Transactions of KSAE, Vol. 33, No. 5, pp.353-363 (May, 2025)

pISSN 1225-6382 / eISSN 2234-0149
DOI http://dx.doi.org/10.7467/KSAE.2025.33.5.353

〈응 용 논 문〉

자율주행 오픈소스 플랫폼 기반 디지털 트윈 시뮬레이션 환경 구현

이 환 홍[1] · 김 학 주[1] · 기 석 철[*2]

충북대학교 스마트카협동과정[1] · 충북대학교 지능로봇공학과[2]

Implementation of a Simulator Environment Based on an Open-Source Autonomous Driving Platform

Hwan-Hong Lee[1] · Hakjoo Kim[1] · Seok-Cheol Kee[*2]

[1]Department of Smart Car Engineering, Chungbuk National University, Chungbuk 28644, Korea
[2]Department of Intelligent Systems and Robotics, Chungbuk National University, Chungbuk 28644, Korea
(Received 23 December 2024 / Revised 22 January 2025 / Accepted 22 January 2025)

Abstract : This paper presented the construction of a virtual environment for autonomous vehicle experimentation using various open-source platforms and commercial software. Point cloud maps and Lanelet2 maps were designed based on autonomous vehicle and real testbed information, which were then used to generate virtual environment maps for the CARLA simulator. The comparison of the simulation environment with the real environment revealed that the generated virtual maps provide the necessary accuracy for vehicle position tracking, sensor integration, and overall system performance testing, functioning as a reliable platform for autonomous vehicle technology testing. The functionality of the open-source platform and the implemented digital twin environment was verified through various driving scenarios, confirming the autonomous vehicle's reasonable path-following performance. These results demonstrate that using open platforms, such as the CARLA simulator, allows for efficient development and testing of autonomous vehicles in real and simulation environments.

Key words : Digital twin(디지털 트윈), Open-source platform(오픈소스 플랫폼), Autoware(오토웨어), CARLA(칼라), Autonomous driving(자율주행)

Nomenclature

GNSS	: global navigation satellite system
HD	: high definition
HILS	: hardware-in-the-loop-simulation
ICP	: iterative closest point
IMU	: inertial measurement unit
LIO-SAM	: lidar-inertial odometry with slam
MDPS	: motor-driven power steering
MPC	: model predictive control
MRC	: minimal risk condition
NDT	: normal distribution transform
PG	: proving ground
PCD	: point cloud data
ROS	: robot operating system
VILS	: vehicle in the loop simulation

1. 서 론

최근 자율주행 자동차의 수요 증가함에 따라 안전성과 편의성을 확보하기 위한 다양한 기술들이 개발되고 있다. 고도화된 자율주행 자동차의 기술들을 테스트하기 위한 실제 도로 환경 및 인프라들을 반영한 주행 시나리오 설계 및 평가에 대한 중요성 또한 높아지고 있다. 학계와 산업계에서는 실제와 유사한 시뮬레이션 환경 구현에 대한 관심이 커지고 있다. 뿐만 아니라 복잡하고

[*]A part of this paper was presented at the KSAE 2023 Fall Conference and Exhibition
[*]Corresponding author, E-mail: sckee@chungbuk.ac.kr

고도화된 자율주행 알고리즘 개발 및 적용을 위하여 오픈 소스의 활용에 대한 관심이 점점 커지고 있으며, 비용 절감 및 개발 기간 절감을 위해 활용되고 있다.

Dosovitskiy 등[1]은 CARLA 시뮬레이터를 활용하여 자율주행 차량 테스트 환경을 구축하는 방법을 제시하였다. Pirri 등[2]은 협력적 주행 시뮬레이션을 위한 도구와 아키텍처를 제안하였으며, Shan 등[3]은 LiDAR 관성 측위 기법(LIO-SAM)을 통해 매핑 및 위치 추정 성능을 향상시켰다. 이원종과 기석철[4]은 HD 지도를 기반으로 자율주행 PG 내 실험 환경 구축을 하였고, VILS 실험을 통해 자율주행 차량의 제어 성능을 평가하였다. Kato 등[5]은 Autoware 오픈 소스 플랫폼을 임베디드 시스템에 적용하여 자율주행 차량을 구현하였으며, 이 연구는 오픈 소스 플랫폼을 활용한 자율주행 시스템의 구현 방안을 제시한다.

1) ~ 5)의 기존 연구는 제한된 환경과 단순화된 조건에서 자율주행 시스템을 검증하는 데 초점을 맞추었다. 본 논문은 PG 환경에서 일반 주행 및 정적 장애물 회피 시나리오를 통해 가상 환경과 실제 데이터를 비교 검증하여 기존 연구보다 현실적이고 포괄적인 테스트를 수행했다.

Jeong 등[6]은 도시 환경에서 자율주행을 위한 고해상도 지도 생성 방법론을 제시하였고, Kim 등[7]은 부산 EDC 환경에 특화된 디지털 트윈 기반 시뮬레이션 구축, 장준석 등[8]은 디지털 트윈 기반 MRC검증 시스템 개발, 강원율 등[9]은 자율주행 차량 평가를 위한 가상 환경 개발 방법론을 제시하였다.

6) ~ 9)의 특정 지역이나 조건에 특화된 기존 연구는 일반적인 환경에서의 적용 가능성을 충분히 검증하지 못하였다. 본 논문은 특정 지역에 특화되었지만, 실제 도심의 곡률과 유사한 도로 환경과 일반적인 도로 주행 조건을 반영하여 실험을 수행하였다. 이를 통해 환경 적합성을 검증하고, 데이터 일치성을 입증함으로써 기존 연구의 한계를 보완했다.

Chen 등[10]은 HILS 통합 시뮬레이션 시스템을 제안하여 하드웨어와 소프트웨어의 통합 테스트를 하였고, Meng 등[11]은 자율주행 차량의 테스트를 디지털 트윈 이론을 기반으로 수행하는 방법을 제시하며, 가상 환경에서의 자율주행 검증 시스템을 개발하였고, Kim과 Kang[12]은 대규모 협력 테스트를 위한 가상 환경을 구축하였다. Yu 등[13]은 디지털 트윈 기반 자율주행 시스템 개발 패러다임을 제시하였다.

10) ~ 13)의 초기 단계의 하드웨어-소프트웨어 통합 실험을 다룬 기존 연구는 실제 도로 환경에서의 적용 가능성을 구체적으로 평가하지 못한 한계가 존재한다. 본 논문은 실제 데이터를 활용해 구축된 가상 환경과 실제 도로 간의 유사성을 비교하며, 정밀한 실험 결과를 통해 VILS 가능성을 확인했다.

Schwarz와 Wang[14]은 연결된 자율주행 차량에서 디지털 트윈의 역할을 설명하고, 자율주행과 관련된 기술적 요구사항 및 디지털 트윈의 중요성을 강조한다. Chetverikov 등[15]은 Trimmed Iterative Closest Point(ICP) 알고리즘을 제시하였다. 이 알고리즘은 기존 ICP의 단점을 개선하여 이상치(Outliers)에 대해 강건한 성능을 제공하며, 3D 점군 데이터의 정합 정확도를 향상시키고자 하였다. 이는 LiDAR를 활용한 자율주행 차량의 3D 환경 인식 및 맵핑 정확도를 높이는 핵심 기술로 평가된다. Wang 등[16]은 디지털 트윈을 활용한 교통 모델링을 제안하였고 Niaz 등[17]은 디지털 트윈 기반 성능 테스트 방법론을 제안하였다. 이 연구는 디지털 트윈을 활용한 시뮬레이션 및 검증을 다루며, 가상 환경 구축과 시나리오 검증의 중요성을 강조한다. Wang 등[18]은 Unity 게임 엔진을 활용한 디지털 트윈 기반 시뮬레이션 환경을 개발하였고, Kanakagiri[19]는 디지털 트윈을 활용한 가상 시뮬레이션 환경을 구축하였다. 마지막으로, 유찬학 등[20]은 ADAS 점검을 위한 VILS 기반 시뮬레이션 연구를 수행하였다.

14) ~ 20)의 정적 조건 중심의 실험에 머문 기존 연구는 동적 환경이나 복잡한 조건에서의 검증이 부족했다. 본 논문은 기존 연구와 마찬가지로 정적 조건 중심의 실험을 수행했지만, 다양한 곡률 조건과 정적 장애물 회피 시나리오를 포함한 실험을 통해 가상 PG 환경과 실제 PG 환경 간 데이터를 정량적 · 정성적으로 비교 분석했다. 이를 통해 두 환경 간 데이터의 유사성을 확인하고, 기존 연구에서 다루지 않았던 측면을 보완했다.

본 논문에서는 실제 환경에서 테스트 시 발생하는 비용, 시간, 공간, 안전 문제 등의 제약을 해결하기 위해 자율주행 PG인 C-track의[21] 실제 환경 지도를 기반으로 CARLA 시뮬레이터 가상 환경 지도를 구축하는 방법을 제시한다. 수집한 센서 데이터를 기반으로 정밀한 PCD 지도를 제작하고, 차로 중심 정보를 GNSS 데이터를 기반으로 추출하여 Lanelet2 형식의 HD 지도를 작성하였다. HD 지도는 자율주행 오픈 소스 플랫폼인 Autoware universe에서 활용되는 형식으로, 작성된 HD 지도와 상용 소프트웨어인 MathWorks의 RoadRunner와 오픈 소스 플랫폼인 Epic games의 Unreal engine을 활용해 CARLA 시뮬레이터에서 사용할 가상 환경 지도를 제작하였다.

구현된 PG의 가상 및 실제 환경 간 유사도 및 정확도를 평가하고 다양한 자율주행 시나리오를 통해 가상 PG 환경의 신뢰성과 정확성을 확인하였다.

본 논문의 목차는 다음과 같이 구성되었다. 2장에서는

오픈 소스 플랫폼 실험 환경 구현을 기술한다. 3장에서는 디지털 트윈 자율주행 시뮬레이션 환경 구성에 대해서 기술하며, 4장에서는 실험 및 평가를 기술한다. 마지막으로 5장에서는 본 논문의 결론 및 향후 계획을 기술한다.

2. 오픈 소스 플랫폼 실험 환경 구축

2.1 시뮬레이션 환경 구성

본 논문에서는 자율주행 자동차 시스템을 구현하기 위해서 ROS2(Galactic)와 Ubuntu 20.04가 적용되었으며, 오픈 소스 플랫폼인 Autoware universe와 CARLA 시뮬레이터,[1] Unreal engine을 활용하였다.

ROS는 로봇 소프트웨어 개발을 위한 오픈 소스 프레임워크로, 다양한 기능을 제공하는 패키지를 통해 자율주행 차량의 센서 데이터 수집 및 알고리즘 실행을 지원한다. Ubuntu는 안정성과 사용 편의성 덕분에 로봇 및 차량 개발 등 다양한 환경에서 널리 활용된다. CARLA 시뮬레이터는 Unreal engine과 함께 사용되어 사실적인 그래픽과 물리적 특성을 제공하며, 자율주행 알고리즘을 테스트하고 검증할 수 있는 고품질의 가상 환경을 제공하고, 다양한 주행 시나리오 구성과 환경 설정을 지원한다.

Table 1은 실험을 위하여 구성된 시뮬레이션용 PC 정보를 나타내며 Fig. 1은 OS, 미들웨어 등 시스템 구성을 보여준다.

Table 1 Simulation PC information

CPU	Intel i7-13700K
RAM	64 GB
GPU	NVIDIA RTX 3090 Ti
Nvidia driver CUDA	565.57.01 11.6

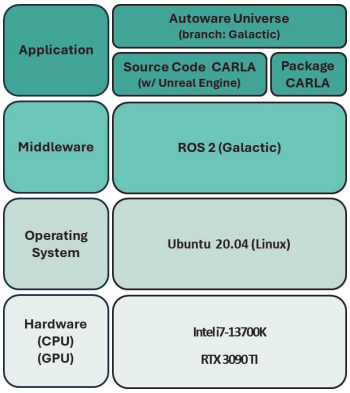
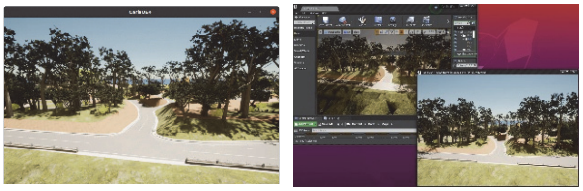


Fig. 1 System configuration for experimenting with autonomous driving open-source platforms

2.2 CARLA 오픈 시뮬레이터

CARLA 시뮬레이터는 다양한 버전을 거치며 많은 연구자와 사용자들에게 검증된 플랫폼으로, 안정성과 신뢰성을 바탕으로 자율주행 연구에서 널리 활용되고 있다. 특히, 활발한 커뮤니티 활동을 통해 문제 해결과 추가 자료를 손쉽게 지원받을 수 있어 연구 진행에 실질적인 도움을 준다. 또한, 다양한 플랫폼과 높은 호환성을 제공하여 여러 환경 및 기술을 결합한 테스트가 가능하다는 점과 더불어, 사용자 정의 지도 제작 기능을 통해 연구 목적에 최적화된 실험 환경을 설계할 수 있어 유연성과 확장성을 갖춘 도구로 평가되어 CARLA 시뮬레이터를 사용하였다. 본 논문에 사용된 모든 CARLA 버전은 0.9.13이다.[22]

Fig. 2의 (a)는 단순히 서버 실행을 위한 Package CARLA이고, (b)는 서버와 함께 Unreal engine을 사용하여 다양한 Asset을 편집할 수 있는 Source code CARLA이다. 본 섹션에서는 오픈 소스 플랫폼 간의 연동을 위하여 Package CARLA를 사용하였다.



(a) Package CARLA (b) Source Code CARLA

Fig. 2 Screen examples of package carla and source code carla

2.3 CARLA - ROS Bridge

오픈 소스인 CARLA-ROS bridge[23]는 CARLA 시뮬레이터와 ROS 간의 통신을 가능하게 하는 중요한 구성 요소로, ROS 토픽을 CARLA 시뮬레이터와 연동하여 센서 데이터 수집, 차량 제어, 시뮬레이션 환경 설정 등을 실시간으로 수행할 수 있게 한다. Fig. 3은 CARLA-ROS bridge의 통신 구조를 나타낸다.[2]

2.4 Autoware 자율주행 오픈 플랫폼

CARLA-ROS bridge를 통해 Package CARLA와 ROS 간의 통신이 연결 후, ROS 기반의 오픈 소스 플랫폼인 Autoware universe를 적용하였다. Autoware는 Tier IV에서 주도적으로 개발되었으며, 이후 Autoware foundation이 설립되어 이 프로젝트의 지속적인 발전과 글로벌 커뮤니티를 지원하고 있다. Autoware foundation은 자율주행 시스템의 개발과 표준화를 촉진하며, 다양한 개발자

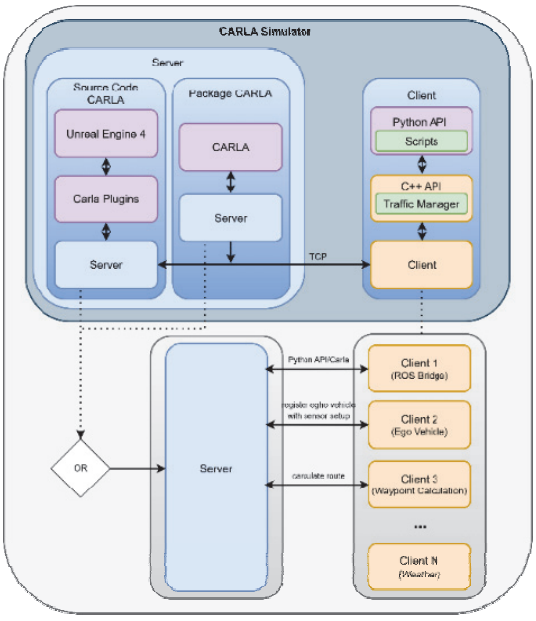


Fig. 3 CARLA - ROS bridge communication structure

와 기업들이 참여하는 오픈 소스 프로젝트를 지원하고 있다. Autoware는 자율주행 차량의 기능 구현을 위한 핵심 소프트웨어 스택을 제공하며, 주로 ROS 2 기반으로 자율주행 알고리즘을 통합하는 데 사용된다. Autoware universe는 Autoware의 최신 버전으로, 자율주행 시스템을 설계, 개발, 테스트하는 데 필요한 다양한 모듈과 기능을 제공하며, 특히 모듈화 된 구조와 확장성을 통해 효율적인 개발 환경을 지원한다. 또한 해당 플랫폼은 CARLA 시뮬레이터와 원활하게 연동되어 자율주행 알고리즘을 테스트하고 검증할 수 있는 환경을 제공한다. 본 논문에서는 Autoware universe에서 제공하는 다양한 버전 중에서 ROS2 Galactic 버전과 호환되는 알고리즘들이 적용되었다.[24]

2.5 Autoware - CARLA 연동 실험

Autoware universe는 CARLA ROS Bridge를 포함하는 op_carla 패키지를 활용하여 실행된다.[26] op_carla는 CARLA 시뮬레이터와 Autoware universe 간의 원활한 연동을 제공하며, 이를 통해 자율주행 시스템을 테스트하고 검증할 수 있는 기본 시뮬레이션 환경을 구축할 수 있다. 기본 환경 시뮬레이션은 두 가지 방법으로 실험할 수 있다. 첫 번째는 Package CARLA를 통해 서버를 실행하고, CARLA-ROS bridge를 통해 ROS와 연동하여 수동 주행 및 다양한 기능을 사용하는 방법이며, 두 번째는 op_carla 패키지를 사용하여 CARLA-ROS bridge와 Autoware universe를 연동하여 기존 CARLA-ROS bridge

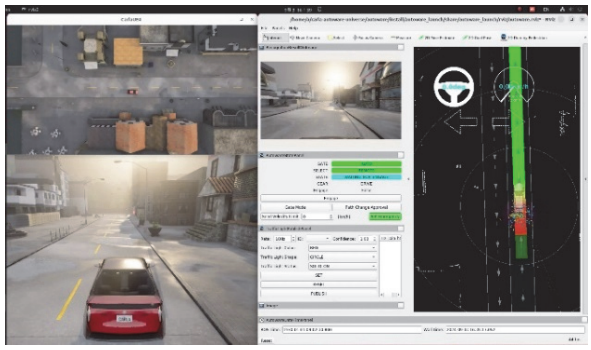


Fig. 4 Autonomous driving system test environment with carla simulator and autoware universe

의 기능을 Autoware universe와 함께 활용하는 방법이다.

Fig. 4는 CARLA 시뮬레이터에서 기본 제공되는 Town01의 PCD 지도, HD 지도와 두 번째 방법을 통해 CARLA 시뮬레이터에서 단순 차로 주행 시나리오를 수행하는 결과를 나타낸다.

3. 디지털 트윈 자율주행 시뮬레이션 환경 구축

자율주행 시스템을 안전하고 검증하기 위하여 가상 PG 환경을 구현하였다. PG는 실제 자율주행 자동차의 로직을 검증하고 평가하는 장소로 다양한 연구소 및 학교에서 운영되고 있다. 시간 및 공간적인 제약으로 인해 실제 PG를 사용하기 어려운 경우와 실제 PG에서 실험 이전에 실제 PG와 유사한 환경을 가지고 있는 가상 PG에서의 실험은 합리적인 검증을 위하여 필수적이다. 실제 PG에 대한 지도를 제작하고, 이를 기반으로 상용 소프트웨어인 MathWorks사의 RoadRunner와 Tier IV의 Vector map builder, Source code CARLA, Unreal engine을 활용하여 CARLA 시뮬레이터에서 구현될 가상 PG 환경을 제작하였다.

3.1 실제 환경 지도 제작

실제 PG에서의 PCD 지도 작성을 위하여 LIO-SAM 알고리즘[3]이 사용되었으며, HD 지도를 작성하기 위하여 Vector map builder가 활용되었다.[25] LIO-SAM은 LiDAR와 IMU 데이터를 결합하여 정밀한 위치 정보를 생성하며, 강도 정보를 통해 도로의 형태, 차선 요소들을 파악할 수 있다. Vector map builder는 도로 네트워크 정보를 기반으로 HD 맵을 생성하는 데 사용되며, 이를 통해 차량이 주행할 수 있는 안전하고 효율적인 경로 생성이 가능하다. Table 2는 LIO-SAM 알고리즘에서 사용된 센서들의 정보를 나타내고, Fig. 5는 차량 제원과 차량에 부착된 센서들의 위치를 나타낸다.

Table 2 Configuration sensors for real-world mapping

LiDAR	Ouster OS 1 - 32 ch
GNSS receiver GNSS antenna RTK	PwrPak7D™ VEXXIS® GNSS-500 Series Antennas MRD-1000v2
IMU	MicroStrain 3DM-GV7-AHRS

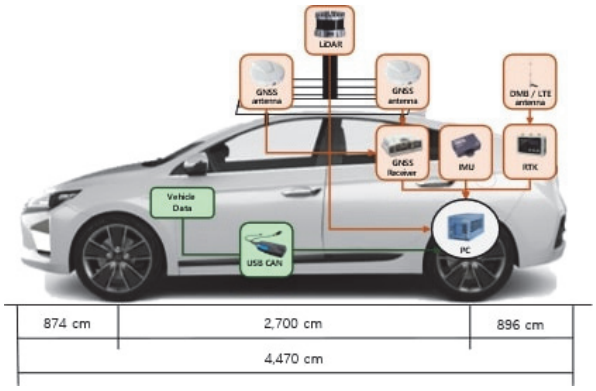


Fig. 5 Vehicle and sensor placement

생성된 지도를 Fig. 6의 과정처럼 CloudCompare 소프트웨어[27]를 사용하여 회전 변환 행렬을 계산한 후, 자율주행 PG의 오프셋 값 반영 후, 이를 회전시켜 지도를 조정하였다. 이후, CloudCompare 내재된 ICP 알고리즘[15]을 활용하여 PCD 지도를 개선하였다.

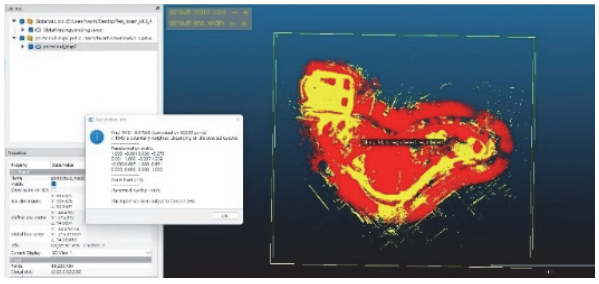


Fig. 6 Finding a rotation transformation matrix in CloudCompare

완성된 PCD 지도를 기반으로 Vector map builder에서 도로 네트워크 정보를 생성한다. Vector map builder를 활용하여 Lanelet2 형식의 HD 지도를 작성하기 위하여 각 차로의 중앙선을 csv 파일로 작성한 후, 이를 Lanelet2 지도 형식으로 변환하였다. Fig. 7은 이 과정을 통해 생성된 자율주행 PG의 일부 HD 지도를 보여준다.

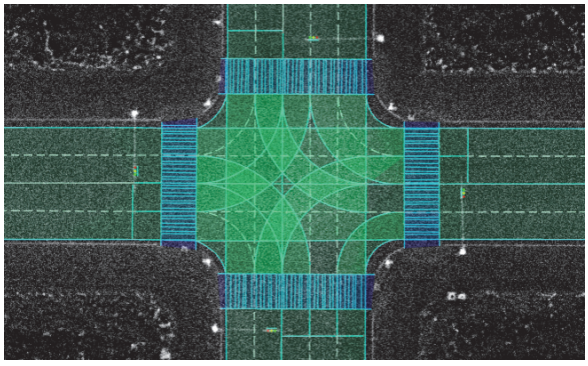


Fig. 7 Selected hd maps in autonomous PG

3.2 CARLA 시뮬레이터 지도

CARLA 시뮬레이터 지도는 앞서 제작한 PCD 지도와 HD 지도를 기반으로, MathWorks의 RoadRunner와 Epic games의 Unreal engine을 활용해 CARLA에서 사용할 가상 환경 지도를 제작했다. RoadRunner는 고해상도의 도로 환경을 생성하는 상용 소프트웨어이며, Unreal engine은 사실적인 그래픽과 물리적 특성을 구현하고, 이를 필요한 확장자로 변환하는 데 사용되었다. 이 두 도구의 조합으로 자율주행 차량이 실제 도로와 유사한 환경에서 주행할 수 있도록 가상 환경을 설계하였다.

추가적으로, RoadRunner 환경에서는 Fig. 8과 같이 PCD 지도와 HD 지도를 동시에 표시하여 도로 환경을 제작했으며, 최종적으로 fbx, xodr 확장자를 가진 파일을 생성하였다. 가상 PG 환경 구현을 위하여 2.2절에서 언급된 Source code CARLA를 활용하였다. Source code CARLA는 Unreal engine과 사용이 가능하며, 이 때, Unreal engine은 4.26 버전을 사용했다. RoadRunner에서 변환된 파일들을 Source code CARLA에 적용하여 가상 PG 환경을 생성하였다. Fig. 9는 최종적으로 만들어진 가상 PG 환경을 나타낸다.

Fig. 8 A scene with a point cloud map and a Lanelet2 map in the Roadrunner environment

Fig. 9 The resulting virtual PG environment

3.3 가상 환경 지도 제작

3.1절에서와 마찬가지로, LIO-SAM 알고리즘을 사용하여 가상 환경 내에서 PCD 지도를 작성하였다. 다만, 3.1절에서는 실제 환경에서 LIO-SAM을 활용하여 제작한 반면, 본 절에서는 CARLA 시뮬레이터 환경 내에서 시뮬레이션 데이터를 기반으로 지도를 제작하였다. 또한, HD 지도는 실제 환경에서 사용된 Lanelet2 형식을 적용하여, 가상 환경에서도 동일한 도로 네트워크 정보를 사용할 수 있도록 했다.

4. 실험 및 평가

4.1절은 실제 환경에서 작성된 PCD 지도와 HD 지도, 그리고 가상 환경에서 생성된 지도를 비교하였다. 자율주행 알고리즘의 정확성을 평가하였다. 4.2절과 4.3절에서는 각각 차로 주행, 정적 장애물 회피 시나리오를 통해 다양한 곡률 조건 상황에서 주행 시스템의 안정성을 검증하였다.

4.1 지도 비교

실험의 첫 번째 단계로 PCD 지도, HD 지도를 시각적으로 비교하였다. 이 과정에서는 도로의 형태와 차선 정보를 중심으로 두 지도의 유사성을 평가했다. 특히, 나무와 같은 비정형적으로 생성된 주변 환경 요소들을 제외한 도로 환경의 정확도를 검토하여 자율주행 차량이 도로를 올바르게 인식할 수 있는지 확인했다.

Fig. 10과 Fig. 11은 동일한 HD 지도를 활용하여 각각의 환경에서 생성된 PCD 지도와 비교하며, 본문에서 두 환경 간의 매칭 정도를 정성적으로 보여주기 위해 사용되었다. Fig. 10은 실제 환경에서 수집된 PCD 지도와 HD 지도의 비교를, Fig. 11은 CARLA 환경에서 생성된 PCD 지도와 HD 지도의 비교를 나타낸다.

Fig. 10, Fig. 11의 (a)는 PG 내 도심로 교차로, (b)는 순환로의 갓길 버스정류장, (c)는 주행성능시험장의 코너

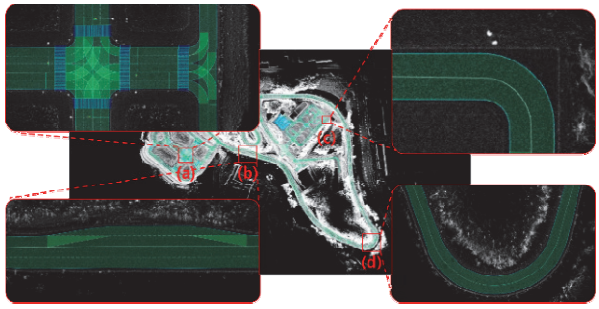


Fig. 10 Real-world PCD Map and HD Map

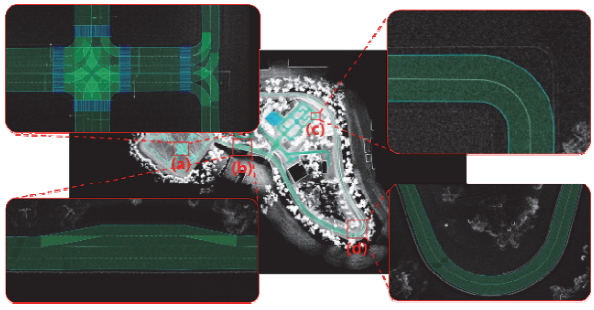


Fig. 11 CARLA Virtual Environment PCD Map and Real-world HD Map

구간, (d)는 순환로의 헤어핀 구간을 확대해서 보여준다. Fig. 11은 CARLA 환경에서 생성된 PCD 지도와 실제 환경에서 사용되는 두 환경의 지도 일치 정도를 정성적으로 확인할 수 있다.

실제 환경과 CARLA 환경에서 생성된 PCD 지도 데이터는 각각 약 6,600만 개와 6,200만 개의 포인트로 구성되어 있으며, 이 중 6,000만 개의 포인트 클라우드 데이터에 대해 ICP 알고리즘을 적용하였다. 결과는 Table 3에 정리되어 있으며, T와 R 행렬은 각각 병진 변환과 회전 변환을 나타낸다. ICP 알고리즘을 통해 두 환경 간 인접한 포인트들 간의 위치 차이를 정량적으로 평가한 결과, RMS 값은 약 0.038 m로, 두 데이터 간 높은 유사도와 정

Table 3 ICP matching results between PCD maps

Parameter	Value
Transformation Matrix (T)	$\begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix}$
Rotation Matrix (R)	$\begin{bmatrix} 1.000 & 0.003 & -0.000 \\ -0.003 & 1.000 & -0.001 \\ 0.000 & 0.001 & 1.000 \end{bmatrix}$
Translation Vector (t)	$\begin{bmatrix} 1.689 \\ -0.001 \\ 0.016 \end{bmatrix}$
RMS	*0.038351 [m]*

합도를 확인할 수 있었다.

4.2 전역 경로 추종 시나리오

본 절에서 언급된 전역 경로 추종 시나리오는 가상 PG 환경과 실제 PG 환경에서 일반 주행 성능 평가를 위해 순환로를, 급격한 회전 성능 평가를 위해 도심로를 활용하여 수행되었다.

Table 4 Normal distribution transform parameters

Parameters	Division	Value	Unit
Step size	Real vehicle	0.1	[-]
	Virtual vehicle	0.3	
Regulation scale factor	Real vehicle	0.01	[-]
	Virtual vehicle	0.01	
Convergence criteria	Real vehicle	0.01	[-]
	Virtual vehicle	0.01	
Resolution	Real vehicle	3.0	[m]
	Virtual vehicle	2.5	

Table 5 Model predictive control parameters

Control gains	Decription	Environment	Value
N_p	Predictions steps	Real	50
		Virtual	50
Δt_c	Control period	Real	0.01
		Virtual	0.1
Q	Weight matrix	Real	$\begin{bmatrix}0.1 & 0 & 0\\ 0 & 0.0 & 0\\ 0 & 0 & 0.3\end{bmatrix}$
		Virtual	$\begin{bmatrix}1.0 & 0 & 0\\ 0 & 0.0 & 0\\ 0 & 0 & 0.3\end{bmatrix}$
R	Weight matrix	Real	1.0
		Virtual	1.0
τ_c	Contstnat time delay	Real	0.24
		Virtual	0.0

Table 4는 Autoware universe의 ndt_scan_matcher 알고리즘[28]에서 사용되는 파라미터를 나타내며, 가상 환경과 실제 환경은 비정형으로 생성된 주변 환경의 차이로 인해 각각 다른 NDT 파라미터가 적용되었다.

Table 5는 MPC 기반 횡방향 제어 알고리즘의 주요 파라미터 값을 나타낸다. 가상 환경과 실제 환경의 차량 동역학 모델이 서로 다르기 때문에, 두 환경에서 차량의 거동을 유사하게 만들기 위해 가상 PG 환경과 실제 PG 환경에 각각 다른 파라미터가 적용되었다. 추후, 동일한 차량 동역학 모델로 변경하여 동일한 파라미터를 적용할 계획이다.

Fig. 12와 Fig. 14는 전역 경로 추종 시나리오에서 PG 내 순환로와 도심로에 해당하는 가상 및 실제 자율주행

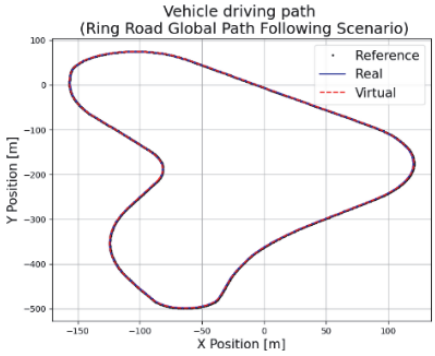


Fig. 12 Vehicle driving paths in a ring road global path following scenario

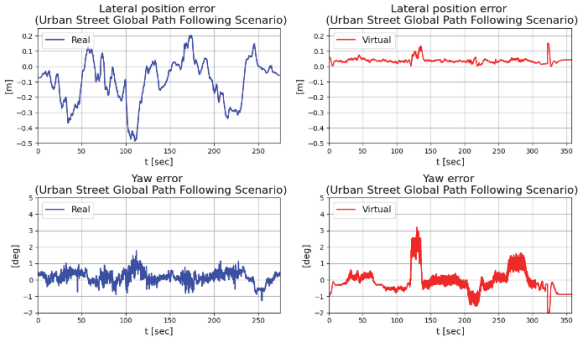


Fig. 13 Lateral position error and yaw error of vehicles in a ring road global path following scenario

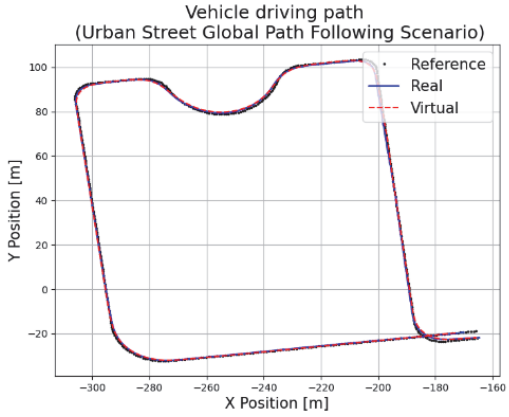


Fig. 14 Vehicle driving paths in an urban street global path following scenario

자동차의 주행 경로를 나타낸다. 우측 상단의 각주는 각각 기준 경로(첫 번째), 실제 자율주행 자동차의 주행 경로(두 번째), 가상 자율주행 자동차의 주행 경로(세 번째 점선)로 표시된다. 순환로에서는 일반적인 주행 상황에서 곡률에 대한 주행 평가를 진행하며, 도심로에서는 급격한 좌회전 및 우회전 상황에서 곡률에 대한 주행 평가를 수행한다.

Fig. 13, Fig. 15는 전역 경로 추종 상황에서 발생된 횡방향 오차 및 요 각도 오차를 나타낸다. 사진의 왼쪽의 그래프는 실제 자율주행 자동차에 대한 횡방향 오차 그래프이고, 오른쪽의 그래프는 가상 자율주행 자동차의 횡방향 오차 그래프이다.

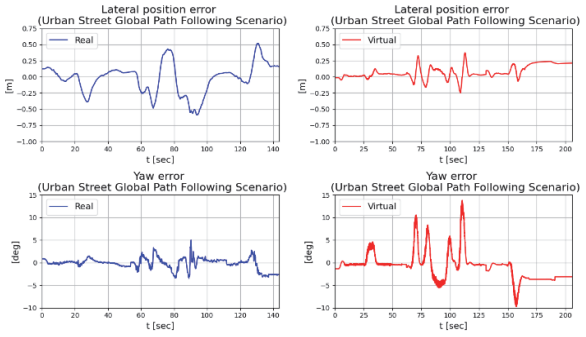


Fig. 15 Lateral position error and yaw error of vehicles in an urban street global path following scenario

이러한 경향은 Table 6의 오차의 정량적 지표에서 유사한 값을 나타내었다. 가상 환경과 실제 환경의 오차는 제어 이득 값, 차량 동역학 모델링의 차이로 인해 서로 달라질 수 있다. 표에 나타난 두 환경 각각의 오차에 대한 차이는 두 환경에서의 차량이 서로 다른 제어 이득 값을 적용한 것과 실제와 가상 환경 간 차량 모델링의 차이에 의한 결과로 판단되어진다.

Table 6과 Table 7은 4.2절 전역 경로 추종 시나리오에 대한 횡방향 오차, 요 각도 오차의 RMS(Root Mean Squares), STD(Standard Deviation), Max(Absolute MAX) 값을 나타낸다.

향후, 언급된 문제를 해결하기 위해 실제 자율주행 자동차의 제원 및 동역학적 파라미터를 가상 자율주행 자동차에 정교하게 접목하여, 제어 이득 값을 동일하게 적용했을 때 유사한 주행 성능을 확보할 계획이다. 또한, 차량 모델을 보다 정밀하게 설정하면 두 환경 간 유사도가 더욱 향상될 것으로 예상된다.

Table 6 Lateral position error result in global path following scenario

Case	Global path following scenario			
Error type	Lateral position error			
Area	Ring road		Urban street	
Environment	Real	Virtual	Real	Virtual
RMS	0.1737	**0.0423**	0.2347	**0.1253**
STD	0.1440	**0.0182**	0.2337	**0.1000**
MAX	0.4869	**0.1507**	0.5912	**0.3736**

Table 7 Yaw error result in global path following scenario

Case	Global path following scenario			
Error type	Yaw error			
Area	Ring road		Urban street	
Environment	Real	Virtual	Real	Virtual
RMS	**0.3995**	0.7011	**1.2598**	3.1224
STD	**0.3639**	0.6939	**1.2497**	3.0083
MAX	**1.7860**	3.2092	**4.9796**	13.7859

4.3 정적 장애물 회피 시나리오

정적 장애물 회피 시나리오는 4.2절 전역 경로 추종 시나리오와 동일하며, 순환로와 도심로 모두 직진 구간에서 정적 장애물을 회피하는 비슷한 양상을 보인다고 판단하여, 순환로에서의 정적 장애물 회피만 진행하였다. 순환로에서는 두 개의 정적 장애물을 회피하고, 정적 장애물은 주행 경로 상 두 개의 차선이 존재하는 구간에 가상의 차량을 설치하여 구현하였다. 가상의 정적 장애물의 데이터들은 가상 PG 환경에서 Autoware universe의 인지 알고리즘인 lidar_centerpoint를 활용하여 생성하였다.

Fig. 16은 자율주행 시스템의 실제 환경과 가상 환경

Fig. 16 Real-world and virtual environment validation experiments for autonomous driving systems

에서의 검증 실험 장면을 보여준다. 가상의 정적 장애물은 실제 PG 환경과 동일한 위치에 존재했고, 장애물을 회피하는 주행을 성공적으로 수행하였다. 이를 통해 VILS 시스템의 가능성을 확인하였으며, 가상 PG 환경이 실제 환경과 높은 유사성을 갖는다는 것을 확인하였다.

Figs. 17, 18은 4.2절의 순환로에 해당하는 주행 경로와 동일 하지만, 정적 장애물의 위치가 추가되어 있으며, 정적 장애물의 위치를 상세히 표시하고 이에 대한 회피 주행 경로를 보여준다.

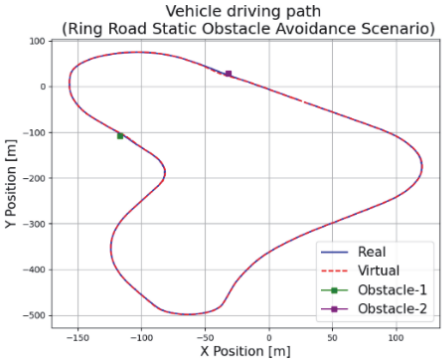


Fig. 17 Vehicle driving path and static obstacle locations in a ring road static obstacle avoidance scenario

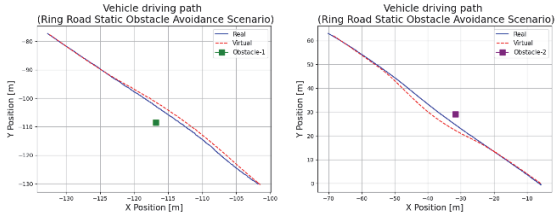


Fig. 18 Static obstacle location in a ring road static obstacle avoidance scenario [Zoom-in]

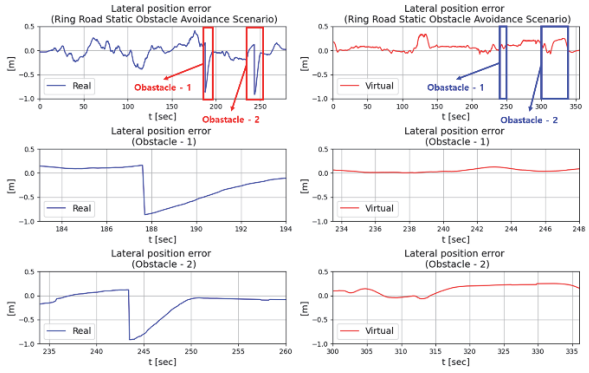


Fig. 19 Lateral position error of vehicles in a ring road static obstacle avoidance scenario

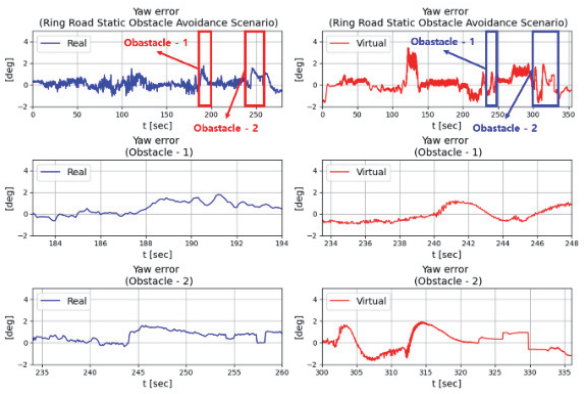


Fig. 20 Yaw error of vehicles in a ring road static obstacle avoidance scenario

Fig. 19는 정적 장애물 회피 시나리오 주행 전반에 걸친 횡방향 오차와 회피 상황에서의 횡방향 오차를 나타내며, Fig. 20은 전반적인 상황의 요 각도 오차와 회피 상황의 요 각도 오차를 보여준다. Figs. 19, 20의 경향은 Table 8, 9의 결과와 유사하다.

실제 환경에서는 회피 순간 큰 횡방향 오차가 발생하는데, 이는 정적 장애물 회피 직후 지역 경로가 바로 변경되는 현상으로 인해 발생하는 현상이며, 이후에는 안정적으로 수렴하는 양상을 확인할 수 있었다. 이는 알고

Table 8 Lateral position error result in static obstacle avoidance scenario

Case	Static obstacle avoidance scenario			
Area	Ring road			
Error type	Lateral position error			
Obstacle number	Obstacle - 1		Obstacle - 2	
Environment	Real	Virtual	Real	Virtual
RMS	0.3950	**0.0586**	0.2996	**0.1677**
STD	0.3350	**0.0335**	0.2600	**0.1018**
MAX	0.8674	**0.1248**	0.9161	**0.2505**

Table 9 Yaw error result in static obstacle avoidance scenario

Case	Static obstacle avoidance scenario			
Area	Ring road			
Error type	Yaw error			
Obstacle number	Obstacle - 1		Obstacle - 2	
Environment	Real	Virtual	Real	Virtual
RMS	0.8093	**0.6584**	**0.7898**	0.9129
STD	**0.6038**	0.6442	**0.4788**	0.9071
MAX	1.7787	**1.1988**	**1.5891**	1.9189

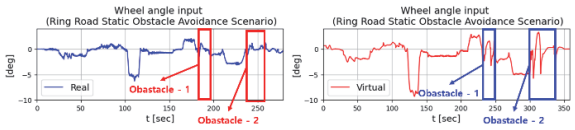


Fig. 21 Wheel angle input of vehicles in a ring road static obstacle avoidance scenario

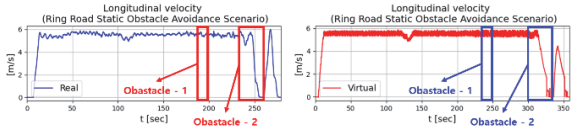


Fig. 22 Longitudinal velocity of vehicles in a ring road static obstacle avoidance scenario

리즘 개선을 통한 성능 향상이 가능할 것으로 보인다.

Fig. 21은 Autoware의 Kinematic MPC에서 계산된 전반적인 정적 장애물 회피 시나리오에서의 제어 입력을 나타낸다. 계산된 제어 입력을 가상 및 실제 자율주행 자동차의 MDPS 값으로 인가하기 위하여 스팅어링 휠의 최대값, 타이어 조향 각도의 최대 값을 활용하였다. Fig. 22는 차량의 종방향 속도를 나타낸 것으로, 직진 구간에서 약 5 ~ 6 m/s로 주행하였고, 회전이 있는 구간에서 5 m/s보다 작은 속도로 주행하였으며, 회피 순간의 상황에서는 종방향 속도가 5 ~ 6 m/s로 유지되었다가 목표지점에 도달 시 0 m/s에 수렴하는 것을 보여준다. 종방향 속도가 0 m/s로 나타나는 구간은 자율주행 자동차의 Closed-loop를 구현하기 위해 목표 위치를 여러 번 지정한 결과이다.

Table 8, 9는 4.3절에 해당하는 시나리오의 결과로 각각 정적 장애물 회피 순간에 대한 횡방향 오차, 요 각도 오차의 RMS, STD, MAX 값을 나타낸다.

실제 자율주행 자동차와 가상 환경 자율주행 자동차의 거동은 조향각과 종방향 속도에서 노이즈 크기와 값의 범위에서 차이를 보였지만, 가상 환경 자율주행 자동차의 횡방향, 종방향 행동은 실제 환경 자율주행 자동차와 유사한 결과를 나타냈다. 이는 Fig. 15에서 언급한 바와 같이 두 환경에서의 자동차 동역학 모델 차이로 인해 발생한 문제로, 향후 도출할 계획이다.

5. 결 론

본 논문의 기여는 다음과 같다.

- 가상 PG 환경 구현 및 정밀 지도 제작:
  실제 PG 환경에서 활용되는 OSM 파일형식의 HD 지도와 PCD 지도를 기반으로 Road runner에서 정지선, 신호등 등 다양한 도로 시설들이 포함된 가상 PG 환경을 구현하였다.
- 자율 주행 시나리오 검증 및 안전성 평가:
  제작된 PCD 지도, HD 지도를 가상 환경에 적용하여 경로 추종, 정적 장애물 회피 시나리오를 통해 가상 PG 및 실차 PG 디지털 트윈 환경에서 자율주행 자동차의 주행 안전성을 검증하였다.

본 연구에서는 Autoware universe와 CARLA 시뮬레이터를 활용하여 자율주행 시스템의 테스트 및 검증을 위한 가상 환경을 구축하고, 다양한 주행 시나리오에서 성능을 평가하였다. 실험 결과, 제작된 가상 PG 환경과 실제 PG 환경에서 활용되는 HD 지도와 PCD 지도의 높은 정확성을 확인할 수 있었으며, 다양한 주행 시나리오를 통해 구현된 가상 PG 환경의 정확성과 자율주행 알고리즘의 주행 성능을 검증하였다.

그러나 도심로의 곡률이 큰 회전 구간에서 제어 문제로 인해 예상보다 높은 오차율과 실제와 비슷하지 않은 경향을 보이는 문제가 확인되었다 이는 가상 환경뿐만 아니라 실제 환경에서도 유사한 문제가 발생할 수 있음을 알 수 있으며, 두 환경 간 제어 파라미터는 같지만, 가상환경의 차량모델이 아직 실제 자율주행 자동차의 역학 모델을 완벽히 반영하지 못했기 때문에 실제 환경과 가상 환경에서의 오차 결과가 서로 차이가 난 것으로 보여진다. 따라서 향후 실제 자율주행 자동차의 모델을 가상 환경의 자율주행 자동차의 역학 모델로 만드는 것에 목표가 있다. 또한 제어 문제를 해결하기 위해 알고리즘 경량화, 가상 환경 정밀도 개선, 추가 시나리오 검증을 통해 자율주행 시스템의 성능과 신뢰성을 강화하고자 한다. 본 연구를 통해 구축한 자율주행 시스템이 효과적으로 작동함을 확인하였으며, 지속적인 연구를 통해 자율주행 기술의 안전성과 신뢰성을 강화할 것이다. 이러한 연구 결과는 향후 자율주행 기술 발전에 중요한 기초 자료로 활용될 것으로 기대된다.

후 기

이 논문은 2024년도 정부(산업통상자원부)의 재원으로 한국산업기술기획평가원의 지원을 받아 수행된 연구임(RS-2024-00408818).

References

1) A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez and V. Koltun, “CARLA: An Open Urban Driving Simulator,” Conference on Robot Learning, PMLR, pp.1-16, 2017.
2) P. Pirri, C. Pahl, N. El Ioini and H. R. Barzegar, “Towards Cooperative Maneuvering Simulation:

Tools and Architecture," 2021 IEEE 18th Annual Consumer Communications & Networking Conference (CCNC), pp.1-6, 2021.
3) T. Shan, B. Englot, D. Meyers, W. Wang, C. Ratti and D. Rus, "LIO-SAM: Tightly-Coupled LiDAR Inertial Odometry via Smoothing and Mapping," IEEE/RSJ International Conference on Intelligent Robots and Systems(IROS), pp.5135-5142, 2020.
4) W. Lee and S. C. Kee, "Implementation of VILS Systems for Autonomous Driving Testbed Based on HD Map," Transactions of KSAE, Vol.31, No.7, pp.503-511, 2023.
5) S. Kato, S. Tokunaga, Y. Maruyama, S. Maeda, M. Hirabayashi, Y. Kitsukawa, A. Monrroy, T. Ando, Y. Fujii and T. Azumi, "Autoware on Board: Enabling Autonomous Vehicles with Embedded Systems," ACM/IEEE 9th International Conference on Cyber-Physical Systems(ICCPS), pp.287-296, 2018.
6) J. Jeong, J. Y. Yoon, H. Lee, H. Darweesh and W. Sung, "Tutorial on High-Definition Map Generation for Automated Driving in Urban Environments," Sensors, Vol.22, No.18, Paper No.7056, 2022.
7) M. S. Kim, J. H. Park and M. S. Sim, "A Study on Real-Time Autonomous Driving Simulation System Construction Based on Digital Twin – Focused on Busan EDC," Journal of Cadastre & Land InformatiX, Vol.53, No.2, pp.53-66, 2023.
8) J. S. Jang, B. H. Kwon, K. W. Park and J. Hong, "Development of Simulation Environment for Verification of Digital Twin-Based MRC(Minimal Risk Condition) Decision System," Proceedings of Symposium of the Korean Institute of Communications and Information Sciences, pp.600-603, 2022.
9) W. Kang, J. Jo, M. Lee, D. Kang, M. Hyun and S. Heo, "A Study on the Methodology to Develop Virtual Drive Environment for Autonomous Driving Evaluation," Transactions of KSAE, Vol.29, No.6, pp.547-556, 2021.
10) Y. Chen, S. Chen, T. Zhang, S. Zhang and N. Zheng, "Autonomous Vehicle Testing and Validation Platform: Integrated Simulation System with Hardware in the Loop," 2018 IEEE Intelligent Vehicles Symposium(IV), pp.949-956, 2018.
11) Z. Meng, S. Zhao, H. Chen, M. Hu, Y. Tang and Y. Song, "The Vehicle Testing Based on Digital Twins Theory for Autonomous Vehicles," IEEE Journal of Radio Frequency Identification, Vol.6, pp.710-714, 2022.
12) B. G. Kim and E. Kang, "Toward Large Scale Test for Autonomous Driving Software in Collaborative Virtual Environment," IEEE Access, Vol.11, pp.72641-72654, 2023.
13) B. Yu, C. Chen, J. Tang, S. Liu and J. -L. Gaudiot, "Autonomous Vehicles Digital Twin: A Practical Paradigm for Autonomous Driving System Development," Computer, Vol.55, No.9, pp.26-34, 2022.
14) C. Schwarz and Z. Wang, "The Role of Digital Twins in Connected and Automated Vehicles," IEEE Intelligent Transportation Systems Magazine, Vol.14, No.6, pp.41-51, 2022.
15) D. Chetverikov, D. Svirko, D. Stepanov and P. Krsek, "The Trimmed Iterative Closest Point Algorithm," 2002 International Conference on Pattern Recognition, Vol.3, pp.545-548, 2002.
16) S. -H. Wang, C. -H. Tu and J. -C. Juang, "Automatic Traffic Modelling for Creating Digital Twins to Facilitate Autonomous Vehicle Development," Connection Science, Vol.34, No.1, pp.1018-1037, 2022.
17) A. Niaz, M. U. Shoukat, Y. Jia, S. Khan, F. Niaz and M. U. Raza, "Autonomous Driving Test Method Based on Digital Twin: A Survey," International Conference on Computing, Electronic and Electrical Engineering(ICE Cube), pp.1-7, 2021.
18) Z. Wang, K. Han and P. Tiwari, "Digital Twin Simulation of Connected and Automated Vehicles with the Unity Game Engine," 2021 IEEE 1st International Conference on Digital Twins and Parallel Intelligence(DTPI), pp.1-4, 2021.
19) A. Kanakagiri, "Development of a Virtual Simulation Environment for Autonomous Driving Using Digital Twins," Ph.D. Thesis, Technische Hochschule Ingolstadt, Ingolstadt, Germany, 2021.
20) C. Yu, J. Jung, H. Lee, Y. Gwon and H. Lee, "A Study on the Establishment of VILS Based on Simulation for ADAS Inspection," Transactions of KSAE, Vol.30, No.11, pp.873-879, 2022.
21) Chungbuk National University C-Track, https://www.youtube.com/watch?v=9FAdXlwPCgU
22) https://github.com/carla-simulator/carla/releases/tag/0.9.13/
23) https://github.com/carla-simulator/ros-bridge
24) https://autowarefoundation.github.io/autoware-documentation/galactic/installation/
25) https://github.com/hatem-darweesh
26) https://docs.web.auto/en/user-manuals/vector-map-builder/introduction
27) https://cloudcompare-org.danielgm.net/release/
28) https://autowarefoundation.github.io/autoware.universe/latest/localization/autoware_ndt_scan_matcher