Journal of Institute of Control, Robotics and Systems (2025) 31(3):196-205
https://doi.org/10.5302/J.ICROS.2025.24.0284 ISSN:1976-5622 eISSN:2233-4335

라이다 기반 객체 추적을 위한 실시간 통합 프레임워크

Realtime and Integrated Framework for LiDAR-based Object Tracking

이 규 석, 김 가 나, 이 재 준, 김 학 일*
(Gyuseok Lee[1], Kana Kim[1], Jejun Lee[1], and Hakil Kim[1,*])
[1]Department of Electrical and Computer Engineering, Inha University

Abstract: This study proposes a real-time integrated framework for LiDAR-based object tracking in autonomous driving environments. Advancements in LiDAR sensors are increasing point cloud data collection, leading to a demand for reliable real-time processing methods. The proposed framework applies voxelization and ground removal techniques to reduce computational load and integrates clustering and deep learning-based object recognition to ensure stability. Combining the point cloud data from LiDAR and the IMU data corrects distortions and refines real-time object movement, enabling accurate tracking in dynamic environments. This framework supports a maximum detection range of 100 m, with a computation time of 52 ms, a positional error of 1.06 m, a heading error of 3.79°, a relative velocity error of 1.46 m/s, and an average tracking frame count of 101, thereby improving object recognition accuracy and tracking performance while fulfilling real-time processing requirements.

Keywords: LiDAR, autonomous driving, 3D object detection, object tracking

I. 서론

자율주행은 차량이 스스로 환경을 인식하고 결정을 내리며, 운전자 없이도 주행을 가능하게 하는 혁신적인 기술이다. 자율주행차량의 주변 환경 인식 성능은 매우 중요하며, 카메라, 레이더, 라이다와 같은 센서와 함께 빠르게 발전하고 있다[1]. 그 중에서도 라이다[2]는 빛의 이동 시간을 측정하여 물체까지 정확한 거리를 파악하는 센서로 수백 미터 떨어진 물체도 감지할 수 있으며, 3차원 공간 데이터를 고해상도로 취득할 수 있다는 장점이 있어 자율주행 분야에서 많이 활용되고 있다.

라이다에서 수집된 포인트 클라우드는 주변 환경을 정밀하게 나타내는 3차원 데이터 집합으로, 자율주행차량이 다른 차량, 보행자, 그리고 기타 장애물을 인식하고 위치를 파악하는데 결정적인 정보를 제공한다. 라이다 기술의 발전과 함께 최근에는 점점 더 높은 반사율과 분해능을 제공하는 라이다 센서들이 등장하고 있다.

고해상도의 라이다는 더욱 정밀한 3차원 데이터를 제공하지만, 그에 따라 생성되는 포인트 클라우드의 크기도 비례하여 증가하게 된다. 이는 대량의 데이터를 실시간으로 처리해야 하는 자율주행 시스템에서 효율적인 연산의 중요성을 부각시키고 있다. 따라서, 보다 최적화된 알고리즘의 개발이 필수적이며, 인식 정확도와 함께 포인트 클라우드의 효율적인 처리가 시스템 성능을 좌우하는 핵심 요소로 떠오르고 있다.

본 논문에서는 그림 1과 같이 라이다 기반 객체 추적 프레임워크를 모듈 단위로 구성하고, 포인트 클라우드를 처리하고 분석하는 각 단계에서 관성 측정 장치(IMU, Inertial Measurement Unit), GPS와 같은 다른 센서들과 결합하여 사용한다. 프레임워크는 Cloud Segmentation, Object Classification, Tracking 세 개 독립된 모듈로 구성되어 있으며, 본 논문에서 제안하는 프레임워크의 목적은 아래와 같다.

1) 포인트 클라우드 분할을 통한 계산 복잡도 감소
2) 분할된 포인트 클라우드 기반의 효율적인 연산
3) 독립된 모듈을 활용한 실시간 동작
4) IMU를 활용한 객체 위치 추정 정확도 향상
5) 클러스터링과 딥러닝을 결합한 객체 추적 안정화

제안하는 프레임워크는 초당 포인트 개수(PPS, Point Per Second)가 115,200인 64채널 라이다를 차량 루프에 장착하여 수집한 데이터셋을 기반으로 성능을 평가한다. 라이다가 장착된 자율주행차량과 추적하고자 하는 목표차량에 측위 센서를 설치하여 단일 객체에 대한 인식 및 추적 성능을 평가하고, 각 모듈의 평균 동작 시간을 측정 및 프레임워크의 전체 동작 시간을 계산하여 실시간성을 검증한다.

2장에서는 자율주행에서 객체 추적과 관련된 기존 연구들을 분석하고, 3장에서는 제안하는 프레임워크의 세부적인 구성 요소를 각 모듈 단위로 설명한다. 4장에서는 프레임 워크의 성능을 도심 환경에서 수집된 라이다 데이터를 사용하여 평가한다.

* Corresponding Author
Manuscript received December 13, 2024; revised January 8, 2025; accepted January 23, 2025
이규석: 인하대학교 전기컴퓨터공학과 대학원생(rbtjr98@inha.edu, ORCID 0009-0008-7913-6192)
김가나: 인하대학교 전기컴퓨터공학과 대학원생(kka-na@inha.edu, ORCID 0009-0001-8866-7162)
이재준: 인하대학교 전기컴퓨터공학과 대학원생(leejaejun122@inha.edu, ORCID 0009-0003-0624-6116)
김학일: 인하대학교 전기컴퓨터공학과 교수(hikim@inha.ac.kr, ORCID 0000-0003-4232-3804)
※ 본 논문은 교육부 및 한국연구재단의 4단계 두뇌한국(BK)21 사업(혁신인재 양성사업)과 2024년도 정부(산업부)의 재원으로 한국산업기술진흥원의 지원을 받아 수행된 연구임(과제번호: P0020536).

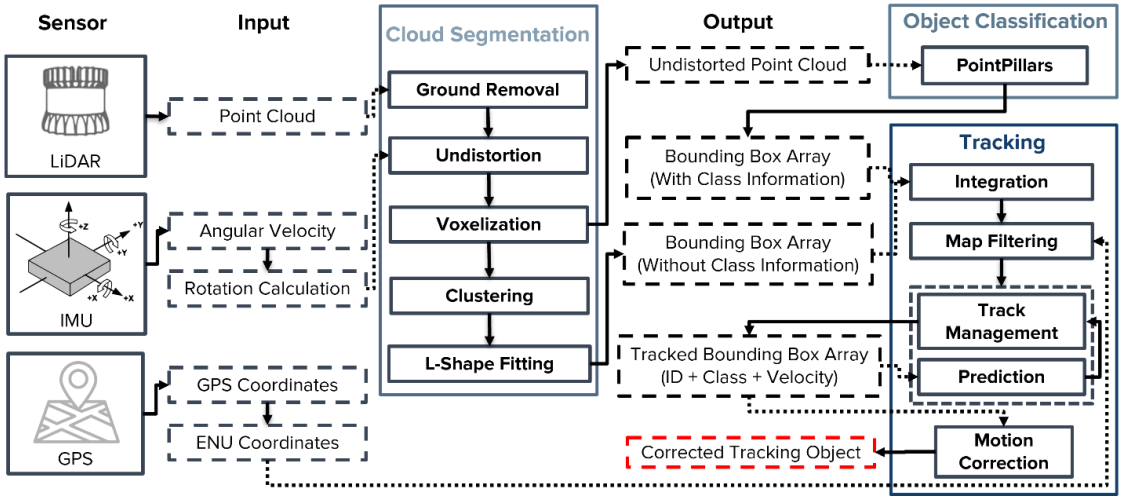


그림 1. 세 개의 모듈로 구성된 프레임워크.
Fig. 1. Framework consisting of three threads.

II. 관련 연구

1. 왜곡 보정

라이다와 같은 시간 차 측정(ToF, Time of Flight) 방식의 센서는 차량이 이동 중일 때 왜곡이 발생한다. 이는 차량 이동 중 수집하는 포인트 클라우드가 물체의 실제 위치와 어긋나게 되어 인식 정확도를 저하시킬 수 있으며, 높은 정확도가 요구되는 자율주행에서 사고로 이어질 가능성이 있다. 따라서, 정확한 포인트 클라우드 데이터를 기반으로 한 객체 탐지를 위해 왜곡 보정 알고리즘은 필수적이다. 이러한 왜곡 보정 기법에는 [3-6]이 있으며, 본 논문에서는 차량에서 주로 사용되는 IMU 기반으로 왜곡 보정이 가능한 [6]을 사용한다.

2. 지면 제거

지면 포인트는 객체 탐지에 있어 관심 영역이 아니며, 이를 제거함으로써 성능 향상과 효율적인 연산이 가능하다. 지면 분할 알고리즘에는 [7-13]이 있으며, 그중 Patchwork++ [13]의 분석 결과에 따르면 도심 환경에서 취득한 포인트 클라우드의 절반 이상은 지면에 해당한다. 이는 라이다 센서의 분해능이 높아질수록 지면 포인트 클라우드의 크기도 증가함을 의미한다. 본 논문에서는 언급된 방법 중 인식 정확도가 가장 높고 연산 속도가 빠른 Patchwork++ [13]을 사용한다.

3. 클러스터링

클러스터링은 데이터를 유사한 특성을 가진 그룹으로 나누는 비지도 학습(unsupervised learning) 기법 중 하나로, 포인트 클라우드를 분할하여 객체를 구분하는 데 사용된다. 인접한 포인트들을 군집화 하여 장애물을 탐지하는 방식으로 연산 속도가 빠르고 모든 장애물을 탐지할 수 있다는 장점이 있다. 그러나 탐지된 객체의 크기가 정확하지 않거나, 분산이 일정하지 않아 정확한 방향을 구하기 어렵다는 단점도 존재한다. 차량에 사용하는 클러스터링 기법은 [14-18]가 있으며, 본 논문에서는 언급된 방법 중 가장 인식 정확도가 가장 높은 적응형 클러스터링[16]을 사용한다.

4. L자형 피팅

클러스터링을 통해 얻은 클러스터는 라이다 좌표계 원점을 기준으로 'L'자 형태를 띈다. 객체 전체에 해당하는 데이터가 아닌, 일부의 포인트 클라우드만 사용하여 형태를 추정해야 하는 라이다 기반 클러스터링에서 주 성분 분석(PCA, Principal Component Analysis)과 같은 방법을 사용하게 될 경우 실제 객체의 방향과 일치하지 않는 상자를 생성할 수 있다. 이러한 문제를 해결하기 위해 라이다 기반 클러스터링에서는 포인트 클라우드의 특성을 반영한 L자형 피팅[19]이 사용되며, 본 논문에서는 [19]의 분산, 넓이, 거리 기반 방법 중 연산 속도가 가장 빠른 거리 기반 기법을 사용한다.

5. 딥러닝

라이다 기반 딥러닝 객체 인식에는 크게 투영 기반 방법[20], 복셀(voxel) 기반 방법[21-25], 그리고 원시 포인트 클라우드 기반 방법[26-27]이 있다. 투영 기반 방법은 3차원 포인트 클라우드를 2차원 평면에 투영하는 방식으로 2차원 이미지 처리 기법을 사용할 수 있으나, 투영 과정에서 정보 손실이 발생할 수 있다. 원시 포인트 클라우드 기반 방법은 3차원 포인트 클라우드를 그대로 사용하여 정보 손실이 적지만, 계산 비용이 높아 실시간성이 떨어진다는 단점이 있다. 이 중 복셀 기반 방법은 포인트 클라우드를 일정한 복셀 또는 기둥(pillar)으로 변환하여 데이터를 구조화하는 방식으로, 3차원 데이터를 사용하여 손실을 줄이면서 계산 효율성을 높일 수 있다. 본 논문에서는 복셀 기반 방법 중 하나인 PointPillars [24]를 사용하여 딥러닝 기반 정확한 객체 인식과 함께 실시간 성능을 확보한다.

III. 프레임워크 구성

1. Cloud Segmentation 모듈

Cloud Segmentation 모듈은 지면 제거(ground removal), 왜곡 보정(undistortion), 복셀화(voxelization), 클러스터링(clustering), 그리고 L자형 피팅(L-shape fitting)으로 구성된다. 해당 모듈에서 객체 인식의 입력에 사용되는 포인트 클라우드의 분할이 이루어지며, 각 모듈은 순차적으로 동작한다.

지면 제거 모듈은 라이다 입력으로 들어온 포인트 클라우드 P에 지면 제거 기법 Patchwork++ [13]를 사용하여, 식 (1)과 같이 지면 P_{ground}가 제거된 포인트 클라우드 $P_{\text{nonground}}$를

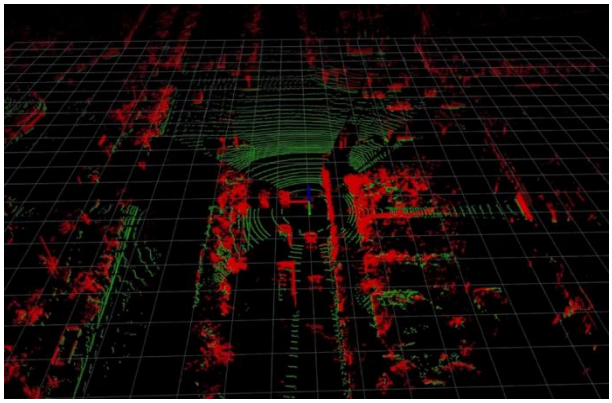


그림 2. 도심에서 지면 제거.
Fig. 2. Ground removal in urban environment.

출력한다. 그림 2는 지면 제거 모듈의 실행 결과로, 초록색으로 표현된 지면 P_{ground}와 빨간색으로 표시된 비지면 $P_{\text{nonground}}$을 확인할 수 있다.

$$P \xrightarrow{Patchwork++} \left(P_{\text{ground}}, P_{\text{nonground}}\right),$$
$$P_{\text{ground}} \cap P_{\text{nonground}} = \emptyset \quad (1)$$

왜곡 보정 모듈에서는 $P_{\text{nonground}}$와 IMU의 회전 정보를 각 포인트의 타임 스탬프에 따라 동기화하여 포인트 클라우드 왜곡을 보정한다. 이를 위해 현재 포인트 클라우드 P의 입력 시점 t_p와 이전 프레임의 입력 시점 t_{p-1} 사이에 측정된 각속도를 누적하여 두 프레임 사이의 회전 변환 R을 계산한다. P의 입력 시점은 스캔이 시작된 순간의 시간이고, 각 포인트는 물체에 반사되는 시점의 시간 정보를 포함하고 있다. 이러한 시간 정보와 라이다 스캔 주기를 활용하여 차량 이동에 따른 포인트 클라우드의 왜곡을 보정할 수 있다.

t_p와 $P_{\text{nonground}}$의 각 포인트 p_i의 측정 시점 t_i의 차이를 반영하여 각 포인트에 대한 회전 변환 R_i를 계산하며, 포인트 클라우드의 입력 시점으로부터 각 포인트의 측정 시간이 t_p와 t_{p-1} 사이에서 차지하는 비율을 R에 적용한다. 즉, 한번의 스캔 동안 하나의 회전 변환을 계산하고, 각 p_i의 측정 시점에 대한 개별 변환을 R을 사용하여 비율적으로 적용해줌으로써 효율적인 연산이 가능하다. 각 포인트에 R_i의 역변환 R_i^{-1}를 적용하여 식 (2)와 같이 왜곡이 보정된 포인트 클라우드 $P_{\text{undistorted}}$를 얻는다[6].

$$P_{\text{undistorted}} = \left\{ R_i^{-1} \cdot p_i \mid p_i \in P_{\text{nonground}} \right\},$$
$$R_i = \frac{t_i - t_{\text{p}}}{t_p - t_{p-1}} \cdot R \quad (2)$$

그림 3과 같이 정밀 지도(HD-Map, High-Definition Map) 위 동일한 시점에서$P_{\text{nonground}}$와 $P_{\text{undistorted}}$를 투영할 경우, 좌측의 빨간색으로 표현된 $P_{\text{nonground}}$에서 좌회전 시 표시한 부분의 정지 차량이 한쪽 차선으로 치우쳐져 있는 것을 볼 수 있다. 이를 왜곡 보정 모듈을 사용하여 보정할 경우, 우측의 흰색으로 표현된 $P_{\text{undistorted}}$에서 차량 포인트 클라우드가 차선 중앙에 위치하는 것을 확인할 수 있다. 왜곡 보정 모듈은 포인트 클라우드 내 모든 포인트를 스캔 시작 시점의 위치로

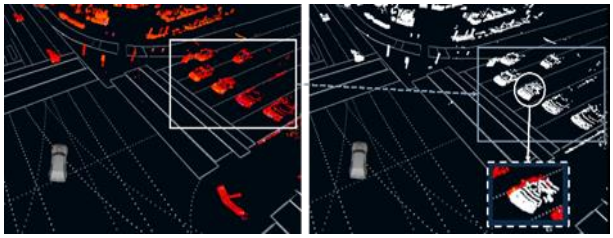


그림 3. 왜곡 보정 전후 비교.
Fig. 3. Comparison of undistortion before and after.

변환하여 후속 과정에서 정확한 객체 인식 및 추적이 가능하게 한다. $P_{\text{undistorted}}$는 복셀화 모듈에 들어가기 전, Object Classification 모듈의 입력으로 사용된다.

복셀화 모듈은 $P_{\text{undistorted}}$를 입력으로 받으며, 클러스터링 모듈에 들어가기 전 포인트 개수를 줄이기 위해 사용된다. 복셀화는 3차원 공간을 격자(grid) 형태의 작은 큐브인 복셀로 나누어 데이터를 구조화하는 과정으로 일정 공간 내에 있는 모든 포인트를 하나의 단위로 묶어 표현한다. 이는 포인트 클라우드를 일정한 해상도로 나누어 데이터의 양을 줄이므로, 연산 속도를 향상시킨다. 식 (3)과 같이 $P_{\text{undistorted}}$를 다운샘플링한 포인트 클라우드 $P_{\text{downsampled}}$를 구한다. l은 복셀의 크기를 의미하며, V_{ijk}는 복셀 (i,j,k)에 속하는 포인트 집합을 의미한다. 그림 4는 좌측의 복셀화 전 $P_{\text{undistorted}}$와 우측의 복셀화 후 $P_{\text{downsampled}}$를 비교한 모습이다. 복셀화 동작 후 포인트 클라우드의 개수가 감소하면서 밀도가 균일해진 것을 확인할 수 있다.

$$P_{\text{downsampled}} =$$
$$\left\{ \frac{1}{|V_{ijk}|} \sum_{p \in V_{ijk}} p \;\middle|\; V_{ijk} \left\{ p \in P_{\text{undistorted}} \;\middle|\; \left\lfloor \frac{p}{l} \right\rfloor = (i,j,k) \right\} \right\} \quad (3)$$

클러스터링 모듈은 $P_{\text{downsampled}}$을 입력으로 받아 적응형 클러스터링[17]을 수행한다. 적응형 클러스터링[16]은 유클리디안 클러스터링[15] 기반의 기법으로, 먼저 식 (4)와 같이 입력 포인트 클라우드를 원점과의 거리 기준 N개의 영역으로 분할한다. 식 (4)에서 p_0는 라이다 좌표계 원점을 의미하며, $P_{\text{downsampled}}$의 포인트 p_j는 n번째 영역의 거리 범위 $[r_{\text{min}}^n, r_{\text{max}}^n]$에 속할 때 영역 P_n에 포함된다.

$$P_{\text{downsampled}} =$$
$$\cup_{n=1}^{N} \left\{ p_j \in P_{\text{downsampled}} \mid d(p_j, p_0) \in [r_{\text{min}}^n, r_{\text{max}}^n] \right\} \quad (4)$$

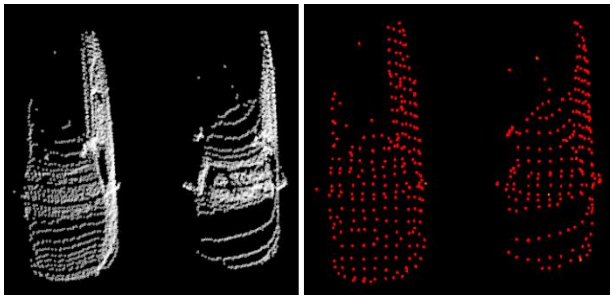


그림 4. 복셀화 전후 비교.
Fig. 4. Comparison of voxelization before and after.

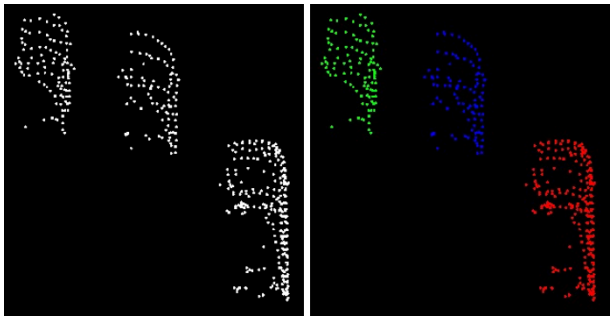


그림 5. 자동차 클러스터링.
Fig. 5. Vehicle clustering.

각 영역 P_n에 해당하는 클러스터 C_k^n의 집합을 식 (5)와 같이 구한다. α는 증가 속도 조정 계수이며, ϵ_n은 n에 비례하여 선형적으로 증가한다. p_i와 p_j 간의 거리가 ϵ_n 이하일 때 해당 포인트는 클러스터 C_k^n에 포함되며, 결과적으로 C를 식 (6)과 같이 구한다. 이를 통해 그림 5와 같이 좌측 $P_{\text{downsampled}}$를 입력으로, 우측 클러스터의 집합 C가 출력된다.

$$C_k^n = \{p_j \mid d(p_i, p_j) \le \epsilon_n, \forall p_j \in P_n \setminus \cup_{l=1}^{k-1} C_l^n\},$$

$$\epsilon_n = \epsilon + \alpha \cdot n \tag{5}$$

$$C = \cup_{n=1}^{N} C_k^n \tag{6}$$

L자형 피팅[19] 모듈은 C를 입력으로 받아 클러스터 형태에 맞는 경계 상자를 출력한다. 식 (7)과 같이 각 클러스터 C_k의 포인트를 z 값을 제외한 C_k'으로 변환하고, 식 (8)과 같이 최솟값과 최댓값 계산을 통해 경계 상자 B를 구한다.

$$C' = \cup_{k=1}^{n}\{C_k' \mid C_k' = \{(x,y) \mid (x,y,z) \in C_k\}\} \tag{7}$$

$$B = \left\{B_k \;\middle|\; B_k = \text{Rect}\begin{pmatrix}\min(x_k), \max(x_k),\\ \min(y_k),\ \max(y_k)\end{pmatrix}\right\},$$

$$(x_k, y_k) \in C_k' \tag{8}$$

그림 6과 같이 각 클러스터 C_k'를 xy평면에 투영한 후 θ만큼 회전해가며 경계 상자 B_k의 각 변 사이 거리를 계산한다. 회전 변환을 적용한 $C_k'(\theta)$의 포인트 $p_\theta(x_{p,\theta}, y_{p,\theta})$와 경계 상자 B_k사이의 최소 거리 $d(p_\theta, B_k)$를 식 (9)와 같이

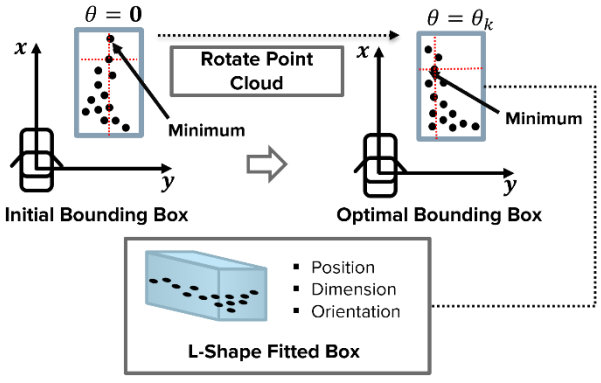


그림 6. L자형 피팅 방법.
Fig. 6. L-Shape fitting method.

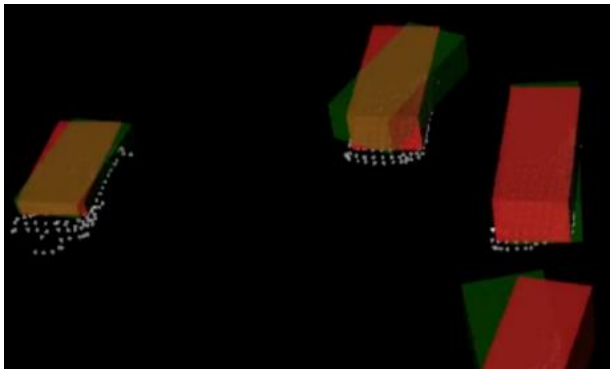


그림 7. PCA와 L자형 피팅 비교.
Fig. 7. Comparison of PCA and L-shape fitting.

정의한다. d_{top}, d_{bottom}, d_{left}, d_{right}는 각각 B_k의 상단 변, 하단 변, 왼쪽 변, 오른쪽 변과 C_k^n사이의 거리를 의미하며, p_θ가 B_k의 변 위에 위치할 경우 $d(p_\theta, B_k)$의 값은 0이 되며, 그 외는 각 변 중 가장 가까운 변과의 거리에 해당한다.

$$d(p_\theta, B_k) = \begin{cases} 0 & \text{if } x_{p,\theta} \in [\min(x_k), \max(x_k)] \text{ and } y_{p,\theta} \in [\min(y_k), \max(y_k)] \\ \min\begin{pmatrix} d_{\text{top}},\\ d_{\text{bottom}},\\ d_{\text{left}},\\ d_{\text{right}} \end{pmatrix} & otherwise \end{cases} \tag{9}$$

$C_k'(\theta)$의 포인트 p_θ와 B_k를 사용하여 비용 함수 $Cost(\theta)$를 식 (10)과 같이 정의한다. $max(d(p, B_k), \epsilon)$에서 최소 거리 임계값 ϵ을 사용하여, 거리가 0이거나 너무 작을 때 비용 함수의 값이 무한대로 치솟는 것을 방지한다. 각 $Cost(\theta)$가 최소화되는 최적의 θ_k는 식 (11)과 같이 구한다.

$$\text{Cost}(\theta) = \sum_{k=1}^{n} \sum_{p_\theta \in C_k'(\theta)} \frac{1}{\max(d(p_\theta, B_k), \epsilon)} \tag{10}$$

$$\theta_k = argmin_{\theta \in \left[0, \frac{\pi}{2}, \theta += \Delta\theta\right]} \text{Cost}(\theta) \tag{11}$$

θ는 C_k'의 중점을 기준으로 0°부터 180°까지 $\Delta\theta$만큼 증가하며 최적의 θ_k를 도출한다. 구해진 각 클러스터에 대한 경계 상자의 정확한 회전 각도 θ_k를 사용하여 식 (12)와 같이 회전된 경계 상자 $B_k(\theta_k)$의 집합 B_{cluster}를 구한다.

$$B_{\text{cluster}} = \cup_{k=1}^{n} B_k(\theta_k) \tag{12}$$

그림 7에서 초록색 경계 상자는 PCA, 빨간색 경계 상자는 L자형 피팅이 적용된 결과이다. PCA는 포인트 클라우드의 분산이 가장 큰 방향을 기준으로 경계 상자를 생성하지만, L자형 피팅은 객체의 형태를 반영하여 정확한 방향의 경계 상자를 생성한다. 구해진 B_{cluster}는 Tracking 모듈의 입력으로 사용된다.

2. Object Classification 모듈

Object Classification 모듈에서는 $P_{\text{undistorted}}$를 PointPillars 모듈의 입력으로 받아 딥러닝 기반으로 객체를 추론한다. 해당 모듈에서는 포인트 클라우드를 복셀 단위로 구조화한 후, 2차원 네트워크를 사용하여 특징을 추출하고 객체를 탐지한다. 탐지

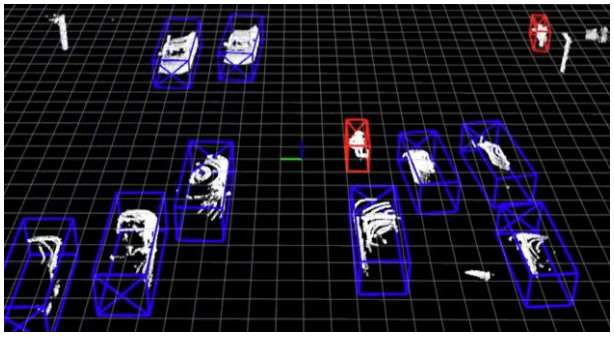


그림 8. 도심에서 딥러닝 기반 객체 탐지.
Fig. 8. Deep learning-based object detection in urban environments.

결과로 추론된 객체의 위치, 크기, 방향 및 클래스 정보를 포함한 경계 상자 B_{deep}이 출력되며, B_{deep}은 B_{cluster}와 함께 Tracking 모듈의 입력으로 사용된다. 그림 8에서 파란색 경계 상자와 빨간색 경계 상자는 각각 차량과 오토바이 운전자를 탐지한 결과이다.

3. Tracking 모듈

Tracking 모듈은 통합(integration), 지도 필터링(map filtering), 추적 관리(track management)와 예측(prediction), 움직임 보정(motion correction)으로 구성된다. 해당 모듈에서 Cloud Segmentation 모듈로부터 B_{cluster}, Object Classification 모듈로부터 B_{deep}을 입력으로 받아 통합 및 객체 추적을 수행하며, 최종적으로 필터링 및 움직임에 대한 보정이 적용된 추적 객체 $B_{\text{corrected}}$를 출력한다.

통합 모듈은 동일한 포인트 클라우드에서 출력한 두 경계 상자 B_{cluster}와 B_{deep}을 IOU (Intersection Over Union)를 계산하여 식 (13)과 같이 $B_{\text{integration}}$으로 통합한다. IOU 값이 임계값 τ를 초과할 경우 객체의 형태를 더 정확하게 추론하는 B_{deep}을 우선적으로 사용하며, 그렇지 않은 경우 B_{cluster}를 사용한다. 그림 9에서 빨간색 경계 상자는 B_{cluster}, 초록색 경계 상자는 B_{deep}을 의미한다.

$$B_{\text{integration}} = \begin{cases} B_{\text{deep}} & \text{if IOU}(B_{\text{cluster}}, B_{\text{deep}}) > \tau \\ B_{\text{cluster}} & otherwise \end{cases} \quad (13)$$

지도 필터링 모듈은 $B_{\text{integration}}$을 입력으로 받아 관심 영역 안에 있는 경계 상자를 필터링하여 B_{filtered}를 출력한다. 본 논문에서는 필터링 기법으로 μ-Lanelet [28] 기반의 필터링 방법을 사용한다. μ-Lanelet [28]은 ENU (East-North-Up coordinate

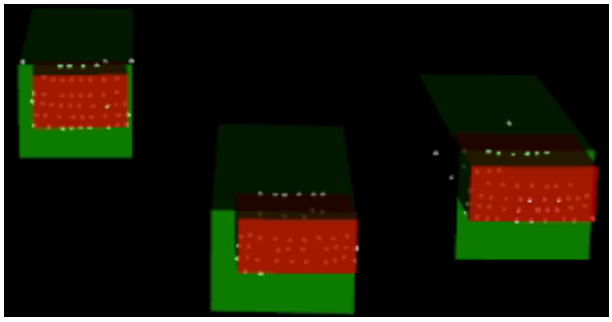


그림 9. 클러스터링과 딥러닝 비교.
Fig. 9. Comparison of clustering and deep learning.

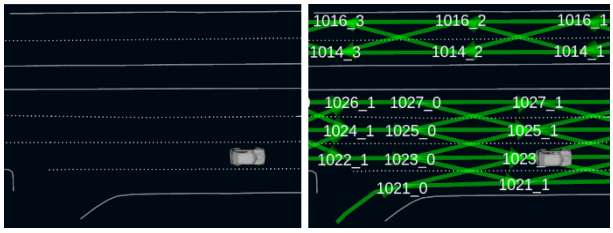


그림 10. 정밀 지도를 μ-Lanelet으로 변환.
Fig. 10. Conversion from HD-Map to μ-Lanelet.

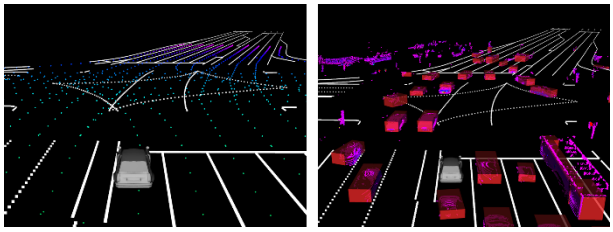


그림 11. 필터링된 정밀 지도 위의 경계 상자.
Fig. 11. Filtered bounding boxes on the HD-Map.

system) 좌표계로 변환된 정밀 지도를 기반으로, 각 차로를 일정 구간 세분화하여 노드화한 그래프이다. 그림 10에서 우측의 숫자로 표현된 각 점들에 해당한다.

ENU 좌표계에서 차량이 위치한 노드를 기준으로 인접 노드 N_{adjacent}를 구하여, N_{adjacent}와 $B_{\text{integration}}$과의 거리 $\text{dist}(b,n)$을 계산한다. $\text{dist}(b,n)$을 사용하여 임계값 r 내에 위치한 B_{filtered}를 식 (14)와 같이 계산한다. 그림 11의 좌측 지도 위 포인트는 N_{adjacent}이며, 이를 통해 필터링된 우측의 빨간색 경계 상자 B_{filtered}는 정밀 지도 위에 위치한 객체만 포함한다.

$$B_{\text{filtered}} = \{b \in B_{\text{integration}} \mid \exists n \in N_{\text{adjacent}}: \text{dist}(b,n) \leq r\} \quad (14)$$

추적 관리와 예측 모듈에서는 B_{filtered}를 입력으로 받아 등속 모델을 기반으로 칼만 필터(kalman filter) 기반의 객체 추적[29]을 수행한다. 이를 통해 객체의 상태를 예측하고 갱신하여 연속적이고 안정적인 추적 결과 B_{tracked}를 생성한다. 예측 단계에서는 이전 상태와 상태 전이 행렬 A, 프로세스 노이즈 공분산 Q 이용해 상태를 예측하며, 갱신 단계에서는 관측 행렬 H, 관측 노이즈 공분산 R, 칼만 이득 K를 통해 관측값과 예측값의 오차를 보정한다. 이 과정은 단계적으로 수행되며, 관측값과 기존 트랙 간 연관성은 최적 할당 알고리즘을 통해 계산된다. 예측과 갱신 과정을 포함한 칼만 필터 기반 추적은 다음 식 (15)와 같다.

$$B_{\text{tracked}} = A\hat{B}_{\text{prev}} + Q + K\left(B_{\text{filtered}} - H\left(A\hat{B}_{\text{prev}} + Q + R\right)\right) \quad (15)$$

식 (15)에서 $\hat{B}_{\text{prev}}$는 이전 상태를 나타내며, $A\hat{B}_{\text{prev}} + Q + R$는 프로세스 노이즈가 반영된 상태 예측 값이다. K는 관측값과 예측값 간 불확실성을 조정하는 가중치 역할을 하며, 관측값 B_{filtered}와 예측값 간 오차를 보정하여 최종 상태를 갱신한다. 즉, 객체의 동적 상태를 지속적으로 추적하며, 관측 데이터의 잡음을 최소화하고 추적의 정확성과 안정성을 높인다. 그림 12에서 구해진 B_{tracked}는 객체의 위치, 크기, 방향, 클래스, 속도 정보를 포함하며, 추적된 시점의 고유

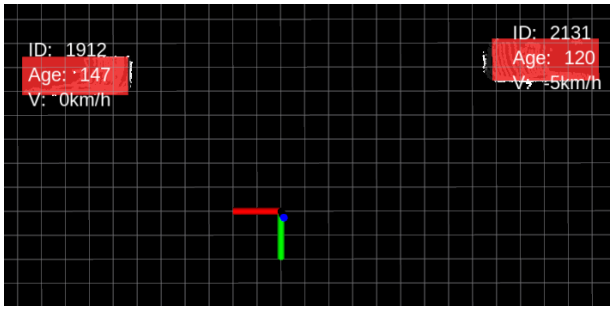


그림 12. 경계 상자로 표현된 추적 객체.
Fig. 12. Tracked object with bounding box.

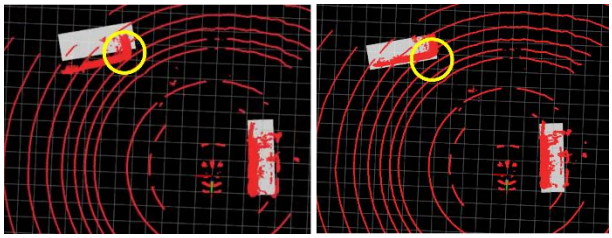


그림 13. 움직임 보정 유무 비교.
Fig. 13. Comparison of motion correction with and without.

ID, 추적된 시점으로부터의 나이(age), 그리고 각 프레임에서의 상대 속도(v)를 포함한다.

움직임 보정 모듈에서는 B_{tracked}를 입력으로 받아 객체의 상대적인 움직임이 보정된 $B_{\text{corrected}}$를 출력한다. 포인트 클라우드의 입력 시점 t_p는 라이다가 스캔을 시작한 시간으로, 해당 시점에서 각 모듈을 통과하여 추적 객체를 얻기까지 연산 시간이 소요된다. 이에 현재 시점 t_c와 t_p 사이의 시간인 Δt 동안 B_{tracked}의 이동량을 계산하여 보정함으로써 현재 시점에서 정확한 추적 객체의 위치를 식 (16)과 같이 계산한다. 객체의 진행 방향 yaw 기준 x 및 y 좌표에서의 이동량을 계산하여 각각 더함으로써, $B_{\text{corrected}}$를 구한다. 그림 13에서 좌측 경계 상자와 우측 경계 상자는 각각 B_{tracked}와 $B_{\text{corrected}}$를 의미하며, 움직임이 보정된 $B_{\text{corrected}}$가 실시간 포인트 클라우드와 비교할 때, 더 정확한 것을 확인할 수 있다.

$$B_{\text{corrected}} = B_{\text{tracked}} + \begin{pmatrix} v \cdot \Delta t \cdot \cos(\text{yaw}) \\ v \cdot \Delta t \cdot \sin(\text{yaw}) \end{pmatrix} \quad (16)$$

IV. 실험 및 결과

1. 실험 환경 구성

본 논문에서는 제안하는 프레임워크의 인식 및 실시간 성능을 평가하기 위해 그림 14와 같은 자율주행차량 및 목표차량을 사용한다. 자율주행차량에 장착된 라이다는 10hz로 PPS가 최대 1,152,000인 Hesai Pandar 64를 사용하며, 포인트 클라우드를 미터 단위 기준으로 x축 ±100, y축 ±100, z축 +0.3만큼 잘라내어 입력으로 사용한다. 측위 센서로 듀얼 안테나가 장착된 Novatel PwrPak7-E1을 INS (Inertial Navigation System)로 사용하며, 국토지리정보원에서 제공하는 RTK (Real Time Kinematic) 보정 신호를 수신하여 정확도가 ±2cm 이내인 위치 정보와 각도 오차가 0.1° 이내인 방향 정보 취득이

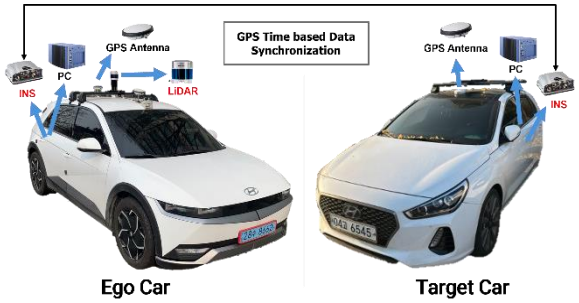


그림 14. 자율주행차량 및 목표차량 구성.
Fig. 14. Configuration of autonomous vehicle and target vehicle.

가능하다. IMU는 125hz로 동작하는 PwrPak7-E1에 내장된 Epson EG320N을 사용하며, 최종적으로 위치, 방향, 속도, GPS 시간 정보를 200hz로 취득한다.

목표차량에 장착된 측위 센서는 자율주행차량과 동일하며, 같은 데이터를 취득한다. 두 차량에서 취득한 GPS 위치 및 방향, 속도를 GT (Ground Truth)로 사용하기 위해 GPS 시간을 기준으로 데이터를 동기화하여 동일한 시점에서의 측위 정보와 포인트 클라우드를 구한다. 그림 15의 좌측과 우측은 각각 동기화가 되지 않은 상태와 동기화된 상태의 데이터를 지도에 투영한 결과이다. 그림 16과 같이 절대 좌표계 내에서 자율주행차량과 목표차량의 위치를 계산하고, 상대 좌표계인 라이다 좌표계에서 탐지한 객체의 위치와 방향을 계산하여 두 값을 비교함으로써 성능을 평가한다. 추적 평가 지표와 함께 두 INS에서 측정한 절대 속도를 기준으로 상대 속도를 계산하여, 추적된 객체의 속도를 평가한다. 탐지한 경계 상자의 IOU를 평가하기 위해 목표차량 Hyundai i30의 제원 기준 전장과 전폭을 사용한다.

프레임워크 동작 및 연산 시간 측정을 위하여 Intel i7-11700KF와 Nvidia GeForce RTX 3060Ti 및 32GB RAM이 장착된 PC를 사용하며, 딥러닝 추론을 위한 학습 데이터셋으로 자율주행차량과 동일한 라이다를 사용하여 구축한 Pandaset [30]을 사용한다. Cloud Segmentation과 Tracking 모듈은 모두 CPU 환경에서 동작하며, Object Classification 모듈은 GPU 환경에서 동작한다. 연산 시간은 모듈 단위로 측정하며, 전체 동작 시간 측정 시 입력 포인트 클라우드를 기준으로 각 모듈의 병렬 연산을 고려하여 계산한다.

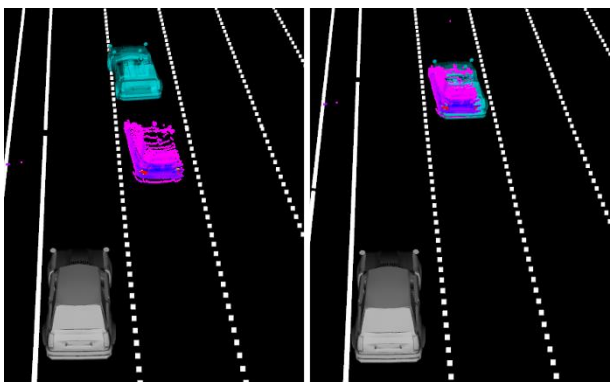


그림 15. GPS 시간 기준 데이터 동기화.
Fig. 15. Data synchronization based on GPS time.

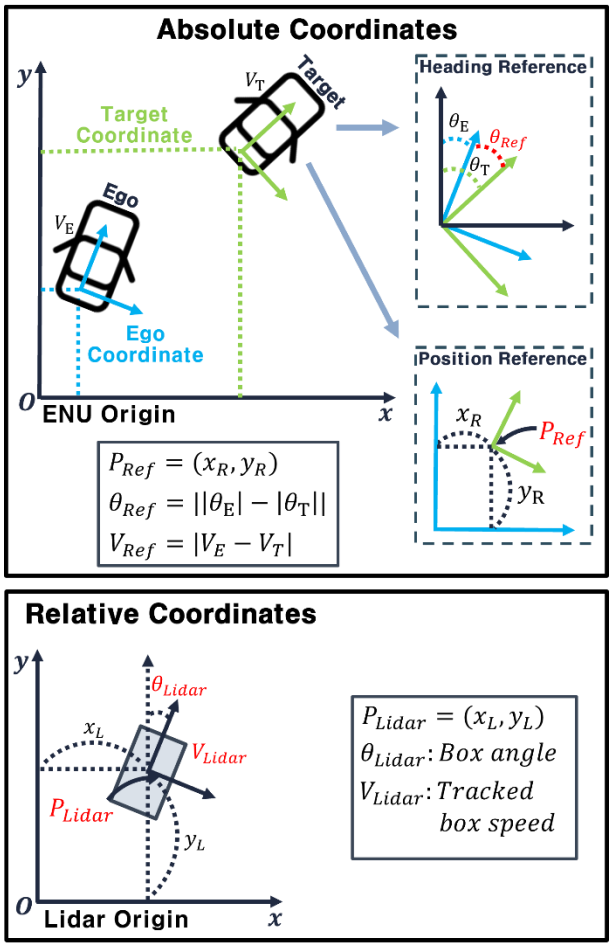


그림 16. 평가 방법.

Fig. 16. Evaluation method.

그림 17. 정밀 지도 내 실험 경로.

Fig. 17. Experimental route with the HD-Map.

제안하는 프레임워크는 그림 17의 인천 연수구 송도동 센트럴 파크 인근 일반도로에서 자율주행차량과 목표차량이 노란색 경로를 따라 주행하며 수집한 데이터셋으로 실험 및 평가를 수행한다. 목표차량 1대로 단일 객체에 대한 성능을 평가하며, 해당 도로는 국토지리정보원에서 제공하는 정밀 지도가 구축되어 있어 μ-Lanelet [28] 기반 필터링이 가능하다. 두 차량을 동일한 지도에 투영하여 절대 좌표계를 설정하고 위치를 계산한다.

2. 실험 및 결과

표 1은 지면이 제거된 포인트 클라우드 기반 클러스터링에서 PCA와 L자형 피팅 방법을 비교한 결과이다. 평가 데이터셋은 차량 간 거리 10~20m, 각도 차이 10° 이내, 목표차량이 옆 차선에 위치한 데이터를 사용해 'L' 형태 클러스터가

표 1. 클러스터링에서 PCA와 L자형 피팅 성능 비교.

Table 1. Comparison of PCA and L-shaped fitting performance in clustering.

Method	IOU	Heading Error (°)		
		min	max	Average
Clustering with PCA	0.45	3.58	40.15	20.26
Clustering with L	**0.55**	0.02	24.20	**7.44**

표 2. 지면 제거 및 복셀화에 대한 객체 탐지 성능 비교.

Table 2. Comparison of object detection performance for ground removal and voxelization.

Method	Precision	IOU	Position Error (m)	Heading Error (°)	Time (ms)
Clustering w/o G/R	0.51	0.10	2.13	4.87	1590
Clustering	0.99	**0.23**	**1.77**	**4.25**	1329
Clustering with V	**0.99**	0.22	1.78	4.48	**15**
Deep learning w/o G/R	0.70	**0.72**	**0.51**	1.79	33
Deep learning	**0.73**	0.70	0.59	**1.63**	**31**

나타나는 구간에서 실험한다. 비교 결과, L자형 피팅이 IOU와 방향 오차(heading error) 측면에서 더 높은 정확도를 보인다. L자형 피팅의 IOU는 0.55로 PCA 기반 방법의 0.45보다 0.1 높으며, 평균 방향 오차도 7.44°로 PCA 기반 방법의 20.26°와 비교하여 정확도가 약 13° 향상된 결과를 보인다.

표 2는 직진 구간 데이터셋에서 지면 제거와 복셀화 적용 여부에 따른 객체 탐지 성능을 비교하며, 클러스터링과 딥러닝 기반 탐지 모두에서 정밀도(precision), IOU, 위치 오차(position error), 방향 오차, 연산 시간(time)을 평가한 결과이다. 클러스터링 기반 탐지에서 지면 제거를 적용하면 정밀도와 IOU가 각각 0.51에서 0.99, 0.10에서 0.23으로 약 2배 향상 되며, 위치 오차는 1.77m로 감소하고, 연산 시간은 약 16% 단축된다. 복셀화를 추가로 적용할 경우 정밀도는 유지되지만, IOU는 소폭 감소하며 위치 및 방향 오차는 기존 수준을 유지한다. 동시에, 연산 시간이 1329ms에서 15ms로 98% 이상 크게 단축된다. 딥러닝 기반 탐지는 지면 제거에 관계없이 일정한 정밀도와 IOU를 유지하며, 지면 제거 적용 시 방향 오차가 소폭 감소하며 연산 시간이 약 6% 단축된다.

표 3은 자율주행차량이 좌회전 경로에서 주차된 목표차량을 인식하여 왜곡 보정 유무가 탐지 성능에 미치는 영향을 비교한 결과이다. 클러스터링에서는 왜곡 보정을 적용하지 않을 경우 IOU가 0.22, 방향 오차가 6.63°로 나타나지만, 보정을

표 3. 왜곡 보정에 대한 객체 탐지 성능 비교.

Table 3. Comparison of object detection performance for undistortion.

Method	IOU	Position Error (m)	Heading Error (°)
Clustering w/o U	0.22	1.48	6.63
Clustering	**0.30**	**1.47**	**2.45**
Deep Learning w/o U	0.44	0.98	1.66
Deep Learning	**0.56**	**0.91**	**1.14**

표 4. 거리별 클러스터링 및 딥러닝 추적 성능 비교.

Table 4. Comparison of clustering and deep learning tracking performance by distance.

Distance (m)	Method	Precision	IOU	Position Error (m)	Heading Error (°)	Velocity Error (m/s)	Tracked Frames	Frag
0 ~ 20	**Clustering**	**1.0**	0.55	**0.30**	7.44	0.47	**100**	**1**
	Deep Learning	0.57	**0.67**	0.75	**4.80**	**0.33**	14	7
20 ~ 40	**Clustering**	**1.0**	0.28	**0.76**	8.84	1.07	**100**	1
	Deep Learning	0.99	**0.44**	0.79	**5.87**	**1.02**	99	1
40 ~ 60	**Clustering**	1.0	0.21	1.57	3.62	1.77	100	1
	Deep Learning	**1.0**	**0.27**	**1.15**	**1.40**	**1.43**	**200**	**0**
60 ~ 80	**Clustering**	**1.0**	0.04	1.71	3.85	**2.62**	**67**	2
	Deep Learning	0.52	**0.10**	**1.59**	**1.93**	2.71	52	**1**
80 ~ 100	**Clustering**	**0.97**	**0.01**	**2.09**	**3.89**	**2.19**	**39**	**4**
	Deep Learning	0.0	N/A	N/A	N/A	N/A	0	N/A

표 5. 통합 추적 성능 평가.

Table 5. Integrated tracking performance evaluation.

Method	Precision	IOU	Position Error (m)	Heading Error (°)	Velocity Error (m/s)	Tracked Frames	Frag
Clustering	0.99	0.25	1.18	5.82	1.48	81	9
Deep Learning	0.67	**0.39**	**0.99**	**3.69**	**1.21**	73	9
Clustering + Deep Learning	0.99	0.35	1.13	3.79	1.46	101	7
Clustering + Deep Learning + Motion Correction	**0.99**	0.37	1.06	3.79	1.46	**101**	**7**

표 6. 제안하는 프레임워크의 모듈 별 연산 시간(ms)

Table 6. Computation time per module of the proposed framework (ms).

	Cloud Segmentation				Object Classification	Tracking		
Statatics	**Ground Removal**	**Undistortion**	**Clustering with Voxelization**	**L-Shape Fitting**	**PointPillars**	**Integration with Filtering**	**Tracking* with Motion Correction**	**Total**
Min	7.4	1.6	9.6	6.1	27	0.089	0.72	37
Max	25	5.3	30	20	91	1.2	26	148
Average	15	3.3	16	10	31	0.25	2.9	**52**

*Track Management + Prediction

적용하면 IOU는 0.30으로 증가하고 방향 오차는 2.45도로 크게 감소한다. 딥러닝 기반 탐지에서도 보정 미적용 시 IOU는 0.44, 방향 오차는 1.66°로 측정되며, 보정을 적용하면 IOU가 0.56으로 증가하고 방향 오차는 1.14° 개선된다.

표 4는 거리별 동일한 길이의 데이터셋을 활용하여, 클러스터링과 딥러닝 기반 추적 성능을 연산 시간을 고려한 실시간 객체 위치를 기준으로 평가한 결과이다. 각 구간의 데이터 수집 프레임을 200으로 설정하여 동일한 조건에서 객체의 추적 시간(tracked frames)과 추적 중단 횟수(fragmentation)를 측정한다. 0~20m 구간에서는 클러스터링과 딥러닝 모두 IOU가 각각 0.55와 0.67로 높은 결과를 보이지만, 거리가 멀어질수록 딥러닝은 IOU를 비교적 안정적으로 유지하는 반면, 클러스터링은 IOU가 급격히 감소한다. 딥러닝은 60~80m 구간에서 정밀도가 0.52로 크게 떨어지며, 80~100m 구간에서 객체 인식이 불가능한 한계를 보인다. 반면 클러스터링은 모든 거리에서 정밀도가 0.97 이상으로 유지되며, 80~100m 구간에서도 평균 39프레임의 추적 성능을 보이지만, IOU와 위치 오차, 상대 속도 계산에서 전반적으로 딥러닝보다 낮은 성능을 보인다.

표 5는 클러스터링, 딥러닝, 클러스터링과 딥러닝 통합, 그리고 움직임 보정을 추가한 통합 기법의 추적 성능을 비교한 결과로, 표 4와 같은 데이터셋을 사용하며 전체 프레임에서의 인식 및 추적 성능을 실시간으로 평가한다. 클러스터링은 정밀도가 0.99로 높지만, IOU는 0.25로 낮으며, 딥러닝은 IOU가 0.39로 클러스터링보다 높지만 정밀도가 0.57으로 낮다. 위치 오차는 딥러닝이 0.99m로 클러스터링보다 높은 정확도를 보이지만, 정밀도 내 평균 추적 시간이 73 프레임으로 클러스터링의 81 프레임에 비해 불안정하다. 이에 두 기법을 통합하여 사용할 경우 클러스터링에 비해 IOU가 0.35로 향상되고 위치 오차는 1.13m로 감소하며, 딥러닝에 비해 정밀도가 0.99로 증가하여 안정성이 향상된다. 움직임 보정을 추가할 경우 IOU가 0.37로 증가하고 위치 오차는 1.06m로 감소하여 인식 정확도가 향상된다.

표 6은 자율주행차량이 그림 17의 경로를 주행하며 수집한 데이터셋을 기반으로 제안된 프레임워크의 전체 연산 시간을 계산하기 위해 각 모듈의 연산 시간을 측정한 결과를 나타낸다. 전처리 과정에 해당하는 지면 제거와 왜곡 보정은 각각 15ms와 3.3ms로 20ms 이내에 처리된다. 이후 복셀화를

적용한 클러스터링과 L자형 피팅에서는 총 26ms가 소요되며, Object Classification 모듈의 객체 탐지 단계는 31ms로 가장 높은 연산 시간을 요구한다. 클러스터링과 딥러닝 기반 객체 인식을 병렬 처리하여 전체 연산 시간을 효과적으로 단축하며, 이를 통해 25ms 이상의 연산 시간을 확보한다. 전체 처리 시간은 52ms로 측정되며, 10hz로 동작하는 라이다의 포인트 클라우드를 100ms 주기 내에서 실시간성을 유지한다.

V. 결론

본 연구는 라이다 센서의 탐지 거리 증가와 해상도 향상으로 인해 급증하는 포인트 클라우드 데이터를 실시간으로 정밀하고 효율적으로 처리하기 위한 새로운 통합 프레임워크를 제안한다. 제안된 프레임워크는 지면 제거, 왜곡 보정, 복셀화, 클러스터링 및 딥러닝 기반 객체 인식을 포함하는 모듈 구조를 기반으로 한다. 이를 통해 라이다 데이터를 효과적으로 전처리하고, 연산량을 줄이며, 데이터의 신뢰도를 높인다. 또한, 클러스터링과 딥러닝 기법을 결합하여 인지 성능 향상과 함께 객체 추적의 안정성을 강화한다. 각 모듈의 유효성과 성능은 단계적인 실험을 통해 검증하며, 이를 통해 연산 시간을 효과적으로 단축하고 객체 추적의 정확성과 신뢰성을 동시에 확보한다.

향후 연구에서는 가속 모델링을 통한 객체 추적 정확도 향상과 다중 객체 실험 및 타 추적 알고리즘과의 비교를 통해 성능을 검증하고자 한다. 더 높은 분해능과 PPS를 제공하는 라이다와 FMCW 기술이 적용된 라이다 및 고체형(solid-state) 라이다에서 실시간성을 평가하여 다양한 라이다 시스템에서의 적용 가능성을 확대하고자 한다.

부록

실험에 사용된 파라미터는 Pandar 64 라이다를 기준으로 실험적으로 설정됨.

복셀화: 복셀 크기 $l = 0.2$

클러스터링: 영역 $N = 5$, 기본 임계값 $\epsilon = 0.5$, 증가 계수 $\alpha = 0.1$

L자형 피팅: 최소 거리 임계값 $\epsilon = 0.1$, 변화량 $\Delta\theta = 1.0°$

통합 및 필터링: IOU 임계값 $\tau = 0.3$, 노드 임계값 $r = 2.0$

추적: 프로세스 노이즈 공분산 $Q = diag(0.1,0.1,0.1,0.1)$, 관측 노이즈 공분산 $R = diag(0.01,0.01,0.01,0.01)$

REFERENCES

[1] S. H. Lee and W. Y. Choi, “Camera and LiDAR sensor fusion method for object estimation of intelligent robots,” *Journal of Institute of Control, Robotics and Systems (in Korean)*, vol. 30, no. 8, pp. 810-816, Aug. 2024.

[2] S. C. Park and S. M. Lee, “A review on LiDAR-based odometry and mapping technology for autonomous robots,” *Journal of Institute of Control, Robotics and Systems (in Korean)*, vol. 29, no. 1, pp. 31-38, Jan. 2023.

[3] Q. Wu, Q. Meng, Y, Tian, Z. Zhou, C. Luo, W. Mao, P. Zeng, B. Zhang, and Y. Luo, “A method of calibration for the distortion of LiDAR integrating IMU and odometer,” *Sensors*, vol. 22, no. 17, pp. 6716, Aug. 2022.

[4] M. McDermott and J. Rife, “Correcting motion distortion for LIDAR scan-to-map registration,” *IEEE Robotics and Automation Letters*, vol. 8, no. 2, pp. 1-6, Apr. 2023.

[5] W. Yang, Z. Gong, B. Huang, and X. Hong, “Lidar with velocity: Correcting moving objects point cloud distortion from oscillating scanning lidars by fusion with camera,” *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 8241-8248, Jul. 2022.

[6] G. Lee and H. Kim, “Point cloud distortion correction in mobile robots using inertial measurement unit integration,” *IEEE International Conference on Control, Automation and Systems (ICCAS)*, pp. 1-6, Oct. 2024.

[7] M. Himmelsbach, F. v. Hundelshausen, and H.-J. Wuensche, “Fast segmentation of 3D point clouds for ground vehicles,” *2010 IEEE Intelligent Vehicles Symposium*, pp. 560-565, Jun. 2010.

[8] M. A. Fischler and R. C. Bolles, “Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography,” *Communications of the ACM*, vol. 24, no. 6, pp. 381-395, Jun. 1981.

[9] P. Narksri, E. Takeuchi, Y. Ninomiya, Y. Morales, N. Akai, and N. Kawaguchi, “A slope-robust cascaded ground segmentation in 3D point cloud for autonomous vehicles,” *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pp. 497-504, Nov. 2018.

[10] D. Zermas, I. Izzat, and N. Papanikolopoulos, “Fast segmentation of 3D point clouds: A paradigm on lidar data for autonomous vehicle applications,” *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5067-5073, May 2017.

[11] H. Lim, S. Hwang, and H. Myung, “ERASOR: Egocentric ratio of pseudo occupancy-based dynamic object removal for static 3D point cloud map building,” *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 2272-2279, Apr. 2021.

[12] H. Lim, M. Oh, and H. Myung, “Patchwork: Concentric zone-based region-wise ground segmentation with ground likelihood estimation using a 3D LiDAR sensor,” *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 6458-6465, Oct. 2021.

[13] S. Lee, H. Lim, and H. Myung, “Patchwork++: Fast and robust ground segmentation solving partial under-segmentation using 3D point cloud,” *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 13276-13283, Oct. 2022.

[14] I. Bogoslavskyi and C. Stachniss, “Fast range image-based segmentation of sparse 3D laser scans for online operation,” *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 163-169, Oct. 2016.

[15] R. B. Rusu and S. Cousins, “3D is here: Point cloud library (PCL),” *2011 IEEE International Conference on Robotics and Automation*, pp. 1-4, May 2011.

[16] Z. Yan, T. Duckett, and N. Bellotto, “Online learning for human classification in 3D LiDAR-based tracking,” *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 864-871, Sep. 2017.

[17] S. Kato, S. Tokunaga, Y. Maruyama, S. Maeda, M. Hirabayashi, Y. Kitsukawa, A. Monrroy, T. Ando, and Y. Fujii, “Autoware on board: Enabling autonomous vehicles with embedded systems,” *2018 ACM/IEEE 9th International Conference on Cyber-Physical Systems (ICCPS),* pp. 287-296, Apr. 2018.

[18] T. Yang, Y. Li, C. Zhao, D. Yao, G. Chen, L. Sun, T. Krajnik, and Z. Yan, "3D ToF LiDAR in mobile robotics: A review," arXiv preprint arXiv:2202.11025, Feb. 2022.

[19] X. Zhang, W. Xu, C. Dong, and J. M. Dolan, "Efficient L-shape fitting for vehicle detection using laser scanners," *2017 IEEE Intelligent Vehicles Symposium (IV)*, pp. 54-59, Jun. 2017.

[20] 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," *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pp. 3517-3523, Nov. 2018.

[21] Y. Zhou and O. Tuzel, "Voxelnet: End-to-end learning for point cloud based 3D object detection," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4490-4499, Jun. 2018.

[22] Y. Yan, Y. Mao, and B. Li, "Second: Sparsely embedded convolutional detection," *Sensors*, vol. 18, no. 10, pp. 3337, Oct. 2018.

[23] S. Shi, Z. Wang, J. Shi, X. Wang, and H. Li, "From points to parts: 3D object detection from point cloud with part-aware and part-aggregation network," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, no. 8, pp. 2647-2664, Aug. 2020.

[24] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom, "Pointpillars: Fast encoders for object detection from point clouds," *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 12697-12705, Jun. 2019.

[25] S.-W. Hong, H.-J. Han, D.-H. Lee, and H.-I. Kim, "LiDAR-based 3D object detection using self-ensemble feature-preserving PointPillars in urban environments," *Journal of Institute of Control, Robotics and Systems (in Korean)*, vol. 28, no. 4, pp. 340-348, Apr. 2022.

[26] S. Shi, X. Wang, and H. Li, "PointRCNN: 3D object proposal generation and detection from point cloud," *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770-779, Jun. 2019.

[27] S. Shi, C. Guo, L. Jiang, Z. Wang, J. Shi, X. Wang, and H. Li "PV-RCNN: Point-voxel feature set abstraction for 3D object detection," *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10529-10538, Jun. 2020.

[28] S.-H. Bae, H.-J. Han, and H.-K. Kim, "μ-Lanelet graph for precise global path planning of robo-taxi," *Journal of Institute of Control, Robotics and Systems (in Korean)*, vol. 27, no. 11, pp. 925-932, Nov. 2021.

[29] Y. T. Chan, A. G. C. Hu, and J. B. Plant, "A Kalman filter based tracking scheme with input estimation," *IEEE Transactions on Aerospace and Electronic Systems*, vol. AES-15, no. 2, pp. 237-244, Mar. 1979.

[30] P. Xiao, Z. Shao, S. Hao, Z. Zhang, X. Chai, J. Jiao, Z. Li, J. Wu, K. Sun, K. Jiang, Y. Wang, and D. Yang, "Pandaset: Advanced sensor suite dataset for autonomous driving," *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*, pp. 3095-3101, Sep. 2021.

이 규 석

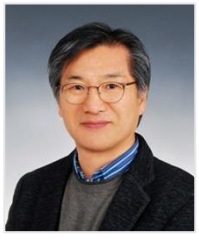
2022년 동양미래대학교 자동화공학과(공학사). 2023년~현재 인하대학교 전기컴퓨터공학과 석사과정. 관심분야는 LiDAR, 3차원 객체 인식, 자율주행.

김 가 나

2021년 인하대학교 정보통신공학과(공학사). 2023년 인하대학교 전기컴퓨터공학과(공학석사). 2023년~현재 인하대학교 전기컴퓨터공학과 박사과정. 관심분야는 자율주행, V2X, 인공지능, 컴퓨터 비전.

이 재 준

2023년 인하대학교 정보통신공학과(공학사). 2024년~현재 인하대학교 전기컴퓨터공학과 석사과정. 관심분야는 딥러닝, 컴퓨터 비전, 자율주행.



김 학 일

1983년 서울대학교 제어계측공학과(공학사). 1985년 Purdue대학교 전기컴퓨터공학과(공학석사). 1990년 Purdue대학교 전기컴퓨터공학과(공학박사). 1990년~현재 인하대학교 스마트모빌리티공학과 교수. 관심분야는 자율주행, 컴퓨터 비전, 패턴인식, 바이오영상처리.