으로 $\{P_i \in R^{p_i \times C}\}, i = t_c, t_c - d, t_c - 2d, \dots, t_c - (n-1)d$ (여기서 P_i 는 C 채널을 가지는 p_i 개의 포인트로 구성된 포인트 클라우드이다.)로 정의된 포인트 클라우드 시계열 데이터를 제시된 모델에 순차 입력하였다. C 채널에는 라이다 좌표계를 기준으로 하는 포인트의 X, Y, Z 좌표와 레이저의 강도, 색과 같은 추가 정보들이 포함된다. 이러한 시계열 포인트 클라우드로부터 본 연구의 모델은 현재 시점 t 의 포인트 클라우드에 존재하는 객체의 3차원 바운딩 박스 $(c_x, c_y, c_z, h, w, l, \theta)$ (c_x, c_y, c_z : 바운딩 박스 중심 x, y, z 좌표, h : 바운딩 박스 높이, w : 바운딩 박스 폭, l : 바운딩 박스 길이, θ : 바운딩 박스 회전각)과 클래스 확률값을 출력한다.

3.2 포인트 클라우드 시계열 데이터 구성

포인트 클라우드 시계열 데이터는 입력 포인트 클라우드 Frame 수 n 과 Frame 사이 간격 d 에 따라 구성되도록 하였다. 예를 들어 현재 시점 $t = t_c$ 를 기준으로 입력 Frame 수는 $n = 3$, 입력 Frame 사이 간격은 $d = 10$ 인 경우 [그림 11]와 같이 $t_c - 20, t_c - 10, t_c$ 에 해당하는 시점의 포인트 클라우드 시계열 데이터가 객체 인식 모델에 순차 입력되는 형태이다. 이러한 포인트 클라우드 시계열 데이터를 통해 현재 시점에서 예측이 어려운 객체에 대해 과거 Frame의 객체 정보로부터 보완 받아 예측 정확도가 향상되도록 하였다. 또한, 시계열 데이터

의 특성 추출을 통해 각 객체의 동적 특성을 학습함으로써 객체 인식 성능을 향상시켰다.

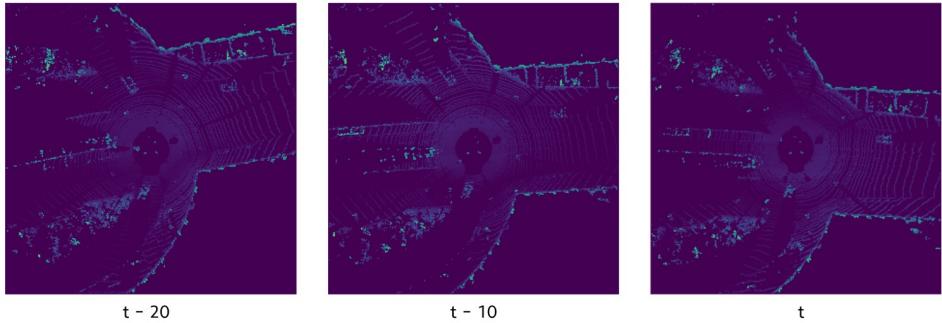


그림 11 An Example of a Point Cloud Sequence ($n = 3, d = 10$)

3.3 객체 형상 특성 추출기

시계열 포인트 클라우드의 각 Frame에서 객체의 형상 특성을 추출하기 위해서 기본적으로 PointPillars [51]를 특성 추출기로 사용하였다. 그리고 PointPillars의 CNN Backbone 네트워크에 CBAM [82]을 적용하여 Attention 메커니즘을 통해 객체의 형상 특성 추출 성능을 강화하였다. 형상 특성 추출기는 [그림 10]의 상단부를 통해 세부적인 구조를 확인할 수 있다.

3.3.1 PointPillars

본 연구에서는 PointPillars [51]의 구조를 차용하여 포인트 클라우드의 특성 추출기로 사용하였고, 포인트 클라우드 시계열 데이터의 각 프레임으로부터 객체의 형상 특성을 추출하도록 하였다. PointPillars의 구조는 [그림 12]에서 나타내었다. PointPillars는 3차원의 포인트 클라우드를 효율적인 인코딩 과정을 통해 Pseudo Image 형태로 변환한다. Pseudo Image는 포인트 클라우드가 아닌 이미지의 형상이므로 간단하게 2D Convolutional Networks에 입력할 수 있는

장점이 있다.

PointPillars는 먼저 Pillar Feature Net (PFN) 구조를 제안하였는데, 이는 포인트 클라우드를 Pillar (기둥) 형태의 Voxel로 Voxelize한 후 PointNet [41]을 간소화한 딥러닝 모델에 Voxelize한 Pillar을 입력하여 Pseudo Image를 생성한다. PFN으로부터 생성된 Pseudo Image는 이후 Region Proposal Networks (RPN) [6]과 비슷한 구조의 2D Convolutional Backbone Networks에 입력되어 객체의 특성을 추출한다. 이렇게 추출된 특성은 최종적으로 Single Shot detector (SSD) [40] 형태의 Detection Head로 입력되어 객체 클래스에 대한 확률값 및 바운딩 박스 회귀값, 객체의 방향에 대한 이산값을 출력한다.

본 논문에서는 포인트 클라우드 시계열 데이터를 시계열 Pseudo Image로 변환하는 데 PFN을 사용하였다. 또한, PointPillars의 2D Convolutional Backbone Networks를 활용하였고, Backbone Networks의 각 2D CNN 블록마다 어텐션 모듈인 CBAM을 적용하여 시계열 데이터로부터 더 나은 형상 특성을 추출할 수 있도록 하였다. 마지막으로 Detection Head는 동일한 조건 하에 성능 비교을 위해 PointPillars와 동일한 Detection Head를 사용하였다.

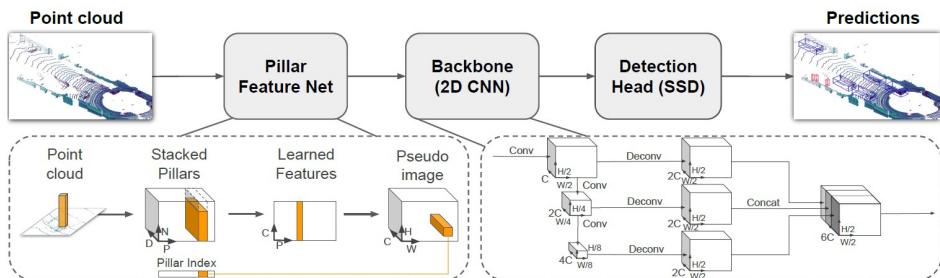


그림 12 Network Overview of PointPillars [51]

3.3.2 CBAM (Convolutional Block Attention Module)

CBAM [82]은 Max Pooling 연산과 Average Pooling 연산을 활용하여 모델의 파라미터 수는 크게 증가시키지 않은 채 Attention 메커니즘을 적용할 수 있어 간단한 구조로 큰 효율성을 발휘할 수 있는 Attention 모듈이다. CBAM에 대한 세부적인 구조는 [그림 13]에 나타내었다.

모듈에 입력되는 특성 $\mathbf{F} \in \mathbb{R}^{C \times H \times W}$ 로부터 CBAM은 Channel Attention Map과 Spatial Attention Map으로 크게 두 가지의 Attention Map을 생성하여 Attention 메커니즘을 적용한다.

Channel Attention Map은 입력 특성의 Channel 방향으로 Max Pooling과 Average Pooling 연산을 각각 하여 각 Channel마다의 대표값을 벡터 형태로 추출한다. 그리고 Shared MLP (Multi-Layer Perceptron)에 입력하여 각 대표값 간의 연관성을 파악하도록 한다. 이후 두 Pooling 연산으로부터의 대표값 벡터를 Element-wise로 덧셈하고 Sigmoid를 적용하여 최종적으로 Channel Attention Map $\mathbf{M}_c \in \mathbb{R}^{C \times 1 \times 1}$ 를 생성한다. CBAM은 \mathbf{M}_c 를 입력 특성 \mathbf{F} 의 크기에 맞게 Broadcast한 후 입력 특성과 Element-wise로 곱셈하여 Channel-wise Attention이 적용된 특성 $\mathbf{F}' \in \mathbb{R}^{C \times H \times W}$ 을 도출한다. 이로써 Channel-wise Attention을 통해 모델이 입력 특성으로부터 무엇 (What)에 집중해야 할지를 학습할 수 있도록 유도한다.

이후 Spatial-wise Attention을 적용하기 위해 입력 특성 \mathbf{F}' 으로부터 공간 (Spatial) 방향으로 Max Pooling과 Average Pooling 연산한다. 그리고 여기서 추출된 두 대표값 행렬을 병합 (Concatenate)하고 1개의 Channel을 출력하는 CNN에 입력한 후 Sigmoid를 적용하여 Spatial Attention Map $\mathbf{M}_s \in \mathbb{R}^{1 \times H \times W}$ 를 생성한다. CBAM은 \mathbf{M}_c 와 마찬가지로 \mathbf{M}_s 를 입력 특성 \mathbf{F}' 의 크기에 맞게 Broadcast하고 입력 특성과 Element-wise로 곱셈하여 Spatial-wise Attention이

적용된 최종 특성 $\mathbf{F}'' \in \mathbb{R}^{C \times H \times W}$ 을 도출한다. 이를 통해 모델이 주어진 과업을 수행하기 위해 입력 특성으로부터 어디 (Where)에 집중해야 할지를 학습하도록 유도하게 된다.

위의 과정에 대한 수식은 아래와 같다.

$$\mathbf{F}' = \mathbf{M}_c(\mathbf{F}) \otimes \mathbf{F}, \quad (3.1)$$

$$\mathbf{F}'' = \mathbf{M}_s(\mathbf{F}') \otimes \mathbf{F}', \quad (3.2)$$

여기서 \otimes 은 Element-wise 곱셈을 나타낸다. 또한, \mathbf{M}_c 와 \mathbf{M}_s 에 대한 수식은 아래와 같다.

$$\begin{aligned} \mathbf{M}_c(\mathbf{F}) &= \sigma(MLP(AvgPool(\mathbf{F})) + MLP(MaxPool(\mathbf{F}))) \\ &= \sigma(\mathbf{W}_1(\mathbf{W}_0(\mathbf{P}_{\text{avg}}^c) + \mathbf{W}_1(\mathbf{W}_0(\mathbf{P}_{\text{max}}^c))), \end{aligned} \quad (3.3)$$

$$\begin{aligned} \mathbf{M}_s(\mathbf{F}) &= \sigma(f([AvgPool(\mathbf{F}); MaxPool(\mathbf{F})])) \\ &= \sigma(f([\mathbf{P}_{\text{avg}}^s; \mathbf{P}_{\text{max}}^s])) \end{aligned} \quad (3.4)$$

여기서 σ 는 Sigmoid 함수를 나타낸다. 식 3.3에서 $\mathbf{P}_{\text{avg}}^c$ 와 $\mathbf{P}_{\text{max}}^c$ 는 각각 Channel Attention Map을 생성하기 위한 Average Pooling 연산과 Max Pooling 연산으로부터의 특성을 나타낸다. 또한, $\mathbf{W}_0 \in \mathbb{R}^{C/r \times C}$ 과 $\mathbf{W}_1 \in \mathbb{R}^{C \times C/r}$ 는 Channel Attention 모듈에서 Shared MLP의 가중치에 해당하며 r 은 [82]에서의 reduction ratio를 나타낸다. 식 3.4에서 f 는 컨볼루션 연산을 나타내며, 컨볼루션 연산의 하이퍼파라미터는 [82]과 동일하게 설정하였다. 또한, $\mathbf{P}_{\text{avg}}^s$ 와 $\mathbf{P}_{\text{max}}^s$ 는 각각 Spatial Attention Map을 생성하기 위한 Average Pooling 연산과 Max Pooling 연산으로부터의 특성을 나타낸다.

본 논문에서는 위와 같은 CBAM을 [그림 10]의 상단부에서 확인할 수 있듯

o], PointPillars의 CNN Backbone Networks에서 매 CNN Block 연산 이후마다 삽입하여 Attention 메커니즘을 적용하였다. [그림 10]에서 청색 화살표가 CNN Block 연산을 나타내며, 이는 PointPillars의 CNN Backbone Networks와 동일한 형태이다. 각각의 CNN Block에 CBAM을 삽입하여 CNN Block으로부터 객체의 형상 특성이 추출될 때마다 해당 특성으로부터 무엇 (What)과 어디 (Where)에 더 집중해야 할지 연산하도록 설계하였다. 이를 통해 포인트 클라우드로부터 객체 형상 특성 추출 성능을 강화하였으며 이에 대한 결과를 4에 나타내었다.

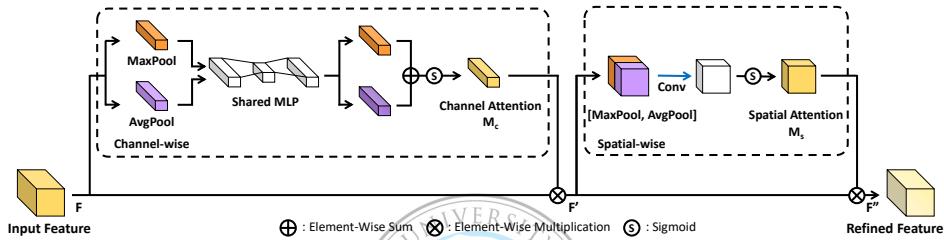


그림 13 The detailed architecture of CBAM

위와 같은 과정은 시계열 포인트 클라우드의 Frame 수 n 만큼 반복된다. n 번의 반복 동안 각 Frame의 포인트 클라우드가 모델의 형상 특성 추출기에 순차 입력되고 형상 추출기로부터 시계열 형상 특성 (Structural Feature Sequence)이 추출된다. 여기서 추출된 시계열 형상 특성은 이후 모델의 동적 특성 추출기로 입력되어 객체의 이동에 따른 동적 특성을 추출하게 된다.

3.4 객체 동적 특성 추출기

시계열 형상 특성으로부터 객체의 동적 특성 (Dynamic)을 추출하기 위한 동적 특성 추출기로는 ConvLSTM [78]모델을 사용하였다. 이에 대한 동작 과정은 [그림 10]의 하단부에서 확인할 수 있다. ConvLSTM에 시계열 형상 특성을 입력할 때에는 먼저 각각의 형상 특성을 ConvLSTM 입력에 맞도록 차원을 설정한 후 병합 (Concatenate)하여 하나의 특성으로 생성한다. 해당 특성을 ConvLSTM

에 입력하면 ConvLSTM은 시간 축 방향으로 순차 연산하여 시계열 형상 특성으로부터 객체의 동적 특성을 추출한다. ConvLSTM의 연산에 대한 수식은 아래와 같다.

$$\begin{aligned}
 i_t &= \sigma(\mathbf{W}_{xi} * \mathbf{X}_t + \mathbf{W}_{hi} * \mathbf{H}_{t-1} + \mathbf{W}_{ci} \otimes \mathbf{C}_{t-1} + b_i), \\
 f_t &= \sigma(\mathbf{W}_{xf} * \mathbf{X}_t + \mathbf{W}_{hf} * \mathbf{H}_{t-1} + \mathbf{W}_{cf} \otimes \mathbf{C}_{t-1} + b_f), \\
 \mathbf{C}_t &= f_t \otimes \mathbf{C}_{t-1} + i_t \otimes \tanh(\mathbf{W}_{xc} * X_t + \mathbf{W}_{hc} * \mathbf{H}_{t-1} + b_c), \quad (3.5) \\
 o_t &= \sigma(\mathbf{W}_{xo} * \mathbf{X}_t + \mathbf{W}_{ho} * \mathbf{H}_{t-1} + \mathbf{W}_{co} \otimes \mathbf{C}_t + b_o), \\
 \mathbf{H}_t &= o_t \otimes \tanh(\mathbf{C}_t)
 \end{aligned}$$

여기서 $*$ 은 컨볼루션 연산을 나타낸다. 또한, i_t 및 f_t , o_t 는 각각 ConvLSTM에서 input gate, forget gate, output gate를 나타내며, \mathbf{C}_t 와 \mathbf{H}_t 는 cell state와 hidden state를 나타낸다. 마지막으로 \mathbf{W} 와 b 는 학습 파라미터와 편향을 나타낸다.

ConvLSTM에서 추출된 객체의 동적 특성은 모델의 객체 인식 Head에 입력되기 전 현시점 객체의 형상 특성과 병합 (Concatenate)되도록 설계하였다 ([그림 10] 하단부 참조). 이를 통해 객체 인식을 수행할 때 형상 특성 추출기는 객체의 형상 특성을, 동적 특성 추출기는 객체의 동적 특성을 추출하는 것에 집중할 수 있도록 유도하였다. 이에 대한 비교 분석은 4에서 확인할 수 있다.

3.5 객체 인식 Head와 Loss (손실) 함수

객체 인식 Head (Detection Head)는 PointPillars와 동일하게 SSD (Single Shot Detector) [40] 모델의 설정을 사용하였고, 객체 인식 Head로부터 객체의 바운딩 박스 값 및 바운딩 박스의 방향, 클래스 확률값을 예측한다.

예측으로부터 손실을 계산할 때에도 PointPillars와 동일한 Loss (손실) 함수를 사용하였다. 먼저 바운딩 박스 예측을 위한 회귀 Loss 함수를 보면, 정답 바운

당 박스와 예측 바운딩 박스 앵커 (Anchor)가 $(c_x, c_y, c_z, h, w, l, \theta)$ 로 정의된다고 하였을 때, Loss 계산을 위한 정답 바운딩 박스와 예측 바운딩 박스의 차이는 아래와 같이 계산된다.

$$\begin{aligned}\Delta c_x &= \frac{c_x^{gt} - c_x^a}{d^a}, \\ \Delta c_y &= \frac{c_y^{gt} - c_y^a}{d^a}, \\ \Delta c_z &= \frac{c_z^{gt} - c_z^a}{d^a}, \\ \Delta w &= \log \frac{w^{gt}}{w^a}, \\ \Delta l &= \log \frac{l^{gt}}{l^a}, \\ \Delta h &= \log \frac{h^{gt}}{h^a}, \\ \Delta \theta &= \sin(\theta^{gt} - \theta^a)\end{aligned}\tag{3.6}$$

여기서 gt 와 a 는 각각 정답값과 예측값을 의미하며, $d^a = \sqrt{(w^a)^2 + (l^a)^2}$ 이다. 위와 같이 계산된 정답값과 예측값의 차이를 통해 회귀 Loss \mathcal{L}_{reg} 는 SmoothL1 함수를 사용하여 아래와 같이 구한다.

$$\mathcal{L}_{reg} = \sum_{b \in c_x, c_y, c_z, w, l, h, \theta} \text{SmoothL1}(\Delta b)\tag{3.7}$$

예측된 바운딩 박스의 회전각만으로는 바운딩 박스의 진행 방향 (Heading)은 알 수 없기 때문에 사용하는 Loss가 방향 Loss \mathcal{L}_{dir} 이며 이는 아래와 같이 CrossEntropy 함수로 정의된다.

$$\mathcal{L}_{dir} = -p_d^{gt} \log(p_d^a) - (1 - p_d^{gt}) \log(1 - p_d^a)\tag{3.8}$$

여기서 p_d 는 방향에 대한 확률값을 나타낸다.

객체의 클래스에 대한 Loss 함수 \mathcal{L}_{cls} 는 아래와 같이 Focal Loss 함수를 사용

하였다.

$$\mathcal{L}_{cls} = -\alpha(1 - p_c^a)^\gamma \log(p_c^a) \quad (3.9)$$

여기서 p_c 는 각 앵커의 클래스에 대한 확률값을 나타낸다. 또한 α 와 γ 는 각각 $\alpha = 0.25$, $\gamma = 2$ 로 설정하였다.

위의 세 가지 Loss를 모두 합한 최종 Loss는 아래와 같이 정의된다.

$$\mathcal{L} = \frac{1}{N_{pos}}(\beta_{reg}\mathcal{L}_{reg} + \beta_{cls}\mathcal{L}_{cls} + \beta_{dir}\mathcal{L}_{dir}) \quad (3.10)$$

여기서 N_{pos} 는 배경이 아닌 실제 객체 앵커의 수를 나타내며 $\beta_{reg} = 2$, $\beta_{cls} = 1$, $\beta_{dir} = 0.2$ 로 설정하였다.

3.6 연산 속도 개선을 위한 Queue 구조 설계

시계열 데이터를 사용함에 있어 가장 큰 결점 중 하나는 데이터가 무조건 순차적으로 모델에 입력되어야 한다는 점이다. 이로 인해 객체 인식 과정에서의 연산 속도는 필연적으로 모델에 입력되는 포인트 클라우드의 프레임 수에 비례하여 감소할 수 밖에 없다.

한편, 자율 주행 여부에 상관 없이 주행 상황을 고려하면, 차량에 장착된 모든 센서로부터의 데이터는 시계열 형태로 수집된다. 따라서 현재 Frame에서 수집된 데이터 (또는 본 연구에서는 객체의 형상 특성)를 저장해놓으면, 해당 데이터는 다음 Frame의 데이터에 대한 과거 데이터로 재활용할 수 있다.

이러한 점에 착안하고 위와 같은 시계열 데이터 사용 시 발생하는 문제를 해소하기 위해 본 연구에서는 FIFO (First-In First-Out) 큐를 활용하여 시계열 포인트 클라우드 데이터로부터의 특성을 재활용하기 위한 간단하면서도 효율적인 연산 파이프라인을 설계하였다. 본 연구에서 사용한 모델이 현시점 t_c 에서부터

$t_c - (n - 1)d$ 시점까지의 총 n 개 Frame으로 구성된 포인트 클라우드 시계열 데이터를 입력 받는다고 가정할 때, 모델의 형상 특성 추출기는 각 Frame의 데이터로부터 객체의 형상 특성을 추출하여 총 n 개의 시계열 객체 형상 특성이 생성된다. 이 때, 모델이 현시점의 객체를 인식하기 위해 과거 $n - 1$ 개의 형상 특성을 활용한다는 점을 고려하면, 여기서 추출된 t_c 시점에서부터 $t_c - (n - 2)d$ 시점까지의 객체 형상 특성들은 다음 시점인 $t_c + d$ 시점에서의 과거 $n - 1$ 개 형상 특성이 될 수 있다. 따라서 본 논문에서는 t_c 시점에서 객체를 인식한 후 여기서 사용된 객체의 형상 특성들을 버리지 않고 정의되어 있는 큐에 저장하여 다음 시점들에서 재사용될 수 있도록 설계하였다.

구체적으로 먼저 형상 특성을 큐에 저장할 때에는 시계열 데이터의 첫 번째 시점 외의 나머지 모든 시점에서는 현시점 t_c 시점에서의 과거 형상 특성 모두가 아닌 t_c 시점의 객체 형상 특성 하나만을 큐에 저장한다. t_c 시점에서의 과거 형상 특성은 과거 시점 ($t_c - d, t_c - 2d, \dots, t_c - (n - 1)d$)에서 이미 큐에 저장되어 있는 상태이기 때문에 $n - 1$ 개의 과거 형상 특성들은 t_c 시점에서 저장하지 않는다. 단, 시계열 데이터의 첫 번째 시점에서는 큐에 어떠한 특성도 저장되어 있지 않으므로 첫 번째 시점에서만 n 개의 모든 형상 특성이 저장되도록 하였다.

그리고 다음 시점인 $t_c + d$ 에서는 큐에 저장되어 있는 $n - 1$ 개의 형상 특성이 $t_c + d$ 시점에서 객체 인식을 하기 위한 과거 형상 특성으로 사용된다. $t_c + d$ 시점에서 객체 인식이 수행되고 나면, 큐에 저장되어 있는 형상 특성 중 가장 첫 번째 인덱스에 위치하고 있는 형상 특성은 다음 시점인 $t_c + 2d$ 에서는 더 이상 필요하지 않기 때문에 큐에서 제거된다. 이러한 프로세스는 시계열 데이터를 처리하는 매 시점마다 반복된다.

또한, 위의 프로세스는 모델을 학습하는 과정이 아닌, 모델을 사용하여 추론하는 과정에서만 적용한다. 학습 과정에서는 모델에 입력되는 시계열 데이터가

연속된 시점으로 차례대로 입력되는 것이 아닌, 데이터가 무작위로 순서가 뒤 바뀌어 입력 (Shuffle)되기 때문에 이러한 프로세스를 적용하기 어렵다. 이렇듯, 간단하면서도 효율적인 큐를 통한 연산을 활용하여 시계열 데이터로부터의 형상 특성을 재활용할 수 있도록 하였고, 이와 관련된 결과는 4에 나타내었다.



제 4 장 객체 인식 실험 결과 및 분석

본 장에서는 객체 인식 실험 결과를 나타내고 이를 분석하였다. 먼저 모델 학습 방법과 학습을 위한 실험 설정을 기술하였다. 이후 모델의 성능을 정량적으로 평가하기 위한 평가지표에 대해 설명하였고, 인식 성능이 가장 좋은 모델에 대한 정량적 결과 및 정성적 결과를 보였다. 마지막으로 본 연구에서 개발한 모델의 성능을 뒷받침하기 위해 Ablation Study 결과를 나타내었다.

4.1 학습 방법

모델을 학습시킬 때에는 [그림 10]의 전체 모델을 한 번에 학습시키는 것이 아닌, 객체 형상 특성 추출기와 동적 특성 추출기를 나누어 학습시켰다. 먼저 객체 형상 특성 추출기를 학습시킬 때에는 단일 Frame의 포인트 클라우드를 사용하였다. 단일 Frame의 포인트 클라우드로부터 객체의 형상 특성을 추출한 후 해당 특성을 객체 인식 Head에 입력하여 객체 인식을 수행하였다. 이는 단일 Frame의 데이터를 사용하는 일반적인 객체 인식으로도 볼 수 있다.

이후 객체 동적 특성 추출기를 학습시킬 때에는 위에서 학습시킨 객체 형상 특성 추출기의 가중치는 고정시켰으며 객체 형상 특성 추출기를 학습시킬 때 함께 학습된 객체 인식 Head는 사용하지 않았다. 또한, 동적 특성 추출기는 시계열 객체 형상 특성을 입력으로 하므로 시계열 포인트 클라우드를 가중치가 고정된 형상 특성 추출기에 순차 입력하여 시계열 객체 형상 특성을 추출한 후 이를 동적 특성 추출기에 입력하여 객체의 동적 특성을 추출하였다. 이후 추출된 동적 특성은 현시점의 형상 특성과 병합 (Concatenate)되고 새로운 객체 인식 Head로 입력됨으로써 객체 인식이 수행되도록 하였다.

학습 과정에서는 Adam [32] Optimizer를 사용하였으며, PointPillars [51]의 설정과 동일하게 학습률 (Learning Rate)의 초기값은 0.0002로 하였고, 15

Epoch 마다 학습률이 0.8배씩 감소하도록 설계하였다. 객체 형상 특성 추출기는 PointPillars와 마찬가지로 총 160 Epoch 만큼 학습시켰으며 객체 동적 특성 추출기는 총 100 Epoch 만큼 학습시켜 결과를 도출하였다.

4.2 실험 설정

4.2.1 데이터셋

KITTI 데이터셋 [22]은 자율주행 기술의 여러 분야에서 모델의 성능을 평가하기 위해 널리 사용되는 오픈 데이터셋이다. 객체 인식 분야에서는 KITTI 객체 인식용(Object Detection Benchmark) 데이터셋을 통해 많은 연구에서 개발된 모델을 평가하였다. 그러나 객체 인식용 데이터셋에 속한 각 Frame의 데이터는 연속된 시계열 데이터 형태로 공개된 것은 아니기 때문에 이러한 데이터셋으로는 입력 데이터를 시계열 형태로 구성할 수 없다. 따라서 본 논문에서는 객체 인식용 데이터셋이 아닌 연속된 여러 개의 장면(Scene)으로 구성되어 있는 객체 추적용 (Object Tracking) 데이터셋을 사용하였다. 객체 추적용 데이터셋 사용 근거를 추가적으로 뒷받침하기 위해 데이터셋 별 클래스 분포를 [그림 14]과 [그림 15]에 나타내었다. 먼저 [그림 14]과 [그림 15]에서 알 수 있듯이, 두 데이터셋의 차량 객체 수는 비슷하다. 그러나 보행자나 사이클리스트 객체는 객체 추적용 데이터셋이 객체 인식용 데이터셋보다 훨씬 많은 것을 알 수 있다. 따라서 크기가 작은 객체에 대한 모델의 인식 성능을 평가할 때에는 객체 추적용 데이터셋이 객체 인식용 데이터셋보다 유효하다는 것을 알 수 있다.

객체 추적 데이터셋은 21개 장면의 학습 데이터셋과 29개 장면의 테스트 데이터셋으로 이루어져 있다. 각각의 장면은 64채널의 라이다가 장착된 차량이 주행을 하며 수집된 데이터이다. 데이터셋이 객체 추적용 데이터셋이므로 테스트 데이터셋에서 객체 인식을 위한 정답 label을 제공하지 않는다. 따라서 먼저

학습 데이터셋을 학습, 검증, 테스트, 세 파트로 분할한 후 학습 및 검증용 데이터셋은 모델의 학습에 사용하였고, 테스트용 데이터셋은 모델의 객체 인식 결과를 평가하는 데 사용하였다. 모델의 성능을 평가할 때에는 KITTI 객체 인식 평가 방법에 맞춰 동일한 평가지표를 통해 차량, 보행자, 사이클리스트, 세 가지 클래스에 대한 객체 인식 결과를 사용하였다.



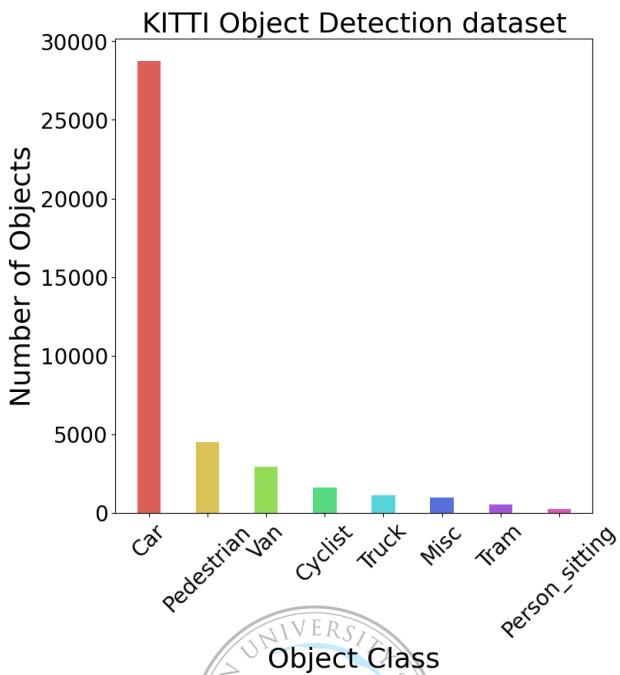


그림 14 Distribution of classes in KITTI Object Detection dataset

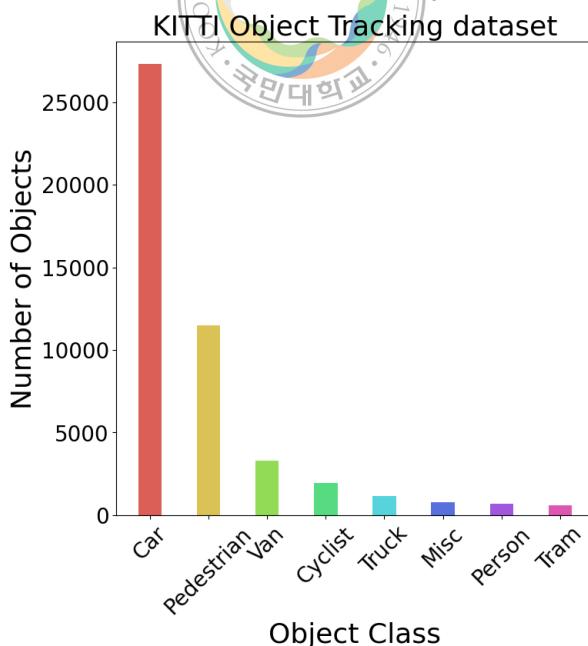


그림 15 Distribution of classes in KITTI Object Tracking dataset

4.2.2 파라미터 설정 및 실험 환경

본 논문에서 실험을 진행할 때에는 PointPillars와 유사한 환경에서 실험하기 위해 파라미터를 기본적으로 PointPillars와 동일하게 설정하였다. Voxelize 시 Pillar의 XY 축에 대한 해상도는 0.16m이고, 각 Pillar에 속할 수 있는 최대 포인트 수는 100개이며 Pillar의 최대 갯수는 12000개로 하였다.

또한, 모델이 예측하는 앵커 박스는 $(c_x, c_y, c_z, h, w, l, \theta)$ 로 설명되며, 0도와 90도로 두 가지 방향으로 적용된다. 양성 일치 (Positive Match) 박스는 앵커 박스와 정답 바운딩 박스 간의 IOU (Intersection Over Union)가 가장 높은 앵커 박스나 IOU가 양성 일치 임계값을 넘는 앵커 박스를 나타내며, 음성 일치 (Negative Match) 박스는 IOU가 음성 일치 임계값보다 낮은 앵커 박스를 나타낸다. 여기서 양성 일치 박스는 참양성 (True Positive, TP)에 해당하며 음성 일치 박스는 거짓양성 (False Positive, FP)에 해당한다. 양성 일치 박스나 음성 일치 박스에 해당하지 않는 앵커 박스는 학습에 사용되지 않았으며, 양성 일치 박스와 음성 일치 박스를 산정하기 위한 임계값은 클래스 별로 아래에 나타내었다. 각 클래스에 따른 파라미터는 아래와 같이 설정하였다.

(a) 차량

차량 객체 학습 시 X, Y, Z 축에 대한 범위는 $[(0, 70.4), (-40, 40), (-3, 1)]$ (m)로 설정하였으며, 앵커 박스의 크기는 폭, 길이, 높이 순으로 $(1.6, 3.9, 1.5)$ (m)로 하였다. 또한, 양성 일치 박스에 대한 임계값은 0.6, 음성 일치 박스에 대한 임계값은 0.45로 설정하였다.

(b) 보행자 및 사이클리스트

보행자 및 사이클리스트 객체 학습 시 X, Y, Z 축에 대한 범위는 $[(0, 48), (-20, 20), (-2.5, 1.5)]$ (m)로 설정하였다. 앵커 박스의 크기는 보행자의 경우 $(0.6, 0.8, 1.73)$ (m), 사이클리스트의 경우 $(0.6, 1.76, 1.73)$ (m)로 하였다.

마지막으로 양성 일치 박스에 대한 임계값은 0.5, 음성 일치 박스에 대한 임계값은 0.35로 설정하였다.

추론 과정에서는 IOU 임계값을 0.5로 하여 Non Maximum Suppression (NMS)을 적용함으로 동일한 객체에 대한 예측 바운딩 박스 중 가장 정확한 박스만 남길 수 있도록 하였다. 추론 과정 시 실험 환경으로는 AMD 사의 Ryzen 3960X CPU와 NVIDIA 사의 3080 GPU가 장착된 데스크톱이 사용되었다.

4.3 평가지표

평가지표는 KITTI 데이터셋[22]에서 사용하는 평가지표인 평균 정밀도 (Average Precision, AP)와 평균 정밀도의 평균 (mean Average Precision, mAP)을 동일하게 사용하였다. 평균 정밀도는 모델이 예측한 결과에 대한 정밀도 (Precision)과 재현율 (Recall)로부터 구할 수 있는데, 먼저 정밀도와 재현율에 대한 식은 아래와 같다.

$$Precision = \frac{TP}{TP + FP}, \quad (4.1)$$

$$Recall = \frac{TP}{TP + FN} \quad (4.2)$$

여기서 TP 는 참양성 (True Positive), FP 는 거짓양성 (False Positive), FN 은 거짓음성 (False Negative)를 의미하며, 모델의 예측 결과를 IOU (Intersection Over Union)에 따라 정답값과 비교하여 결정한다.

IOU는 모델의 예측 바운딩 박스와 정답 바운딩 박스 간의 겹치는 정도를 나타내며 두 바운딩 박스의 교집합 영역을 두 바운딩 박스의 합집합 영역으로 나눈 값을 사용한다. 각 객체마다 IOU를 계산한 후 설정된 IOU 임계값에 따라 계산된

IOU가 임계값 이상이면 *TP*로, 계산된 IOU가 임계값 미만이면 *FP*로 간주하며 정답 바운딩 박스에 대한 예측 결과가 전혀 없는 경우 *FN*으로 간주한다. 본 논문에서는 KITTI 데이터셋과 동일하게 차량 객체는 0.7, 보행자 및 사이클리스트 객체에 대해서는 0.5로 IOU 임계값을 설정하였다. 이와 같이 IOU에 따라 *TP*, *FP*, *FN* 케이스의 객체를 분류하면 정밀도와 재현율을 계산할 수 있다.

정밀도는 모델이 참이라고 예측한 객체 중 실제로 객체일 확률을 의미하고, 재현율은 실제 객체 중 모델이 참이라고 예측한 객체에 대한 확률을 의미한다. 일반적으로 정밀도와 재현율은 trade-off 관계에 있기 때문에 두 지표를 모두 반영하여 하나의 정량적인 값으로 모델의 성능을 측정할 지표가 필요하고, 이 때 사용하는 지표가 평균 정밀도이다.

평균 정밀도를 구할 때에는 정밀도-재현율 곡선 (Recall-Precision Curve)을 사용한다. 정밀도-재현율 곡선은 예측한 객체에 대해 모델이 얼마나 확신을 가지고 예측했는지를 나타내는 Confidence Score에 따라 정밀도와 재현율을 계산하여 그린다. Confidence Score는 객체 분류 시 객체 클래스에 대한 확률값을 사용하거나 또는 객체 클래스에 대한 확률값에 IOU를 곱하여 사용하기도 하는데, 본 논문에서는 객체 클래스에 대한 확률값만 사용하였다. 모델이 예측한 객체를 Confidence Score에 따라 내림차순으로 정렬한 후 가장 Confidence Score가 높은 객체부터 누적하여 정밀도와 재현율을 구하고, 여기서 계산된 정밀도와 재현율을 그래프에 나타낸 것이 정밀도-재현율 곡선이다.

평균 정밀도는 이러한 정밀도-재현율 곡선의 아래 면적에 해당하며, 정밀도와 재현율 모두 0 이상 1 이하의 값을 가지므로 평균 정밀도도 0 이상 1 이하의 값으로 표현된다. 평균 정밀도는 데이터셋에 존재하는 클래스 별로 각각 구하며, 본 논문에서는 평균 정밀도를 구한 후 100을 곱하여 % 단위로 결과를 나타내었다.

평균 정밀도의 평균 (mean Average Precision, mAP)은 위에서 구한 각 클래스별 평균 정밀도를 하나의 수치로 나타내기 위해 단순히 클래스별 평균 정밀도에 평균을 취한 값을 의미한다.

모델의 성능을 평가할 때에는 먼저 객체의 가려짐 정도, 크기 등에 따라 각 객체의 난이도 (Difficulty)을 Easy, Moderate, Hard 케이스로 분류하였다. 또한, 평가지표를 BEV와 3D로 나누어 평균 정밀도 및 평균 정밀도의 평균을 구하였다. 여기서 BEV (Bird's Eye View)는 조감도 상 (2차원)에서의 모델의 성능을 평균 정밀도로 표현한 평가지표이며, 3D는 3차원 공간 상에서의 모델의 성능을 평균 정밀도로 표현한 평가지표이다.

4.4 정량적 결과

모델에 입력되는 포인트 클라우드의 Frame 수 n 을 증가시켜가며 실험을 진행하였고, 이에 대한 정량적 결과를 [표 1]과 [표 2]에 나타내었다. 이 때, 포인트 클라우드의 Frame 사이 간격은 $d = 1$ 로 고정하였다. 각 클래스 및 Difficulty에 대해 성능이 가장 좋은 결과를 볼드체로 표시하였다. 또한, mAP는 PointPillars [51]와 동일하게 Moderate 난이도의 클래스별 평균 정밀도를 평균한 수치를 나타내었다.

먼저 BEV AP 결과 (표 1)를 보면, 모든 클래스 및 난이도에 대해 본 연구에서 개발한 모델이 단일 Frame의 입력 데이터를 사용하는 PointPillars보다 나은 성능을 보였으며, Moderate 케이스의 사이클리스트 인식 결과를 제외하면 $n = 4$ 일 때 모든 클래스에 대해 인식 결과가 가장 좋았으며, mAP 또한 $n = 4$ 일 때 가장 좋은 결과를 보였다.

특히, 단일 Frame만으로 인식이 어려운 Hard 케이스의 객체의 경우 mAP가 가장 높은 $n = 4$ 인 모델을 기준으로 차량, 보행자, 사이클리스트의 순으로

PointPillars에 비해 BEV AP가 각각 4.93%, 5.08%, 5.46% 만큼 크게 증가한 결과를 보였다. 마찬가지로 보행자 및 사이클리스트와 같은 크기가 작은 객체의 경우 $n = 4$ 인 모델을 기준으로 모든 케이스 평균 약 5.20% 만큼 인식 성능이 증가함을 확인하였다.

다음으로 3D AP 결과 (표 2)를 보면, $n = 8$ 일 때 Moderate 및 Hard 케이스의 사이클리스트 인식 결과를 제외하면, 모든 케이스에 대해 기존의 PointPillars 보다 본 논문에서의 모델이 더 나은 성능을 보였다. 또한, Moderate 및 Hard 케이스의 사이클리스트 인식 결과를 제외하면, $n = 4$ 일 때 모든 클래스에 대해 인식 결과가 가장 좋았으며, mAP도 $n = 4$ 일 때 가장 높음을 알 수 있다.

특히, BEV AP와 마찬가지로 Hard 케이스의 객체에 대해 mAP가 가장 높은 $n = 4$ 인 모델을 기준으로 차량, 보행자, 사이클리스트의 순으로 PointPillars에 비해 3D AP가 각각 8.62%, 6.25%, 2.59% 만큼 증가한 결과를 보였으며, 보행자 및 사이클리스트의 크기가 작은 객체에 대해 $n = 4$ 인 모델일 때 모든 케이스 평균 약 4.60%의 증가된 인식 성능을 보였다.

이와 같은 결과를 통해 단일 Frame의 포인트 클라우드만을 사용하는 기존의 PointPillars보다 본 연구에서 개발한 다중 Frame의 시계열 포인트 클라우드를 입력으로 하는 모델을 사용하였을 때 객체 인식 성능이 모든 평가지표에 대해 전반적으로 향상하는 것을 알 수 있다. 또한, 단일 Frame의 포인트 클라우드만으로는 인식이 어려운 크기가 작은 객체 및 Hard 케이스의 객체에 대해서 시계열 포인트 클라우드를 사용함으로써 인식 성능이 매우 큰 폭으로 향상될 수 있음을 증명하였다.

한편, $n = 4$ 인 경우까지는 입력되는 포인트 클라우드의 Frame 수가 증가 할수록 두 평가지표에 대해 모든 클래스에서 AP가 증가하는 경향을 보였으나, $n = 8$ 인 경우에는 오히려 AP가 하락하는 경향을 보였다. 이는 입력되는 포인트

클라우드의 Frame 수 n 이 증가할수록 객체에 대한 정보량은 증가하지만 객체 인식에 사용되는 모델은 n 에 상관없이 동일하기 때문에, 사용된 모델이 객체에 대한 정보량을 모두 수용할 수 있는 용량 (Capacity)을 초과하여 인식 성능이 오히려 떨어진 것으로 예상된다.

연산 속도 측면에서는 본 연구에서 개발한 모델이 시계열 데이터 처리를 위한 순차 연산 및 CBAM을 통한 Attention 메커니즘 적용, 객체 동적 특성 추출을 위한 ConvLSTM 사용 등으로 인해 기존의 PointPillars 대비 크게 감소하는 것을 볼 수 있다. 그러나 이러한 연산 속도 감소를 최소화하고자 Queue 자료구조를 활용한 연산 파이프라인을 설계하였으며 이에 대한 결과를 4.6.4에 나타내었다.

표 1 Results of BEV AP on test dataset (KITTI tracking dataset). n denotes the number of frames of point cloud sequence.

Method	mAP	Car			Pedestrian			Cyclist			Speed (FPS)
		Easy	Mod	Hard	Easy	Mod	Hard	Easy	Mod	Hard	
PointPillars [51] ($n = 1$)	64.32	87.99	78.79	77.57	37.97	35.29	35.05	90.68	78.87	78.89	64.92
Ours ($n = 2$)	69.29	87.80	85.12	81.01	41.85	39.03	39.13	93.60	83.72	83.61	9.07
Ours ($n = 4$)	70.52	89.22	86.63	82.50	43.63	40.54	40.13	94.89	84.38	84.35	7.87
Ours ($n = 8$)	69.39	89.05	86.07	80.41	40.60	37.72	37.75	94.33	84.39	84.28	6.23

표 2 Results of 3D AP on test dataset (KITTI tracking dataset). n denotes the number of frames of point cloud sequence.

Method	mAP	Car			Pedestrian			Cyclist			Speed (FPS)
		Easy	Mod	Hard	Easy	Mod	Hard	Easy	Mod	Hard	
PointPillars [51] ($n = 1$)	59.51	85.36	69.07	66.83	32.16	30.13	30.06	90.68	79.32	79.35	64.92
Ours ($n = 2$)	64.12	84.49	75.08	74.32	36.83	34.33	34.44	93.15	82.96	82.90	9.07
Ours ($n = 4$)	64.84	86.26	76.26	75.45	38.26	36.35	36.31	94.52	81.91	81.94	7.87
Ours ($n = 8$)	62.48	85.26	75.88	75.18	34.89	33.00	33.08	94.05	78.56	78.37	6.23

4.5 정성적 결과

다음으로 정량적 결과에서 mAP가 가장 높았던 모델을 사용하여 본 연구의 정성적 결과를 나타내었다. 정성적 결과에서는 각 클래스에 대한 예측 결과 중 의미있는 결과를 추려 나타내었으며 단일 Frame의 포인트 클라우드만을 사용하는 PointPillars [51]와 비교함으로 본 연구의 우수성을 보였다.

먼저 [그림 16]은 Hard 케이스의 차량 객체에 대한 인식 결과이다. PointPillars는 Hard 케이스의 두 차량 객체 중 한 객체에 대해서는 전혀 예측 결과를 내지 못한 반면, 본 연구에서의 모델은 두 차량 객체 모두 정확한 예측 결과를 도출한 것을 볼 수 있다. 또한, PointPillars가 예측 결과를 도출한 객체에 대해서도 본 연구에서의 모델이 더 높은 점수 (Score)로 예측 결과를 낸 것을 알 수 있다.

[그림 16]에서 포인트 클라우드를 살펴보면, 해당 차량 객체들은 인간의 육안으로도 구별하기 어렵기 때문에 해당 Frame의 포인트 클라우드만으로 Hard 케이스의 차량 객체를 예측하는 것은 객체 인식 모델에 매우 어려운 과업이다. 그러나 본 연구에서는 객체 인식 모델이 해당 Frame뿐 아니라 과거 Frame의 포인트 클라우드로부터 객체 정보를 보완 받아 객체 인식을 수행하기 때문에 Hard 케이스의 객체에 대해서도 정확히 예측하는 결과를 도출하였다.

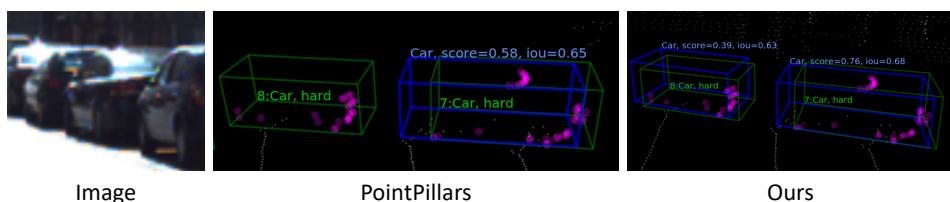


그림 16 The prediction result of hard case car objects. From the left to the right direction, images represent a corresponding camera image, the prediction by PointPillars [51], and the prediction by our model, respectively.

다음은 차량 객체에 대한 거짓양성 (FP) 예측 결과이다. [그림 17]에서 가장 왼쪽 사진을 보면, 사진 상에는 차량 객체가 전혀 존재하지 않으며 건물 출입구

의 형상만을 나타내지만 이에 대한 포인트 클라우드의 경우 육안으로 보았을 때 차량의 형상을 닮아 있다. 이로 인해 단일 Frame의 포인트 클라우드로 객체의 형상 특성만을 학습하는 PointPillars는 [그림 17]의 가운데와 같이 해당 형상에 대해 차량 객체라고 거짓양성 예측 결과를 도출하였다. 반면, 본 연구에서의 모델은 과거 Frame의 데이터로부터 객체의 형상 특성을 보완 받으므로 해당 형상에 대해 예측 결과를 내지 않으며 정확히 예측하는 것을 알 수 있다.

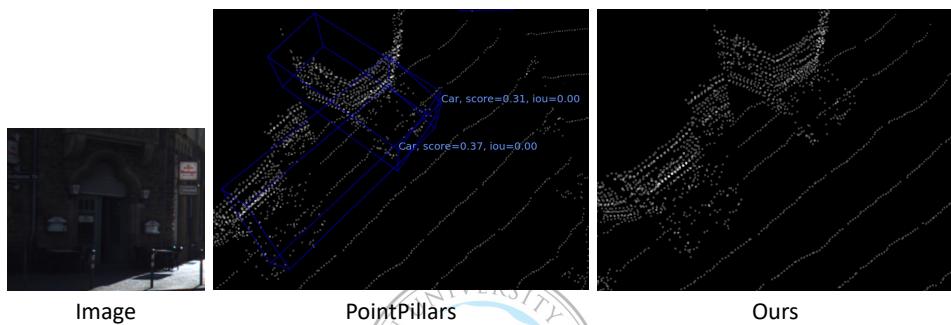


그림 17 The prediction result of false positive car objects. From the left to the right direction, images represent a corresponding camera image, the prediction by PointPillars [51], and the prediction by our model, respectively.

세 번째로 [그림 18]은 자전거를 끌고 가고 있는 보행자 객체에 대한 예측 결과이며 흥미로운 결과를 보였다. 먼저 왼쪽 사진 속의 보행자는 자전거를 타지 않고 끌며 걸어가고 있는 것을 알 수 있다. 따라서 해당 보행자는 사이클리스트가 아닌 보행자로 예측하여야 올바른 예측 결과이다. 그러나 PointPillars는 해당 Frame의 포인트 클라우드로부터 단지 객체의 형상 특성만을 학습하기 때문에 자전거와 사람이라는 형상 특성만으로 해당 객체에 대해 사이클리스트라고 잘못 예측한 결과를 보인다. 이와 다르게 본 연구에서의 모델은 시계열 포인트 클라우드로부터 객체의 형상 특성에 더해 객체의 이동에 따른 동적 특성을 함께 학습하므로 사진 속 보행자의 형상 특성과 보행자 특유의 동적 특성 추출을 통해 정확히 보행자라고 예측한 것을 알 수 있다.

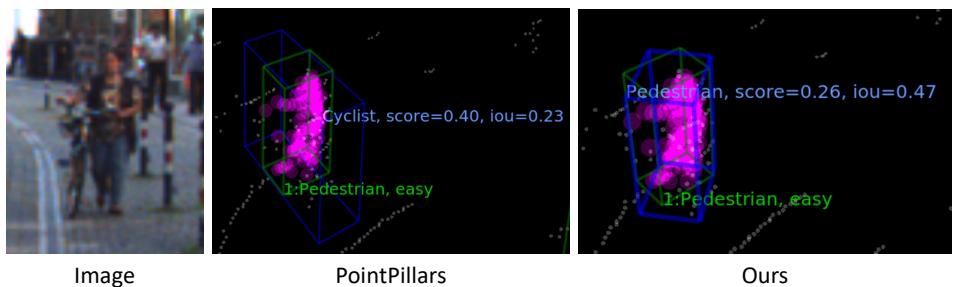


그림 18 The prediction result of a small size object (pedestrian). From the left to the right direction, images represent a corresponding camera image, the prediction by PointPillars [51], and the prediction by our model, respectively.

마지막으로 [그림 19]은 사이클리스트에 대한 거짓양성 예측 결과이다. [그림 19]에서 가장 왼쪽 사진을 보면 자전거가 세워져 있고 사람은 타지 않은 것을 알 수 있다. 지금까지의 결과와 마찬가지로 PointPillars는 객체의 형상 특성만을 학습할 수 있으므로, 포인트 클라우드의 자전거의 형상만으로 해당 객체를 사이클리스트라고 잘못 예측한 결과를 보인다. 그러나 본 연구에서의 모델은 해당 Frame의 형상 특성에 더해 과거 Frame의 포인트 클라우드로부터 객체의 형상 특성을 보완 받고 객체의 동적 특성까지 학습함으로써 [그림 19]의 결과와 같이 정확히 해당 객체를 사이클리스트로 예측하지 않았음을 알 수 있다.

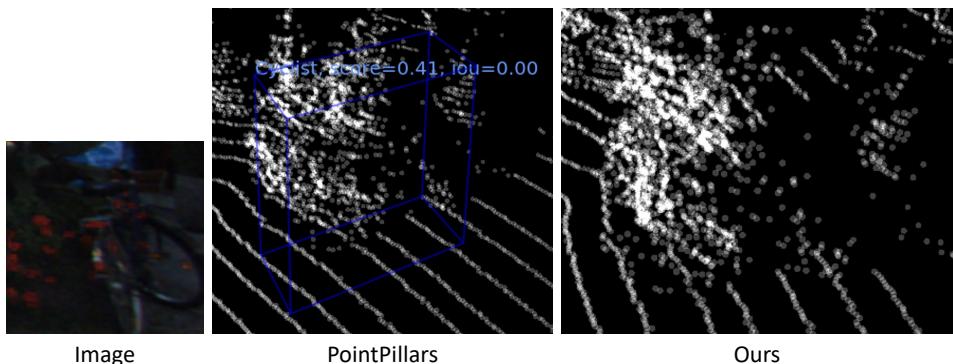


그림 19 The prediction result of a false positive small size object (cyclist). From the left to the right direction, images represent a corresponding camera image, the prediction by PointPillars [51], and the prediction by our model, respectively.

위와 같이 여러 정성적 결과를 통해 본 연구에서의 모델이 시계열 포인트 클라우드로부터 객체의 형상 특성과 동적 특성을 모두 학습함으로써 더 나은 객체 인식 결과를 도출할 수 있음을 보였으며 이를 통해 본 연구의 우수성을 입증하였다.

4.6 Ablation Study

본 연구에서는 모델에 적용한 여러 기술에 대한 성능을 뒷받침하기 위한 Ablation Study를 진행하였으며, 이에 대한 결과를 본 장에 기술하였다.

4.6.1 CBAM Attention 메커니즘 적용

첫 번째는 Attention 모듈인 CBAM [82]에 대한 성능 평가 결과이다. [그림 10]과 같이 본 연구에서는 PointPillars [51]의 CNN Backbone 네트워크에 각 CNN 블록마다 CBAM을 추가하여 입력 특성으로부터 객체의 형상 특성 추출 성능을 강화하였고, 이에 대한 결과를 [표 3]과 [표 4]에서 확인할 수 있다.

먼저 [표 3]은 $n = 1$ 인 경우로, 단일 Frame의 포인트 클라우드 사용 시의

CBAM에 대한 성능을 평가하였다. 그러므로 이는 단일 Frame의 입력 데이터만을 사용하는 일반적인 객체 인식 모델에 대한 결과로도 볼 수 있다. 결과를 보면, 차량 및 보행자에 대한 객체 인식 성능은 기존의 PointPillars보다 CBAM이 적용된 모델이 BEV AP 및 3D AP 모두에서 더 나은 것을 알 수 있다. 반면, 사이클리스트 객체의 경우 Easy 케이스의 객체에 대해서는 CBAM이 적용된 모델의 두 평가지표 성능이 모두 좋았지만, Moderate 및 Hard 케이스에 대한 인식 성능의 경우 기존의 PointPillars가 3D AP에서 더 나은 성능을 보였다. 따라서 이를 심층 분석하기 위한 결과를 [표 4]를 통해 나타내었다.

[표 4]는 $n = 4$ 일 때의 결과로, 본 연구에서 개발한 모델에서 CBAM 적용 여부에 따른 Ablation Study 결과를 나타낸다. [표 3]과 마찬가지로 차량 및 보행자 객체에 대한 인식 성능은 CBAM을 적용하였을 때 더 높은 것을 알 수 있다. 더불어 사이클리스트 객체에 대해서도 $n = 1$ 인 경우와는 다르게 두 평가지표에 대해 모두 CBAM을 적용하였을 때 더 나은 인식 성능을 보였다. 이로써 CBAM이 단일 Frame의 포인트 클라우드를 사용할 때에는 유의미한 결과를 내지 못하더라도, 다중 Frame의 데이터를 사용할 때 Attention 메커니즘으로 각 Frame의 포인트 클라우드로부터 객체 인식에 유효한 객체의 형상 특성을 강조함으로써 인식 성능이 향상될 수 있음을 보였다. 이를 통해 CBAM이 객체 인식 성능 향상에 기여할 수 있음을 입증하였다.

표 3 Comparison of the attention effect of CBAM [82] on PointPillars [51]. n is set to 1.

Method	AP	Car			Pedestrian			Cyclist		
		Easy	Mod	Hard	Easy	Mod	Hard	Easy	Mod	Hard
PointPillars [51] ($n = 1$)	BEV	87.99	78.79	77.57	37.97	35.29	35.05	90.68	78.87	78.89
	3D	85.36	69.07	66.83	32.16	30.13	30.06	90.68	79.32	79.35
PointPillars + CBAM ($n = 1$)	BEV	88.43	83.84	78.42	40.98	37.64	37.44	93.60	79.35	79.14
	3D	85.04	74.66	68.01	38.20	34.92	34.52	93.21	78.84	78.72

표 4 Comparison of the attention effect of CBAM [82] on our model. n is set to 4.

Method	AP	Car			Pedestrian			Cyclist		
		Easy	Mod	Hard	Easy	Mod	Hard	Easy	Mod	Hard
Ours wo CBAM ($n = 4$)	BEV	88.99	84.00	80.64	39.70	37.35	36.11	94.46	82.79	82.64
	3D	87.15	76.22	75.33	34.19	32.33	32.01	94.29	79.81	79.76
Ours w CBAM ($n = 4$)	BEV	89.22	86.63	82.50	43.63	40.54	40.13	94.89	84.38	84.35
	3D	86.26	76.26	75.45	38.26	36.35	36.31	94.52	81.91	81.94

4.6.2 ConvLSTM을 통한 객체 동적 특성 추출

다음으로 ConvLSTM [78]을 사용한 객체 동적 특성 추출에 따른 모델의 성능 결과로, [표 5]에 결과를 나타내었다. [표 5]에서 ConvLSTM을 사용하지 않은 모델의 경우, [그림 10]에서 시계열 형상 특성으로부터 객체의 동적 특성을 추출하지 않고, 시계열 형상 특성을 단순히 병합 (Concatenate)한 후 객체 인식 Head에 입력하여 객체 인식을 수행하도록 하였다.

결과를 보면, ConvLSTM을 사용하지 않았을 때보다 ConvLSTM을 통해 객체의 동적 특성을 추출하여 객체 인식을 수행했을 때 모든 클래스 및 모든 난이도에 대해 두 AP 평가지표 모두 높은 것을 알 수 있다. 이를 통해 ConvLSTM를 사용하여 객체의 동적 특성을 추출함으로써 객체 인식 성능이 향상될 수 있음을 보였다.

표 5 Comparison of the effect of the dynamic feature extraction via ConvLSTM [78]. In the case of the model without ConvLSTM, the n structural features are simply concatenated and directly fed to the Detection Head. n is set to 4.

Method	AP	Car			Pedestrian			Cyclist		
		Easy	Mod	Hard	Easy	Mod	Hard	Easy	Mod	Hard
Ours wo ConvLSTM ($n = 4$)	BEV	88.89	83.96	80.11	41.93	38.86	38.49	92.50	82.06	81.91
	3D	84.17	74.49	71.62	37.92	35.45	35.46	92.21	81.02	80.64
Ours w ConvLSTM ($n = 4$)	BEV	89.22	86.63	82.50	43.63	40.54	40.13	94.89	84.38	84.35
	3D	86.26	76.26	75.45	38.26	36.35	36.31	94.52	81.91	81.94

4.6.3 객체 형상 특성 및 동적 특성의 병합 (Concatenation)

세 번째는 객체의 형상 특성과 동적 특성을 병합하여 객체 인식을 수행함에 따른 성능 평가 결과이며, [표 6]에 이를 나타내었다. [표 6]에서 Dynamic Only는 [그림 10]에서 현시점 t_c 의 객체 형상 특성과 ConvLSTM으로부터의 객체 동적 특성을 병합하지 않고 객체의 동적 특성만을 객체 인식에 사용한 것을 의미한다.

객체 인식 결과에서 Easy 케이스의 차량 객체에 대한 3D AP 결과를 제외하면 객체의 동적 특성만을 사용하는 것보다 객체의 형상 특성과 동적 특성을 병합하여 함께 사용하는 것이 모든 경우에 대해 인식 성능이 개선되는 것을 알 수 있다. Easy 케이스의 차량 객체에 대한 3D AP의 경우에도 차이는 약 0.18% 정도로 작은 차이를 보였다.

LSTM [80] 계열의 모델을 사용할 때에는 LSTM으로부터의 출력 특성만을 사용하여 특정 과업을 수행하는 것이 일반적이다. 그러나 본 연구에서는 ConvLSTM의 출력 특성인 객체 동적 특성만을 사용하는 것이 아니라 ConvLSTM의 입력되었던 현시점 t_c 의 객체 형상 특성을 ConvLSTM으로부터의 객체 동적 특성과 병합하는 방식으로 형상 특성을 다시 활용함으로써 객체 인식을 수행하도록 하였다. 이를 통해 개발한 모델이 객체의 형상 특성과 동적 특성을 나누어 학습할 수 있도록 유도하였고 이러한 방식이 효과적임을 결과를 통해 확인할 수 있었다.

표 6 Comparison of the effect of concatenation of structural and dynamic features. In the case of Dynamic Only, the structural features of current step t are not used. n is set to 4.

Method	AP	Car			Pedestrian			Cyclist		
		Easy	Mod	Hard	Easy	Mod	Hard	Easy	Mod	Hard
Dynamic Only ($n = 4$)	BEV	88.60	85.27	79.63	40.66	37.51	37.37	94.66	82.72	82.68
	3D	86.44	75.41	74.76	35.97	33.81	33.73	94.41	81.91	80.43
Structural + Dynamic ($n = 4$)	BEV	89.22	86.63	82.50	43.63	40.54	40.13	94.89	84.38	84.35
	3D	86.26	76.26	75.45	38.26	36.35	36.31	94.52	81.91	81.94

4.6.4 Queue 자료구조를 통한 연산 속도 개선

마지막 실험을 통해서는 Queue 자료구조를 활용하여 모델의 연산 속도를 개선한 결과를 [표 7]에 나타내었다.

[표 7]에서 먼저 기존의 PointPillars만 사용한 모델 대비 PointPillars에 CBAM이 적용된 모델의 연산 속도가 크게 감소되는 결과를 보이며 CBAM이 CNN 블록 사이마다 적용되면서 연산량이 증가하였음을 알 수 있다. 또한, Queue를 사용하지 않았을 때에는 모델에 입력되는 포인트 클라우드의 Frame 수 n 이 증가할수록 모델의 연산 속도는 이에 반비례하여 급격히 감소하는 결과를 보였다. 반면, Queue를 사용할 때에는 이전 시점에서 기 연산된 객체의 형상 특성을 재활용함으로써 n 이 증가하더라도 모델의 연산 속도 감소량이 그리 크지 않음을 알 수 있고 특히, $n = 8$ 일 때 Queue 미 사용시보다 Queue 사용시 약 4.76배로 연산 속도가 개선되는 것을 확인하였다. 이를 통해 시계열 데이터 사용에 따른 연산 속도 감소는 필연적이기는 하나, 시계열 데이터의 특징에 착안하고 적절한 자료구조를 활용함으로써 연산 속도를 개선할 수 있음을 본 결과를 통해 보였다.

표 7 Comparison of the inference speed between models with/without using the queue.

Method	Speed (FPS)	
	wo Queue	w Queue
PointPillars [51] ($n = 1$)	64.92	-
PointPillars + CBAM ($n = 1$)	10.61	-
Ours ($n = 2$)	5.10	9.07
Ours ($n = 4$)	2.61	7.87
Ours ($n = 8$)	1.31	6.23

제 5 장 결론

본 연구에서는 딥러닝 및 라이다 포인트 클라우드 데이터 기반 3D 객체 인식 기술을 개발하였다. 라이다 포인트 클라우드는 레이저를 통해 객체의 위치를 정밀하게 측정할 수 있지만, 크기가 작은 객체나 다른 사물에 가려진 객체 등에는 레이저가 제대로 도달할 수 없어 이러한 객체는 인식하기가 매우 어렵다는 문제점이 있다. 차량이 이동하며 수집되는 라이다 포인트 클라우드는 시계열 데이터라는 점과 도로 상의 객체는 객체 고유의 동적 특성을 보이며 이동한다는 점에 착안하여 포인트 클라우드를 시계열화하고 객체의 형상 특성 및 동적 특성을 추출함으로써 위와 같은 문제를 해결하고자 하였다.

먼저 사용할 포인트 클라우드의 Frame 수 n 과 Frame 사이 간격 l 을 설정하여 포인트 클라우드 시계열 데이터를 구성하였다. PointPillars를 포인트 클라우드의 특성 추출기로 활용하였고 시계열 데이터를 PointPillars에 순차 입력하여 각 Frame의 포인트 클라우드로부터 객체의 형상 특성을 추출하였다. 또한, 형상 특성 추출 성능을 강화하기 위해 Attention 메커니즘을 활용한 CBAM을 PointPillars CNN Backbone의 각 CNN 블록 사이마다 적용하였고 이를 통해 형상 특성 추출 성능이 향상됨을 확인하였다. 마지막으로 시계열 형상 특성으로부터 객체의 동적 특성을 추출하기 위해 ConvLSTM을 활용하였으며 객체의 형상 특성과 동적 특성을 모두 활용하여 최종적으로 객체 인식이 수행되도록 하였다.

그 결과, 단일 Frame을 사용하였을 때보다 다중 Frame을 사용하였을 때 전

반적인 객체 인식 성능이 향상함과 동시에 크기가 작거나 Hard 케이스의 객체 인식 성능이 크게 향상하는 것을 확인하였다.

5.1 향후 과제

향후 과제로는 먼저 데이터셋 확장이 있다. 본 논문에서는 KITTI Object Tracking 데이터셋 [22]을 활용하였으나, nuScenes 데이터셋 [23]이나 Waymo 데이터셋 [24] 등과 같이 최근 자율주행 분야의 다양한 데이터셋이 오픈 소스 형태로 배포되고 있다. 따라서 이러한 데이터셋에 대한 모델의 예측 성능을 제시하면 본 연구에서 제안한 모델의 성능을 뒷받침할 수 있을 것으로 보인다.

모델의 성능을 개선하는 것도 향후 과제이다. 특히, 본 논문에서는 ConvLSTM [78]을 활용하여 객체의 동적 특성을 추출하였으나, ConvLSTM의 모델 용량 한계로 인해 과대적합이 발생하여 입력되는 포인트 클라우드의 Frame 수 n 이 어느 이상 커질 경우 모델의 성능이 오히려 하락하는 것을 확인하였다. 따라서 향후에는 모델의 동적 특성을 추출할 때에도 ConvLSTM 이 아닌 Attention 메커니즘을 적용하여 시계열 형상 특성으로부터 객체를 인식하는 데 중요한 시공간적 특성을 추출할 수 있는 모델을 설계하고자 한다.

참고 문헌

- [1] M. Park, “The current status and future of autonomous vehicle,” *Summer Annual Conference of IEIE*, pp. 1722–1724, 2017.
- [2] K. Lee, S. Jeon, H. Kim, and D. Kum, “Optimal path tracking control of autonomous vehicle: Adaptive full-state linear quadratic gaussian (lqg) control,” *IEEE Access*, vol. 7, pp. 109 120–109 133, 2019.
- [3] C.-G. ROH, “Configuration and evaluation of local dynamic map platform for connected automated driving in urban roads,” *Journal of Korean Society of Transportation*, vol. 38, no. 1, pp. 42–57, 2020. DOI: 10.7470/jkst.2020.38.1.042. [Online]. Available: <https://doi.org/10.7470/jkst.2020.38.1.042>.
- [4] J. Shuttleworth, “Sae standards news: J3016 automated-driving graphic update,” *SAE International*, 2019.
- [5] J. Redmon and A. Farhadi, “Yolov3: An incremental improvement,” *arXiv preprint arXiv:1804.02767*, 2018.
- [6] S. Ren, K. He, R. Girshick, and J. Sun, “Faster r-cnn: Towards real-time object detection with region proposal networks,” *Advances in neural information processing systems*, vol. 28, 2015.
- [7] S. Shi, X. Wang, and H. Li, “Pointrcnn: 3d object proposal generation and detection from point cloud,” in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 770–779.

- [8] A. El Sallab, I. Sobh, M. Zidan, M. Zahran, and S. Abdelkarim, “Yolo4d: A spatio-temporal approach for real-time multi-object detection and classification from lidar point clouds,” 2018.
- [9] S. Vora, A. H. Lang, B. Helou, and O. Beijbom, “Pointpainting: Sequential fusion for 3d object detection,” in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 4604–4612.
- [10] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia, “Multi-view 3d object detection network for autonomous driving,” in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2017, pp. 1907–1915.
- [11] R. E. Kalman, “A new approach to linear filtering and prediction problems,” 1960.
- [12] S. J. Julier and J. K. Uhlmann, “New extension of the kalman filter to nonlinear systems,” in *Signal processing, sensor fusion, and target recognition VI*, Spie, vol. 3068, 1997, pp. 182–193.
- [13] E. A. Wan and R. Van Der Merwe, “The unscented kalman filter for nonlinear estimation,” in *Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium (Cat. No. 00EX373)*, Ieee, 2000, pp. 153–158.
- [14] T. Yin, X. Zhou, and P. Krahenbuhl, “Center-based 3d object detection and tracking,” in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 11 784–11 793.

- [15] Y. Liu, L. Wang, and M. Liu, “Yolostereo3d: A step back to 2d for efficient stereo 3d detection,” in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2021, pp. 13 018–13 024.
- [16] J. Deng, S. Shi, P. Li, W. Zhou, Y. Zhang, and H. Li, “Voxel r-cnn: Towards high performance voxel-based 3d object detection,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, 2021, pp. 1201–1209.
- [17] J. H. Yoo, Y. Kim, J. Kim, and J. W. Choi, “3d-cvf: Generating joint camera and lidar features using cross-view spatial feature fusion for 3d object detection,” in *European Conference on Computer Vision*, Springer, 2020, pp. 720–736.
- [18] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “Imagenet: A large-scale hierarchical image database,” in *2009 IEEE conference on computer vision and pattern recognition*, Ieee, 2009, pp. 248–255.
- [19] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, *et al.*, “Imagenet large scale visual recognition challenge,” *International journal of computer vision*, vol. 115, no. 3, pp. 211–252, 2015.
- [20] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *arXiv preprint arXiv:1409.1556*, 2014.
- [21] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

- [22] A. Geiger, P. Lenz, and R. Urtasun, “Are we ready for autonomous driving? the kitti vision benchmark suite,” in *2012 IEEE conference on computer vision and pattern recognition*, IEEE, 2012, pp. 3354–3361.
- [23] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, “Nuscenes: A multimodal dataset for autonomous driving,” in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 11 621–11 631.
- [24] P. Sun, H. Kretzschmar, X. Dotiwala, A. Chouard, V. Patnaik, P. Tsui, J. Guo, Y. Zhou, Y. Chai, B. Caine, *et al.*, “Scalability in perception for autonomous driving: Waymo open dataset,” in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 2446–2454.
- [25] A. Géron and H. Bak, *Hands-On Machine Learning with Scikit-Learn, Keras & TensorFlow*. Hanbit Media, 2020.
- [26] B. Xu, N. Wang, T. Chen, and M. Li, “Empirical evaluation of rectified activations in convolutional network,” *arXiv preprint arXiv:1505.00853*, 2015.
- [27] D.-A. Clevert, T. Unterthiner, and S. Hochreiter, “Fast and accurate deep network learning by exponential linear units (elus),” *arXiv preprint arXiv:1511.07289*, 2015.
- [28] G. Klambauer, T. Unterthiner, A. Mayr, and S. Hochreiter, “Self-normalizing neural networks,” *Advances in neural information processing systems*, vol. 30, 2017.

- [29] B. T. Polyak, “Some methods of speeding up the convergence of iteration methods,” *Ussr computational mathematics and mathematical physics*, vol. 4, no. 5, pp. 1–17, 1964.
- [30] J. Duchi, E. Hazan, and Y. Singer, “Adaptive subgradient methods for online learning and stochastic optimization.,” *Journal of machine learning research*, vol. 12, no. 7, 2011.
- [31] T. Tieleman and G. Hinton, “Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude.,” *COURSERA: Neural Networks for Machine Learning*, no. 4, pp. 26–31, 2012.
- [32] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014.
- [33] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” *nature*, vol. 521, no. 7553, pp. 436–444, 2015.
- [34] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Learning internal representations by error propagation,” California Univ San Diego La Jolla Inst for Cognitive Science, Tech. Rep., 1985.
- [35] R. Girshick, J. Donahue, T. Darrell, and J. Malik, “Rich feature hierarchies for accurate object detection and semantic segmentation,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 580–587.
- [36] R. Girshick, “Fast r-cnn,” in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1440–1448.

- [37] K. He, G. Gkioxari, P. Dollár, and R. Girshick, “Mask r-cnn,” in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2961–2969.
- [38] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You only look once: Unified, real-time object detection,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.
- [39] J. Redmon and A. Farhadi, “Yolo9000: Better, faster, stronger,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 7263–7271.
- [40] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, “Ssd: Single shot multibox detector,” in *European conference on computer vision*, Springer, 2016, pp. 21–37.
- [41] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, “Pointnet: Deep learning on point sets for 3d classification and segmentation,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 652–660.
- [42] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, “Pointnet++: Deep hierarchical feature learning on point sets in a metric space,” *Advances in neural information processing systems*, vol. 30, 2017.
- [43] G. P. Meyer, A. Laddha, E. Kee, C. Vallespi-Gonzalez, and C. K. Wellington, “Lasernet: An efficient probabilistic 3d object detector for autonomous driving,” in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 12 677–12 686.

- [44] Z. Yang, Y. Sun, S. Liu, X. Shen, and J. Jia, “Ipod: Intensive point-based object detector for point cloud,” *arXiv preprint arXiv:1812.05276*, 2018.
- [45] J. Ngiam, B. Caine, W. Han, B. Yang, Y. Chai, P. Sun, Y. Zhou, X. Yi, O. Alsharif, P. Nguyen, *et al.*, “Starnet: Targeted computation for object detection in point clouds,” *arXiv preprint arXiv:1908.11069*, 2019.
- [46] C. R. Qi, O. Litany, K. He, and L. J. Guibas, “Deep hough voting for 3d object detection in point clouds,” in *proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 9277–9286.
- [47] Z. Yang, Y. Sun, S. Liu, and J. Jia, “3dssd: Point-based 3d single stage object detector,” in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 11 040–11 048.
- [48] W. Shi and R. Rajkumar, “Point-gnn: Graph neural network for 3d object detection in a point cloud,” in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 1711–1719.
- [49] B. Yang, W. Luo, and R. Urtasun, “Pixor: Real-time 3d object detection from point clouds,” in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2018, pp. 7652–7660.
- [50] Y. Zhou and O. Tuzel, “Voxelnet: End-to-end learning for point cloud based 3d object detection,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4490–4499.
- [51] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom, “Pointpillars: Fast encoders for object detection from point clouds,” in

Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 12 697–12 705.

- [52] J. Ku, M. Mozifian, J. Lee, A. Harakeh, and S. L. Waslander, “Joint 3d proposal generation and object detection from view aggregation,” in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2018, pp. 1–8.
- [53] D. Z. Wang and I. Posner, “Voting for voting in online point cloud object detection.,” in *Robotics: Science and Systems*, Rome, Italy, vol. 1, 2015, pp. 10–15.
- [54] M. Engelcke, D. Rao, D. Z. Wang, C. H. Tong, and I. Posner, “Vote3deep: Fast object detection in 3d point clouds using efficient convolutional neural networks,” in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2017, pp. 1355–1361.
- [55] B. Li, “3d fully convolutional network for vehicle detection in point cloud,” in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2017, pp. 1513–1518.
- [56] S. Song and J. Xiao, “Deep sliding shapes for amodal 3d object detection in rgb-d images,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 808–816.
- [57] M. Simony, S. Milzy, K. Amendey, and H.-M. Gross, “Complex-yolo: An euler-region-proposal for real-time 3d object detection on point clouds,” in *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, 2018, pp. 0–0.

- [58] Y. Yan, Y. Mao, and B. Li, “Second: Sparsely embedded convolutional detection,” *Sensors*, vol. 18, no. 10, p. 3337, 2018.
- [59] M. Ye, S. Xu, and T. Cao, “Hvnet: Hybrid voxel network for lidar based 3d object detection,” in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 1631–1640.
- [60] Y. Wang, A. Fathi, A. Kundu, D. A. Ross, C. Pantofaru, T. Funkhouser, and J. Solomon, “Pillar-based object detection for autonomous driving,” in *European Conference on Computer Vision*, Springer, 2020, pp. 18–34.
- [61] C. R. Qi, W. Liu, C. Wu, H. Su, and L. J. Guibas, “Frustum pointnets for 3d object detection from rgb-d data,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 918–927.
- [62] A. Paigwar, D. Sierra-Gonzalez, Ö. Erkent, and C. Laugier, “Frustum-pointpillars: A multi-stage approach for 3d object detection using rgb camera and lidar,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 2926–2933.
- [63] Z. Wang and K. Jia, “Frustum convnet: Sliding frustums to aggregate local point-wise features for amodal 3d object detection,” in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2019, pp. 1742–1749.
- [64] X. Weng and K. Kitani, “Monocular 3d object detection with pseudo-lidar point cloud,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, 2019, pp. 0–0.

- [65] P. Cao, H. Chen, Y. Zhang, and G. Wang, “Multi-view frustum pointnet for object detection in autonomous driving,” in *2019 IEEE International Conference on Image Processing (ICIP)*, IEEE, 2019, pp. 3896–3899.
- [66] L. Wang, T. Chen, C. Anklam, and B. Goldluecke, “High dimensional frustum pointnet for 3d object detection from camera, lidar, and radar,” in *2020 IEEE Intelligent Vehicles Symposium (IV)*, IEEE, 2020, pp. 1621–1628.
- [67] B. Li, T. Zhang, and T. Xia, “Vehicle detection from 3d lidar using fully convolutional network,” *arXiv preprint arXiv:1608.07916*, 2016.
- [68] J. Zhou, X. Tan, Z. Shao, and L. Ma, “Fvnet: 3d front-view proposal generation for real-time object detection from point clouds,” in *2019 12th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI)*, IEEE, 2019, pp. 1–8.
- [69] Z. Wang, W. Zhan, and M. Tomizuka, “Fusing bird’s eye view lidar point cloud and front view camera image for 3d object detection,” in *2018 IEEE intelligent vehicles symposium (IV)*, IEEE, 2018, pp. 1–6.
- [70] J. Beltrán, C. Guindel, F. M. Moreno, D. Cruzado, F. García, and A. De La Escalera, “Birdnet: A 3d object detection framework from lidar information,” in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, IEEE, 2018, pp. 3517–3523.
- [71] A. Barrera, C. Guindel, J. Beltrán, and F. García, “Birdnet+: End-to-end 3d object detection in lidar bird’s eye view,” in *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, IEEE, 2020, pp. 1–6.

- [72] S. Mohapatra, S. Yogamani, H. Gotzig, S. Milz, and P. Mader, “Bevdetnet: Bird’s eye view lidar point cloud based real-time 3d object detection for autonomous driving,” in *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*, IEEE, 2021, pp. 2809–2815.
- [73] W. Luo, B. Yang, and R. Urtasun, “Fast and furious: Real time end-to-end 3d detection, tracking and motion forecasting with a single convolutional net,” in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2018, pp. 3569–3577.
- [74] J. Yin, J. Shen, C. Guan, D. Zhou, and R. Yang, “Lidar-based online 3d video object detection with graph-based message passing and spatiotemporal transformer attention,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 11 495–11 504.
- [75] C. R. Qi, Y. Zhou, M. Najibi, P. Sun, K. Vo, B. Deng, and D. Anguelov, “Offboard 3d object detection from point cloud sequences,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 6134–6144.
- [76] R. Huang, W. Zhang, A. Kundu, C. Pantofaru, D. A. Ross, T. Funkhouser, and A. Fathi, “An lstm approach to temporal 3d object detection in lidar point clouds,” in *European Conference on Computer Vision*, Springer, 2020, pp. 266–282.
- [77] S. McCrae and A. Zakhor, “3d object detection for autonomous driving using temporal lidar data,” in *2020 IEEE International Conference on Image Processing (ICIP)*, IEEE, 2020, pp. 2661–2665.

- [78] X. Shi, Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, and W.-c. Woo, “Convolutional lstm network: A machine learning approach for precipitation nowcasting,” *Advances in neural information processing systems*, vol. 28, 2015.
- [79] M.-T. Luong, H. Pham, and C. D. Manning, “Effective approaches to attention-based neural machine translation,” *arXiv preprint arXiv:1508.04025*, 2015.
- [80] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
- [81] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, “Attention is all you need,” *Advances in neural information processing systems*, vol. 30, 2017.
- [82] S. Woo, J. Park, J.-Y. Lee, and I. S. Kweon, “Cbam: Convolutional block attention module,” in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 3–19.
- [83] A. Graves, “Generating sequences with recurrent neural networks,” *arXiv preprint arXiv:1308.0850*, 2013.
- [84] I. Sutskever, O. Vinyals, and Q. V. Le, “Sequence to sequence learning with neural networks,” *Advances in neural information processing systems*, vol. 27, 2014.
- [85] J. Donahue, L. Anne Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell, “Long-term recurrent convolutional networks for visual recognition and description,” in

Proceedings of the IEEE conference on computer vision and pattern recognition, 2015, pp. 2625–2634.

- [86] N. Srivastava, E. Mansimov, and R. Salakhudinov, “Unsupervised learning of video representations using lstms,” in *International conference on machine learning*, PMLR, 2015, pp. 843–852.
- [87] N. Xu, L. Yang, Y. Fan, J. Yang, D. Yue, Y. Liang, B. Price, S. Cohen, and T. Huang, “Youtube-vos: Sequence-to-sequence video object segmentation,” in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 585–601.
- [88] H. Zhu, R. Vial, and S. Lu, “Tornado: A spatio-temporal convolutional regression network for video action proposal,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 5813–5821.
- [89] R. Villegas, J. Yang, S. Hong, X. Lin, and H. Lee, “Decomposing motion and content for natural video sequence prediction,” *arXiv preprint arXiv:1706.08033*, 2017.
- [90] X. Sun, S. Wang, M. Wang, Z. Wang, and M. Liu, “A novel coding architecture for lidar point cloud sequence,” *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 5637–5644, 2020.

Abstract

3D Object Detection via Object Structural and Dynamic Feature Extraction of LiDAR Point-Cloud and Validation on Real Vehicle Environment

by Taesan Kim

The Graduate School of
Automobile and IT Convergence,
Kookmin University, Seoul, Korea

As the interest in autonomous driving has dramatically increased in recent few years worldwide, various companies including traditional car manufacturers and giant IT companies, such as Apple or Google, are aiming to research self-driving technologies. In addition, The Society of Autonomous Engineers, SAE, defines 6 levels of autonomous driving according to the level of automation technology to prevent consumers to be confused by the various ranges of each autonomous driving technology and ensure their safety. As such, multiple companies and organizations have been making huge efforts to implement autonomous driving technologies but still, it is needed to progress much more to commercialize safer and more effective self-driving cars than now. In this paper, following these recent research trends, we propose an object detection method based on LiDAR point cloud and deep learning, which is one of the essential parts of autonomous driving technologies.

Recently, in the field of autonomous driving, many works have researched 3D

rather than 2D object detection since accurate prediction about the objects' class, position, and size in 3D space is crucial to ensure safe autonomous driving. Most deep learning-based 3D object detection methods input a single frame of the point cloud. However, with the single frame of the point cloud, it is extremely hard to detect objects that are too small, such as pedestrians or far objects from the LiDAR, or are occluded by other objects so that the laser from the LiDAR rarely reaches these objects. Moreover, every object on the road moves in accordance with its own dynamic features so if these dynamic features can be leveraged, the detection performance could be improved but it is not available with the single frame of data.

Therefore, in this paper, to overcome these constraints, we first serialize data that are input to our object detector. Total n frames of the point cloud from the current step become a point cloud sequence for the input data. And then, the point cloud sequence is sequentially input to the proposed structural feature extractor to obtain the objects' structural features sequence. The structural features sequence is then fed to our dynamic features extractor and it extracts the objects' dynamic features. Finally, we concatenate the dynamic features and the structural features of the current step and feed these concatenated features to the detection head to get the final object's information.

As a result, it is shown that the detection performance improves in all evaluation metrics in all classes. Significantly, the improvement of detection performance on small or hard case objects is remarkable. We prove the presented model performance using the KITTI dataset, which is one of the most popular

datasets in the field of object detection.



Key word : Autonomous Driving, LiDAR Point-Cloud, Deep Learning, 3D
Object Detection, Time-series Data

감사의 글

학위과정 동안 지도 교수님과 동료를 비롯해 끊임없는 응원과 조언을 주신 모든 분들에게 감사의 말씀 드립니다.

이러한 응원과 조언이 있었기에 현 위치까지 성장할 수 있었으며 혼자서는 결코 해내지 못하였을 것입니다. 석사 과정 동안 많은 도움을 주신 모든 분께 다시 한 번 감사하다는 말씀 드리며 이에 보답하기 위해 석사 학위 수여 후에도 계속해서 발전해나가는 사람이 되도록 노력하겠습니다. 감사합니다.

