주차환경에서 자율 주행 차량 실험을 위한 V-Rep 기반 시뮬레이터

임규범· 김민성· 안준우· 김민수· 박재흥

서울대학교 융합과학부

V-Rep Simulator for Autonomous Vehicle Research in Parking lot Environment

GyuBeom Im· Minsung Kim· Joonwoo Ahn· Minsoo Kim· Jaeheung Park

Graduate School of Convergence Science and Technology, Seoul National University

Abstract: We propose a V-Rep vehicle simulator for autonomous vehicle testing in parking lot environment. When we validate a new algorithm or method in the real world, it could be dangerous and lead to safety accident or emergency situation. Therefore, it is beneficial to validate a new algorithm in a virtual simulator before doing in the real world. V-Rep simulator is a popularly used robot simulator. We can create our test environment and validate some algorithms using this simulator. And V-Rep is enable to interact with ROS (Robot Operating System) as a part of virtual parking lot environment. In this paper, we explain how to set up a parking lot environment and an autonomous vehicle model. And we explain how to validate planning, control algorithms using V-Rep and ROS.

Key words: Autonomous Vehicle (자율 주행 자동차), Parking lot Environment (주차장 환경), Vehicle Simulator (차량 시뮬레이터)

1. 서 론

실제 자율주행차량에 검증되지 않은 새로운 알고리즘을 실험할 경우 안전사고나 돌발상황 등 다양한 위험요인이 존재한다. 시뮬레이터의 장점은 안전사고 위험 없이 다양한 환경에서 실험할 수 있다는 점이다. 시뮬레이터의 필요성이 증가하면서 기존의 다양한 자율주행차량 시뮬레이터 프로그램들이 제작되었다[1)2)3)7)]. Mechanical Simulation에서 제작한 CarSim[2)], TASS Inter-national에서 제작한 PreScan[3)] 차량 시뮬레이터는 차량의 다양한 물성치를 변경하며 실험할 수 있다는 장점이 있지만 라이센스 가격이 비싸며 환경을 구축하고 사용하기까지 많은 노력이 필요하다[4)]. 또한, 새로운 알고리즘이나 기능을 적용할 때 사용자가 직접 해당 플랫폼에 맞게 다시 프로그래밍하고 설정해야 하는 불편함이 있다[5)6)]. 이러한 플랫폼 의존성 문제를 해결하기 위해 각각의 알고리즘을 모듈화해서 관리할 수 있는 미들웨어 프로그램인 ROS (Robotics Operat-ing System)가 등장하였다[9)].

최근 간단하면서도 자율주행차량 실험에 필요한 기능들이 구현되어 있는 시뮬레이터들이 많이 제작

(a) 차세대융합기술원 주차장 모형 (b) Sketchup을 통해 제작한 주차장 환경

Fig. 1 Google Sketchup을 통해 제작한 광교 차세대융합기술연구원 주차장의 모습. (a)는 주차장 모형이고 (b)는 Sketchup을 통해 제작한 환경이다.

임규범 E-mail: edwardim@snu.ac.kr.

되었다. 예를 들면, Udacity Self driving Car Simulator, 오픈소스 기반의 Torcs Simulator, OSRF의 Gazebo Simulator, Coppelia의 V-Rep Simulator[7]가 있고 Unreal Game Engine 기반의 AirSim, Carla, Deep Drive2와 같은 시뮬레이터들이 있다. 이 중 ROS와 연동 가능한 시뮬레이터는 V-Rep, Gazebo, Carla이다.

본 논문에서는, V-Rep 시뮬레이터를 사용한 자율주행차량 주차장 환경 시스템을 제안한다. 2장에서는 Google Sketchup으로 주차장 환경을 제작하는 방법과 자율주행차량의 크기 설정 및 센서 배치도에 대해 설명하고, 3장에서는 V-Rep과 ROS 간 연동 과정에 관해 기술한다. 4장에서는 결론으로 마무리를 짓는다.

2. V-Rep 시뮬레이터 주차환경 환경설정

본 시뮬레이터 환경은 크게 주차장 환경과 자율주행차량 모델로 구성되어 있다. 주차장 환경은 Google Sketchup 프로그램을 사용해 제작했으며 이를 V-Rep 상에서 불러와 사용했다. 또한, V-Rep에서 기본적으로 지원하는 Simple Ackermann Steering 차량 모델을 사용해 차량의 기구학적 움직임을 가지는 모델을 사용했으며 여기에 Vision 센서와 LIDAR 센서, IMU 센서, GPS 센서 등을 부착했다.

2.1 주차장 환경 제작

주차장 환경은 경기도 수원시 영통구 광교로에 위치한 차세대융합기술연구원 주차장 환경을 토대로 Sketchup을 사용해 제작했으며 실측을 통해 실제와 유사한 크기의 맵을 제작했다. Sketchup은 Google에서 제공하는 3D Modeling 전문 소프트웨어로써 건물, 공원, 주차장 같은 환경을 만들기 용이하다[8]. 또한, 3D Warehouse라는 모델 공유 사이트를 제공하여 직접 차량이나 건물, 나무 등을 만들지 않고도 쉽게 다운받아서 구현 가능한 장점이 있다. 다운받은 오브젝트들은 3D Warehouse를 통해 불러와 실제 크기에 맞도록 조정한 뒤 배치했다. 최종적으로 제작된 주차환경의 모습은 Fig. 1 과 같다.

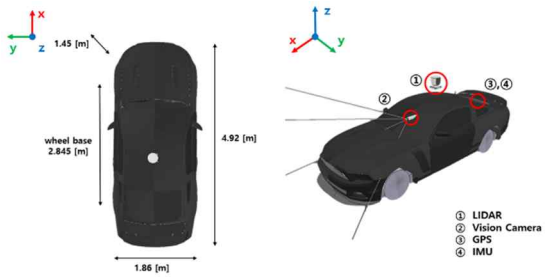

Fig. 2 V-Rep에서 사용한 자율주행차량 모델의 크기 및 센서 배치도. 차량의 크기는 Hyundai 그랜져 HG240 모델의 크기를 기반으로 설정했다. 센서는 Velodyne HDL-64E LIDAR 센서와 Vision, GPS, IMU 센서를 사용했다.

2.2 자율주행차량 모델 및 센서 설정

자율주행차량의 기본 모델은 차량의 Non Holonomic한 움직임 특성을 제공하는 모델을 사용했으며 바퀴 4개와 몸체로 구성되어 있다. 모델 위에 3D Warehouse를 통해 받은 차량의 샤시 파일을 올린 후 Fig. 2 왼쪽 그림과 같이 Hyundai 그랜져 HG240 모델을 기반으로 크기를 수정했다. 차량 모델의 바퀴의 지름은 0.634 [m]이고 후륜 바퀴를 구동모터로 주행한다. 조향은 전륜 바퀴가 회전하며 좌우 바퀴 기준 0.46, 0.60 [rad] (또는 0.60, 0.46 [rad])이 최대 조향각이다. 차량의 크기 및 바퀴의 크기는 V-Rep 환경에서 조정 가능하며 조향각 또한 원하는 수치로 조정할 수 있다. 차량의 센서 배치도는 Fig. 2 오른쪽 그림과 같다.

차량 중앙에 Velodyne HDL-64E LIDAR 센서를 부착했으며 전방에 Vision Camera를 부착했다. 그리고 차체 뒤쪽에 GPS 센서와 IMU 센서를 부착했다. Velodyne HDL-64E LIDAR 센서는 차량의 무게중심 기준으로 (x=-0.5,y=0,z=1.47)[m]에 장착되어 있으며 가로 방향 360° 각도를 4096 등분하며 세로 방향 −24.8° 의 각도를 64 등분하여 Point Cloud 데이터를 받아오도록 설정했다. 또한 Vision Camera는 차량 무게중심 기준으로 (x=0.5,y=0,z=1.0)[m]에 장착되어 있고 FOV (Field of View)는 90° 이며,

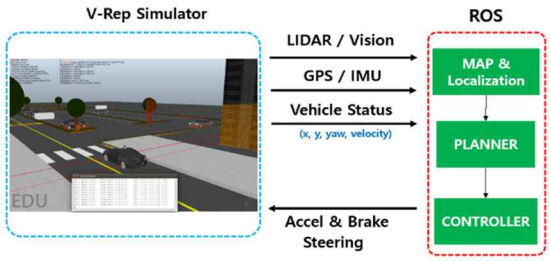

Fig. 3 V-Rep과 ROS의 연동 모습. V-Rep은 하나의 ROS 노드로써 작동하며 센서 데이터들이 토픽으로 퍼블리시되고 Planner와 Controller를 거쳐서 최종적으로 나온 Acceleration, Brake, Steering을 섭스크라이브 받는다.

해상도는 640x480으로 설정했다. 센서는 세부 옵션을 변경할 수 있고 차량 내 장착 위치와 개수를 사용자가 직접 선택할 수 있다. 모든 센서의 위치는 서울대학교 동적로봇시스템 연구실 자율주행차량의 센서 배치도를 참고했다.

3. V-Rep과 ROS 연동

이번 장에서는 V-Rep과 ROS를 연동하는 과정에 대해 설명한다. ROS(Robotics Operating System)란 로봇 기술 개발용 프레임워크로써 로봇에서 사용하는 다양한 프로그램들 간의 통신을 도와주는 미들웨어 프로그램이다[9]. ROS의 실행 단위는 노드(Node)이고 각각의 노드는 독립적으로 존재한다. 노드가 데이터를 받는 과정을 섭스크라이브(Subscribe), 전송하는 과정을 퍼블리시(Publish)라고 하며 데이터를 전송하는 통로를 토픽(Topic)이라고 한다. 또한, 토픽으로 전송되는 데이터 유형을 메세지(Message)라고 한다. ROS를 사용하면 자율주행차량의 복잡한 시스템을 모듈화해서 관리할 수 있으며 여러 유용한 시각화 도구 (RViz, rqt 등) 를 지원하므로 프로그램의 유지 및 보수가 용이하다.

V-Rep은 최신 3.5.0 버전부터 ROS를 기본적으로 지원한다. V-Rep을 실행하면 자동으로 ROS에서 하나의 노드로써 동작한다. LIDAR, Vision, GPS, IMU 등 센서 데이터는 특정한 토픽 이름으로 퍼블리시된다. 또한, 차량의 현재 위치 및 속도 [x, y, yaw, velocity] 값도 토픽으로 퍼블리시된다. V-Rep에서 퍼블리시하는 각각 토픽의 이름과 메세지 타입은 다음과 같다.

- **Velodyne HDL-64E LIDAR Sensor**
  - /velodyne_points (sensor_msgs::PointCloud2)
- **Vision Camera Sensor**
  - /front_camera (sensor_msgs::Image)
- **GPS Sensor**
  - /gps_data (geometry_msgs::Vector3Stamped)
- **IMU Sensor**
  - /imu (sensor_msgs::Imu)
- **Vehicle Status [x,y,yaw,velocity]**
  - /LocalizationData (std_msgs::Float32MultiArray)
- **Global Frame & Vehicle Frames**
  - /tf (geometry_msgs::TransformStamped)
- **Simulation Time**
  - /clock (rosgraph_msgs::Clock)

차량의 현재 속도와 조향각은 V-Rep Auxiliary Viewer를 통해 실시간으로 확인할 수 있으며 ROS을 통해 퍼블리시되므로 다른 ROS 노드에서 섭스크라이브가 가능하다. 또한, 전역 좌표계와 차량 무게중심 좌표계, 센서 좌표계, 차량 바퀴 좌표계 등이 ROS 토픽으로 퍼블리시되어 다른 ROS 노드에서 섭스크라이브할 수 있다.

V-Rep에서 섭스크라이브 받는 토픽의 이름과 메시지 타입은 다음과 같다.

- **Steering Angle [degree]**
  - /SteerAngleData (std_msgs::Float32MultiArray)
- **Acceleration & Brake [m/s]**
  - /VelocityData (std_msgs::Float32MultiArray)

Fig. 2에서 Planner와 Controller를 거쳐서 최종적으로 계산된 조향각과 차량의 속도 값은 위 두 토픽으로부터 섭스크라이브 받는다.

4. 결론

본 논문에서는 주차환경에서 자율주행차량 실험을 위한 V-Rep 시뮬레이터 환경을 제안하였다. Google Sketchup을 사용하여 주차장 환경을 제작하는 방법에 대해 설명했고 자율주행차량의 크기 및 센서 배치도에 대해 설명했다. 자율주행차량의 복잡한 시스템을 모듈화해서 관리할 수 있는 미들웨어 프로그램 ROS에 대해 설명했고 V-Rep과 ROS를 연동하여 시뮬레이터와 자율주행차량 알고리즘을 독립적으로 유지 및 보수할 수 있도록 했다.

본 V-Rep 차량 시뮬레이터 환경은 github을 통해 누구나 쉽게 사용할 수 있도록 공유하고 있다. 주소는 다음과 같다.

https://github.com/tigerk0430/V-Rep-Autonomous-Vehicle-Simulator

또한, Velodyne Sensor와 IMU Sensor는 V-Rep에서 기본적으로 지원하지 않으므로 플러그인 형식으로 추가해야 사용할 수 있다.

https://github.com/tigerk0430/vrep_plugin_velodyne 또는 vrep_plugin_imu

플러그인은 C++ 코드로 작성되어 있으며 센서의 다양한 설정 값을 변경할 수 있다. 예를 들면, Velodyne 센서는 데이터의 범위, 해상도, 채널 수, 데이터 전송 속도, 좌표계, Visualize or hidden 여부, IMU 센서는 데이터의 좌표계, 노이즈 등을 설정할 수 있다.

현재 차량 모델은 단순한 Ackermann Steering 모델이므로 추후 서스펜션 모델을 추가하거나 타이어의 탄성을 구현하는 등의 개선의 여지가 있다. 또한, 주차장 환경뿐 만 아니라 고속도로, 도시 등 다양한 환경을 제작할 수 있다. 많은 사람들이 본 논문의 시뮬레이터를 사용하여 시뮬레이션 초기 환경설정의 어려움과 플랫폼에 의존적인 개발 환경에 구애 받지 않고 자율주행 연구에만 집중할 수 있기를 기대해본다.

References

1) 양관석, 박진현, 정충민, 서명원, 황성호. (2012). 영상처리 알고리즘 검증을 위한 실시간 차량주행 시뮬레이터 개발. 한국자동차공학회 추계학술대회 및 전시회, , 2274-2278.

2) Richard Johansson, David Williams, Anders Berglund, and Pierre Nugues. 2004. Carsim: a system to visualize written road accident reports as animated 3D scenes. In Proceedings of the 2nd Workshop on Text Meaning and Interpretation (TextMean '04). Association for Computational Linguistics, Stroudsburg, PA, USA, 57-64.

3) PreScan, TASS International https://tass.plm.automation.siemens.com/prescan

4) 황성호, 유동연, 김영갑, 한재훈. (2017). 시뮬레이터를 활용한 자율주행자동차 기술. 기계저널, 57(7), 50-54.

5) Rausch, Viktor, et al. "Learning a deep neural net policy for end-to-end control of autonomous vehicles." American Control Conference (ACC), 2017. IEEE, 2017.

6) 김성준, 박용운, 윤주홍, 주상현. (2015). 자율주행용 지상이동체의 라이다 센서 탑재 시뮬레이션. 한국측량학회 학술대회자료집, 233-234.

7) E. Rohmer, S. P. N. Signgh, M. Freese, "V-REP: a Versatile and Scalable Robot Simulation Framework,'' IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS), 2013

8) Aidan Chopra. 2012. Introduction to Google Sketchup (2nd ed.). Wiley Publishing.

9) Quigley, Morgan, et al. "ROS: an open-source Robot Operating System." ICRA workshop on open source software. Vol. 3. No. 3.2. 2009.