오픈 소스 자율주행 플랫폼 Autoware 이해와 이식

숭실대학교 ▌최규진·이효은·백한나·구성우·김강희*

1. 서 론

최근에 자율주행차에 대한 사회적 관심이 높아지는 상황에서 자율주행 SW 플랫폼에 대한 수요도 함께 커지고 있다. 스마트폰에 Android 플랫폼이 있듯이, 자율주행차에 손쉽게 이식할 수 있는 자율주행 SW 플랫폼이 있다면 자율주행 기술에 대한 연구와 개발은 비약적으로 빨라질 것이다. 플랫폼은 연구자들과 개발자들이 동일한 인터페이스를 사용하여 컴포넌트 기술을 개발하는 것을 가능하게 하기 때문에, 불필요한 커뮤니케이션을 없애고 기술 개발에 집중하는 것을 가능하게 한다. 이런 추세를 반영하여 최근에 중국 Baidu사는 Apollo[1]라는 플랫폼을, 일본의 나고야 대학은 Autoware[2]라는 플랫폼을 오픈 소스로 공개하였다.

Autoware는 3차원 포인트 맵와 라이더 센서를 이용한 자차 위치 추정(localization), 라이더 또는 카메라 기반 장애물 탐지(obstacle detection), 벡터 맵 기반 경로 계획(path planning), 경로 추종(path following) 등 자율주행 S/W의 핵심 기능들을 제공한다. Autoware는 저자들이 판단하기로 SAE 레벨 3 수준의 자율 주행, 즉 차선 유지(lane keeping), 적응형 순항 제어(adaptive cruise control), 장애물 회피를 위한 차선 변경(lane change) 등의 기능을 제공한다. autoware.org에 따르면 Autoware는 20개 이상의 차량 모델들에 이식되었고, 30개 이상의 국가들, 100개 이상의 회사들에 의해서 폭넓게 사용되고 있다. 현재 Autoware 소스 코드는 나고야 대학에서 TIER IV라는 회사를 거쳐 Autoware Foundation이라는 비영리 기관으로 이관되어 관리되고 있으며, 2019년 8월 현재 1.12 버전이 최신 버전으로 제공되고 있다. Autoware는 ROS (Robot Operating System) 1.0 버전 위에서 개발되었으며, 현재 ROS 2.0 버전으로 이식을 준비하고 있다.

본 기고문은 Autoware를 실차량에 이식하고 구동할 때 고려해야 할 기술적인 문제들에 대해서 기술하고자 한다. 저자들은 Autoware를 현대 i30 차량에 이식하여 현대자동차가 주최하는 자율주행 경진대회 AVC 2019에 참여한 경험이 있다. 이 경험을 통해서 Autoware는 실차량을 구동하는데 충분한 기술들을 플랫폼에 내재하고 있음을 확인할 수 있었다. 요약하면, Autoware를 구동하기 위해서는 다음 사항들이 요구됨을 확인하였다.

첫째, Autoware는 고정밀 3차원 포인트 맵을 기반으로 한, 맵기반(Map-based) 자율주행만을 지원한다. 3차원 포인트 맵을 사용하지 않는 맵리스(Mapless) 자율주행 기능은 지원하지 않는다.

둘째, Autoware는 전역 경로 계획을 경로점들의 배열로 표현되는 주행유도선들을 포함하는 벡터 맵에 의지한다. 벡터 맵이 주어지지 않는 주차장과 같은 환경에 대해서는 A* 알고리즘을 통해서 경로점을 동적으로 생성할 수 있으나, 일반 도로 환경에서의 전역 경로 계획은 벡터 맵이 주어져야만 한다.

셋째, Autoware는 차량의 전장 시스템과 연결되는 CAN 게이트웨이가 제공된다고 가정한다. Autoware의 최종 제어 신호는 차량의 타겟 속도 v와 타겟 조향 각속도 ω (또는 타겟 조향각 δ)으로 주어진다. 따라서 타겟 속도 v를 차량 가속도 명령으로 변환하는 PID (Proportional-Integral-Derivative) 제어 알고리즘만 작성하면, Autoware에 CAN 게이트웨이를 바로 연결할 수 있다.

넷째, Autoware 구동에 있어서 GPU가 필수적인 사항은 아니다. 그러나 YOLO와 같은 영상 기반 장애물 탐지 알고리즘을 수행하고자 하면, 쓸만한 성능을 얻기 위해서 GPU가 필요하다. Autoware는 GPU를 장착한 PC에서 구동할 수도 있고, NVIDIA Drive PX2 보드에서 구동할 수도 있다.

* 종신회원

† 본 연구는 과학기술정보통신부 및 한국연구재단의 기초연구사업(No. NRF-2017R1A2B4005763)의 지원을 받아 수행되었음.

다섯째, Autoware를 효과적으로 다루기 위해서는 ROS 프로그래밍에 대한 이해가 필요하며, Docker와 같은 리눅스 컨테이너 기술도 이해해야 한다.

Autoware는 약 25만줄의 소스 코드로 구성되어 있으며, 지금 이 시간도 계속적인 업데이트가 이루어지고 있다. 따라서, 본 기고문에서 설명하는 내용이 모든 버전들에 적용된다고 보장할 수는 없다. 참고로, 저자들은 i30 차량에 Autoware 1.10 버전을 이식하여 사용하였다.

본 기고문은 다음과 같이 구성되었다. 2절에서 Autoware의 주요 컴포넌트들을 설명한다. 3절에서 CAN 게이트웨이가 없는 경우에 이것을 대신하기 위해 저자들이 개발한 차량 제어 프레임워크를 기술한다. 4절에서 Autoware가 요구하는 벡터 맵을 제작하는 방법에 대해서 서술한다. 5절에서 결론을 맺는다.

2. Autoware 구조

그림 1은 Autoware 구조를 도식화한 것이다. Autoware는 크게 자차 위치 추정, 장애물 탐지, 경로 계획, 경로 추종, 이상 4가지 모듈들로 구성된다.

첫째, 자차 위치 추정은 GPS와 라이다 센서를 함께 사용함으로써 센티미터 수준의 정밀도를 제공한다. 자차 위치 추정 모듈은 GPS로부터 차량의 위도값과 경도값을 수신하여 자차 위치에 대한 힌트 정보로 삼고, 이 힌트 좌표를 3차원 포인트 맵과 함께 NDT (Normal Distribution Transform)[3] 매칭 알고리즘에 제공하여 자차 위치와 방향을 포함한 자차 포즈(pose)를 결과값으로 출력하게 된다. NDT 매칭 알고리즘은 포인트 맵을 voxel 단위로 나누고, 하나의 voxel 안에 존재하는 모든 포인트들의 좌표들을 평균하여 무게중심점 좌표를 계산한다. 그리고 무게중심점 좌표로부터 각 포인트들까지의 차이값을 계산하여 분산을 구한다. 결과적으로 하나의 voxel은 3차원 정규분포(확률의 등고선들)로 표시되며, 서로 다른 voxel은 서로 다른 정규분포들로 표현된다. NDT 매칭은 라이더 스캔으로 얻어진 포인트 집합을 임의의 수평 이동과 임의의 회전 변환을 통해서 변환했을 때 얻어진 좌표들을 확률의 등고선들에 매칭해보고, 각 좌표가 위치하는 등고선의 확률값, 즉 매칭 확률들을 합산하여 그 합이 최대화가 되는 수평 이동과 회전 변환 값, 결과적으로 자차의 갱신된 위치를 뉴턴 최적화 방법으로 찾아낸다.

둘째, 장애물 탐지는 객체 탐지와 객체 추적으로 구성된다. 자차 주변에 존재하는 장애물들의 위치와 크기를 파악해야만 뒤에서 설명할 주행 상태 기계가 주행을 위해 계획된 후보 궤적들 중에서 어느 궤적을 선택할지 결정할 수 있다. 객체 탐지 모듈은 카메라 영상 처리 알고리즘(예: YOLO[4])에 의해 얻어진 이미지 객체들과 라이다 스캔들에 대한 점군 분할 알고리즘(예: 유클리디언 클러스터링[5])에 의해 얻어진 점군 객체들을 상호 연관시킴으로써 객체들의 위치, 크기, 분류 클래스(보행자, 자전거, 이동 차량 등)를 출력하며, 객체 추적 모듈은 이 객체들에 대해 칼만 필터(Kalman filter) 알고리즘을 이용하여 다음 타임 스텝에서의 객체들의 위치와 속도를 추정한다. 여기서 중요한 것은, 객체들은 차량을 기준으로 한 3차원 좌표계 안에서 모두 한계 상자(bounding box)들로 표현되며, 뒤에 설명할 주행 상태 기계의 입력으로 사용된다는 점이다.

셋째, 경로 계획은 전역적 계획과 지역적 계획으로 나뉜다. 전역적 계획은 벡터 맵, 시작 포즈, 목표 포즈를 입력받아 주행 기준이 되는 기준 경로(reference path)를 계산한다. 벡터 맵이란 3차원 공간 안에서 차

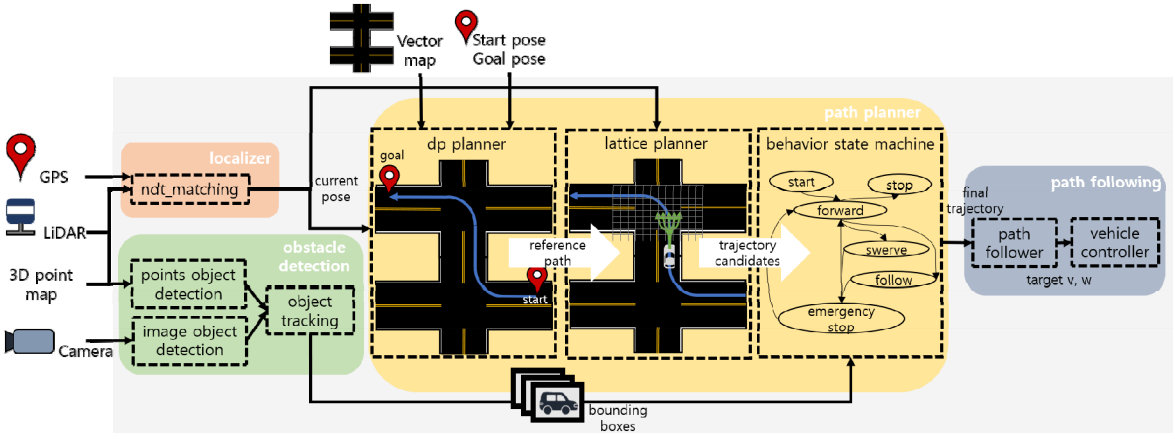


그림 1 Autoware의 구조

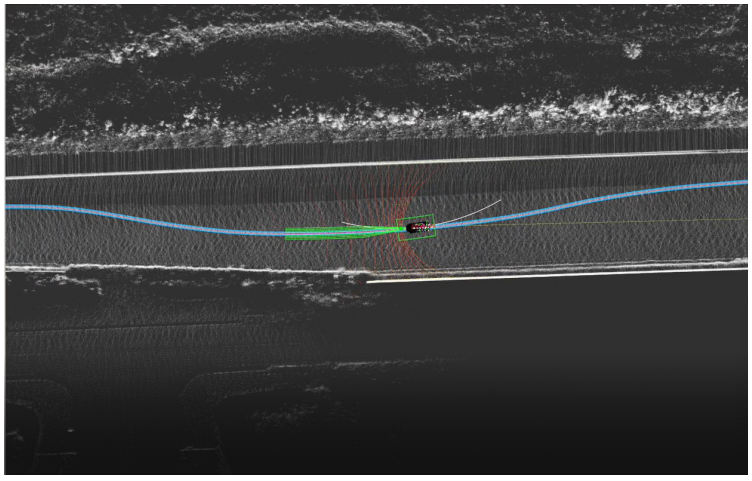


그림 2 wf_simulator를 이용한 Autoware 시뮬레이션 화면

선, 주행 유도선, 정지선, 횡단 보도, 교통 표지판/신호등 위치 등을 점 형식이 아니라 벡터 형식으로 표현한 지도이다. 벡터 맵은 차량 주행이 단순히 장애물을 회피하는 것만으로는 충분치 않고 다양한 교통법규들을 지켜야 한다는 요구사항 때문에 필요하다. 그러나 벡터 맵는 포인트 맵과 다르게 센서들을 이용해서 자동적으로 생성할 수 없고, vector mapper[6]와 같은 도구를 이용하여 수작업으로 제작해야 한다. 자율주행 S/W를 구동하는데 있어서 가장 큰 장애물들 중 하나는 주어진 포인트 맵과 동기화된 벡터 맵을 제작하는 일이다. 벡터 맵 안에서 기준 경로를 계산하는 알고리즘은 동적 프로그래밍(dynamic programming)을 기반으로 한다. 벡터 맵은 일반적으로 차선이 있는 도로의 주행에 필요하다고 간주되지만, 차선이 없는 주차장 주행을 위해서도 사용될 수 있다.

지역적 경로 계획은 기준 경로를 입력받아 기준 경로와 평행한 여러 개의 후보 궤적들을 미리 생성한다. 실내 주행 로봇과 다르게 차량은 고속으로 주행할 수 있기 때문에, 차량이 돌발 장애물을 인식한 경우에 이것을 회피하는 기준 경로를 실시간으로 다시 계산하는 것은 어렵다. 따라서 지역적 경로 계획은 매 순간 후보 궤적들을 여러 개 생성하고, 주행 상태 기계(behavior state machine)는 한계 상자 형태로 전달된 장애물들을 회피하는 후보 궤적을 매 순간 선택함으로써 자율 주행을 실현한다. 후보 궤적 집합은 lattice planner[7]에 의해서 공간 격자(space lattice) 안에서 가능한 모든 격자 교차점들을 각기 다음 경로점(waypoint)로 지정하는 궤적들을 생성함으로써 얻어진다. 주행 상태 기계는 매 순간 새로 생성되는 후보 궤적 집합 안에서 최종 궤적(final trajectory)을 선택하여 출력한다.

마지막으로, 경로 추종은 경로 계획 모듈이 제공한 최종 궤적을 작은 시간 단위들로 나누어 차량이 안정적으로 추종하기 위한 타겟 속도 v와 타겟 조향 각속도 ω를 계산한다. 매 순간 타겟 속도는 사용자 입력 속도와 지역적 경로 계획 알고리즘이 강제하는 가속, 등속, 감속 단계들에 의해서 결정된다. 한편, 타겟 조향 각속도는 타겟 속도를 고려하여 순수 추종(Pure Pursuit)[8] 알고리즘에 의해서 계산된다. 최종적으로 타겟 속도 v는 PID 제어 알고리즘에 의해서 타겟 가속도로 변환되고, 타겟 가속도 a와 타겟 조향 각속도 ω는 CAN 게이트웨이를 통해 차량 전장 시스템에 전달되어야 한다.

Autoware는 실차량이 없는 경우에 시뮬레이터를 사용하여 실행시킬 수도 있다. 가장 간단한 시뮬레이터는 물리 세계에 대한 시뮬레이션 없이 차량 진행 과정만을 시뮬레이션하는 wf_simulator(waypoint follower simulator)이다. Autoware 소스 코드에 포함되어 있으며 실행 오버헤드가 매우 작아서 Autoware 기능 확인에 자주 사용된다. 이외에 물리 세계를 시뮬레이션하는 Gazebo[9], LGSVL[10] 등이 있는데, 이것은 실행 오버헤드가 매우 커서 시뮬레이터만을 위해서 별도의 컴퓨터가 필요한 수준이다.

그림 2는 K-city의 고속주행로 구간에 가상적으로 그려진 주행유도선을 추종하는 Autoware 시뮬레이션 화면이다. 청색선은 Autoware가 생성한 기준 경로이고, 녹색선들은 후보 궤적들이다. 그리고 백색선은 순수 추종 알고리즘이 계산한 곡률을 보여준다.

3. 차량 제어 프레임워크

차량 제어 프레임워크는 차량 전장 시스템에 접속하

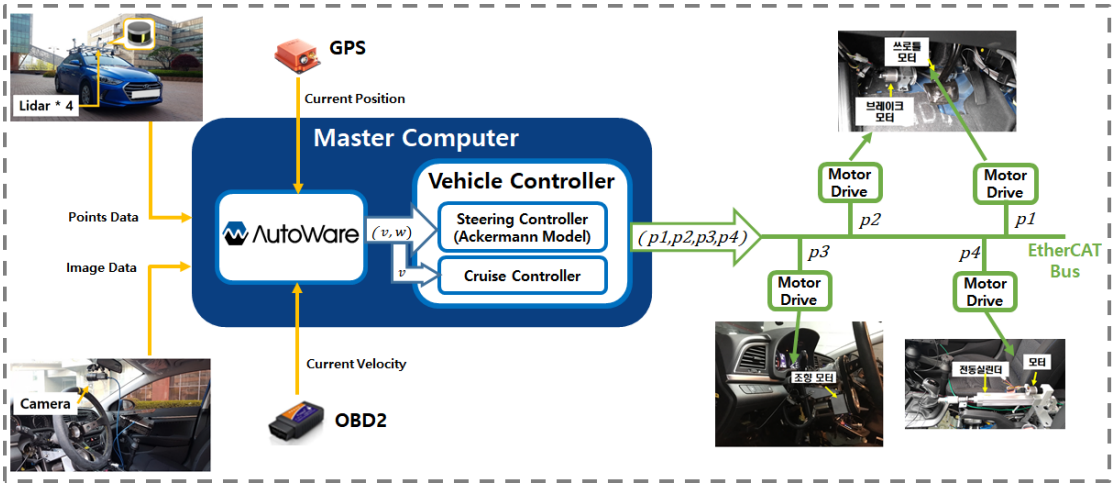


그림 3 Autoware에 연동되는 차량 제어 프레임워크

는 CAN 게이트웨이가 제공되지 않는 경우에 Autoware가 30 Hz마다 출력하는 타겟 속도 v와 타겟 조향 각속도 ω를 집행할 목적으로 개발되었다. 저자들은 AVC 2019에 참가하기 위해서 현대 i30 차량을 위한 차량 제어 프레임워크를 제작했는데, 이것은 이전에 현대 Avante 차량을 위해서 저자들이 개발한 프레임워크[11]와 사실상 동일한 것이다.

그림 3은 개발한 차량 제어 프레임워크를 보여준다. 마스터 컴퓨터를 기준으로 우측에 차량 제어 프레임워크가 도시되어 있다. 차량 제어 프레임워크는 조향 제어 모터, 기어 제어 모터, 가속/감속 페달 제어 모터들, 이상 4개의 모터들로 구성되며 마스터 컴퓨터를 통해서 해당 모터 드라이브에게 위치 명령 (p1, p2, p3, p4)를 전송하여 제어할 수 있다. 모터 드라이브들은 마스터 PC와 이더캣(EtherCAT)[12]이라는 실시간 필드 버스로 연결된다.

3.1 순항 제어

차량이 주어진 속도 v를 따라 정속 주행하도록 차량을 제어하는 것을 순항 제어(Cruise Control)라 한다. 저자들은 차량 제어 프레임워크가 순항 제어를 수행할 수 있도록 PID 제어 알고리즘으로 가속/감속 페달을 제어하는 차량 제어 프로그램(그림 3의 Vehicle Controller)을 개발하였다.

순항 제어를 위한 PID 제어 알고리즘은 타겟 속도 v가 달성되는지를 확인하기 위해서 실제 속도 측정값을 필요로 한다. 저자들은 OBD II (On-Board Diagnostics II)[13] 포트를 이용하여 평균 260 ms 주기(≃ 4 Hz)로 실제 속도값을 얻었고, 이것을 PID 제어 알고리즘에 타겟 속도 v와 함께 입력으로 제공한다.

저자들이 개발한 PID 기반 순항 제어 알고리즘은 각 페달을 잡아당기는 서보 모터를 이용하여 가속(throttle) 페달의 위치 Tpos과 감속(brake) 페달의 위치 Bpos를 제어한다. 위치 변수 Tpos(또는 Bpos)는 최소값 Tmin(또는 Bmin)과 최대값 Tmax(또는 Bmax) 사이의 어떤 값도 가질 수 있다. 최소값은 해당 페달에 아무런 압력이 가해지지 않은 상태를 의미한다.

순항 제어 알고리즘은 Autoware로부터 차량이 달성해야 할 타겟 속도 v를 입력받는데, 설명의 편의를 위해서 v를 기준 속도 V_{ref}로 표기하기로 하자. 그러면, 이 값은 목표 속도 생성기(Target Velocity Generator)에 입력되어 PID 제어 알고리즘이 당장 달성해야 할 단기적인 목표 속도 V_{target}로 변환된다. 이것은 실제 속도와 큰 차이가 있는 기준 속도가 갑자기 주어지는 경우에 단계적으로 기준 속도를 따라잡는 단기적인 목표 속도

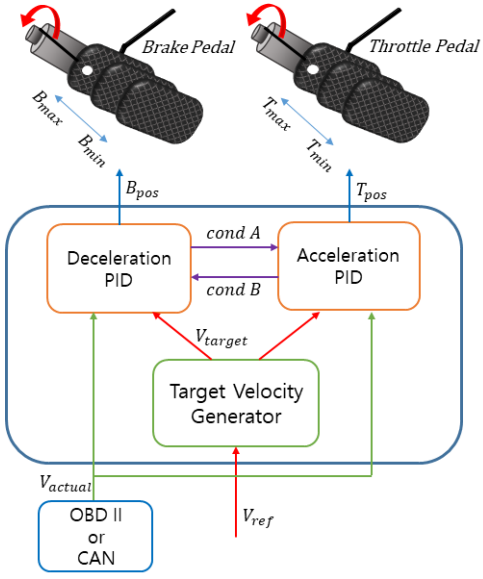


그림 4 PID 제어 알고리즘

들을 생성하여 기준 속도에 빨리 수렴하기 위한 것이다. 사실 Autoware는 지역적 경로 계획 알고리즘이 실제 속도를 고려하여 점진적으로 V_{ref}를 변화시키기 때문에 목표 속도 생성기가 활성화될 가능성이 거의 없다.

순항 제어 알고리즘은 가속 모드에서 실행되는 가속 PID 알고리즘과 감속 모드에서 실행되는 감속 PID 알고리즘, 두 가지로 구성된다. 가속 모드는 $V_{actual} > V_{target} + \Delta$인 조건이 감지되면 감속 모드로 전환되고, 감속 모드는 $V_{actual} < V_{target} - \Delta$인 조건이 감지되면 가속 모드로 전환된다.

PID 알고리즘은 목표 속도와 실제 속도의 차이 e = V_{target} - V_{actual}를 기반으로 페달 위치(Tpos or Bpos)를 e의 크기에 비례 상수 K_P를 곱한 값, e의 누적값에 비례 상수 K_I를 곱한 값, e의 변화율에 비례 상수 K_D를 곱한 값을 합산하여 결정한다. 이 때 사용되는 비례 상수들 K_P, K_I, K_D는 목표 속도에 수렴하는 시간을 최대한 단축하도록 튜닝한다.

저자들은 개발한 PID 제어 알고리즘을 OBD II 포트를 사용하는 경우와 CAN 게이트웨이를 사용하는 경우를 비교하여 평가할 기회가 있었다. OBD를 사용하는 경우는 K_P=600, K_I=25, K_D=300으로 설정하였고, CAN을 사용하는 경우는 K_P=2500, K_I=7, K_D=300으로 설정하였다.

그림 5와 6은 타겟 차량을 평지에서 정지 상태로 출발시켜 V_{ref}=50 또는 70 km/h까지 도달하는 과정에서 V_{actual}의 변화를 보여준다. 그림 5에서 OBD의 경우 수렴 시간은 30초를 넘는데 반해, CAN의 경우 수렴 시간은 6초에 불과하므로 CAN을 이용하는 경우의 수렴 속도가 5배나 빠름을 확인할 수 있다. 이는 CAN의 경우 실제 속도 측정값이 실수형으로 주어져서 OBD의 정수형보다 정밀하고, 업데이트 주기가 10 ms로, OBD의 250 ms보다 25배나 짧기 때문이다. 그림 6에서도 비슷한 경향을 확인할 수 있다.

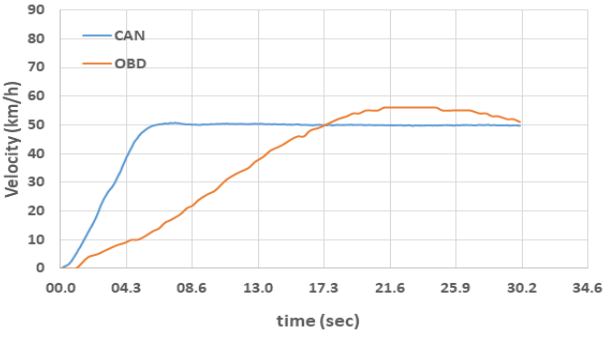


그림 5 V_{ref} = 50 km/h

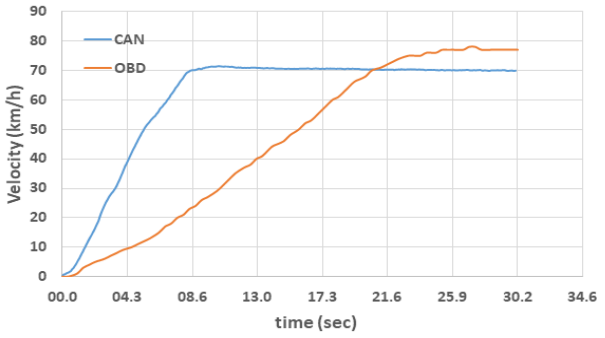


그림 6 V_{ref} = 70 km/h

OBD 포트를 사용하면 Autoware가 제공하는 타겟 속도 v에 대한 수렴 속도가 CAN 게이트웨이를 사용하는 경우보다 확실히 느리다. 그러나, 이것은 차량의 실제 속도값을 얻는 방법적인 차이로서, 저자들이 제안한 차량 제어 프레임워크 상의 문제는 아니다. CAN 게이트웨이가 가용하지 않은 경우에 OBD 포트보다 빠른 수렴 속도를 얻기 위해서는, 자차 위치 추정 모듈 또는 IMU 센서를 이용하는 방안을 고려할 필요가 있다.

3.2 조향 제어

조향 제어는 Autoware가 제공하는 타겟 조향 각속도 ω를 타겟 조향각 δ로 변환하고 이를 조향 모터의 위치 명령으로 변환하는 것이다. 이를 위해 Ackermann 조향 모델[14]을 차량 제어 프로그램 안에 구현하였다. Ackermann 조향 모델은 사륜 자동차를 이륜 자전거로 단순화하여 타겟 조향각 δ를 계산한다. 타겟 조향각 δ는 radian 단위로 계산되는데, 이것은 조향 제어 모터 관점에서 qc(quad counts) 단위의 위치 명령으로 바로 변환 가능하다.

Autoware는 순수 추종 알고리즘에 의해서 조향 각속도 ω를 계산한다. 계산 과정을 이해하기 위해서 그림 7을 참조하자. 이 그림은 차량 뒷바퀴 사이의 중점을 원점으로 삼고, 뒷바퀴 축은 x축, 이에 수직한 축을 y축으로 가정한다. 이 때 (x, y)는 순수 추종 알고리즘이 도달해야 할 경로점이라고 정의하자. 그러면, 순수 추종 알고리즘은 차량을 좌표 (0, 0)에서 (x, y)로 이동시키기 위해서 어떤 부채꼴의 호 L를 따라야 하는지를 계산한다.

이 부채꼴의 호 L은 반지름 r과 원 중점 (r, 0)에 의해서 결정된다. 즉, (r, 0)으로부터 원점까지의 거리와 (x, y)까지의 거리는 모두 r이다. 이 때 부채꼴의 호 L의 곡률은 1/r이라고 말한다.

반지름 r과 부채꼴 호 L이 정해지고 그것들로 정의

되는 부채꼴의 각 θ 사이의 관계는 L = r × θ로 정의된다. 여기서 양변을 미분하면 v = r × ω 라는 공식을 얻는다.

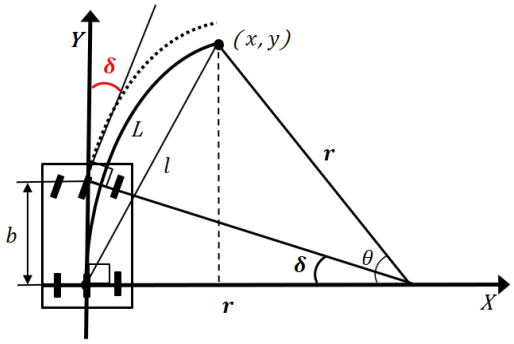


그림 7 순수 추종 알고리즘과 Ackermann 모델의 관계

v는 부채꼴의 호를 따라 이동하는 차량의 속도인데, 이것은 경로 계획 알고리즘이 사용자의 입력을 받아 결정하는 값이다. 따라서, v와 r이 주어지면 타겟 각속도 ω가 자연스럽게 정해진다. 바꾸어 말하면, Autoware는 순수 추종 알고리즘을 통해 차량 제어 프로그램에게 타겟 속도 v와 곡률 계산에 필요한 타겟 각속도 ω를 함께 제공한다. 이로부터 차량 제어 프로그램은 실제 조향 휠의 타겟 조향각 δ를 다음과 같이 얻을 수 있다. 여기서 b는 앞바퀴 축과 뒷바퀴 축 사이의 거리를 의미한다.

$$\delta = \tan^{-1}\left(\frac{b}{r}\right) = \tan^{-1}\left(\frac{b \times \omega}{v}\right)$$

결과적으로, 차량 제어 프로그램은 (v, ω)를 Ackerman 모델을 통해서 (v, δ)로 변환하여 v를 순항 제어기(cruise controller)의 입력으로, δ를 조향 제어기(steering controller)의 입력으로 사용한다.

4. 벡터 맵 제작

Autoware는 CSV 형식의 벡터 맵를 사용하며, 그 안에 주행 유도선(dtlane), 차선(line), 횡단보도(cross), 정지선(stopline) 등의 정보를 포함한다. 이 중에서 주행 유도선은 Autoware의 기준 경로와 주행 궤적 생성에 필수적이다. Autoware는 자율주행 차량이 도달할 수 있는 모든 지점들을 경로점으로 이해하고, 모든 차로의 중앙을 지나가는 경로들을 경로점의 나열들로 표현한다. 이 때 주행 유도선으로 사용될 경로점들은 다음과 같은 요구사항들을 만족해야 한다.

첫째, 모든 경로점들은 등간격(약 1 미터)으로 배치되어야 한다. 둘째, 차량이 교차로를 건너가거나, 차로 변경을 수행하기 위해서는 하나의 경로점이 2개 이상의 다른 경로점들로 분기하도록 미리 연결 링크가 만들어져 있어야 한다. 셋째, 곡선로 상에 위치하는 경로점들은 차량의 조향휠이 급격하게 회전하는 일이 없도록 완만하게 변화하는 곡률을 가져야 한다. 이러한 요구사항들은 Autoware의 전역적 경로 계획 알고리즘이 목적지까지의 최단 경로를 탐색하기 위해서, 그리고 주행 궤적을 생성하는 지역적 경로 계획 알고리즘이 곡률 변화가 작은 궤적들을 생성하기 위해서 필요하다.

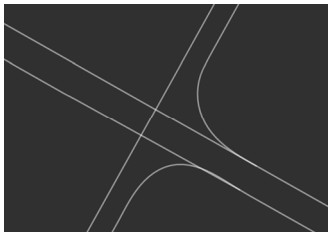
결과적으로 Autoware가 요구하는 주행 유도선들은 상기 요구사항들을 모두 만족하는 경로점들의 시퀀스이어야 한다. 그러나, 단순히 지리정보의 제공을 목적으로 만들어지는 벡터 맵들은 주행 유도선을 제공하지 않거나, 제공한다 해도 상기 요구사항들을 모두 지키지는 않는다. 예를 들어, 국토지리정보원(NGII)[15]가 제공하는 SHP 형식의 벡터 맵은 주행 유도선들을 제공하기는 하지만, 경로점들의 시퀀스 형태가 아니며, 교차로 구간에 대해서는 주행 유도선 자체를 제공하지 않는다. 이는 국토지리정보원이 특정한 자율주행 SW 플랫폼에 의존적인 방식으로 벡터 맵을 제작하지 않기 때문이다.

표 1 NGII 지도 형식과 Autoware 지도 형식의 비교

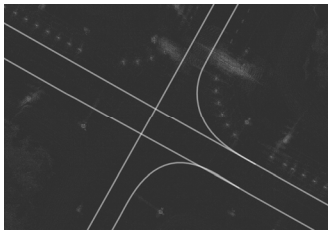
	NGII	Autoware
정보 형태	연속(shp)	이산(csv)
좌표계	GPS 좌표계[16]	지역 좌표계
방향성, 곡률	없음	있음
교차로	끊어짐	연결됨

저자들은 NGII가 제공하는 SHP 형식의 벡터 맵을 Autoware가 사용할 수 있는 CSV 형식의 벡터 맵으로 변환하는 도구를 개발하였다. 개발한 도구는 다음과 같은 입력 파일들을 요구한다: (1) 주행 유도선들의 집합인 LINK 파일(SHP 형식), (2) GPS 좌표를 Autoware가 사용하는 포인트 맵의 좌표로 변환하는 선형 변환 행렬, (3) 실제 차량이 타겟 지역의 차로들을 주행하면서 얻은, 포인트 맵 상의 좌표들의 시퀀스 파일들이다.

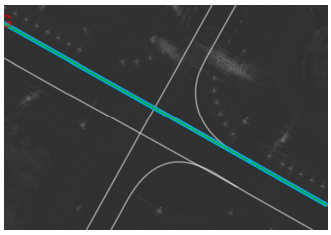
입력 파일들 중에서 (2)와 (3)이 추가로 필요한 이유는 Autoware가 자율주행을 수행하기 위해서는 벡터 맵과 별도로 자차 위치 추정을 위해 포인트 맵이 필요한데, 이 포인트 맵에 다음 두 가지 이슈가 있기 때문이다. 첫째, GPS 좌표를 기준점으로 삼지 않고 제작된



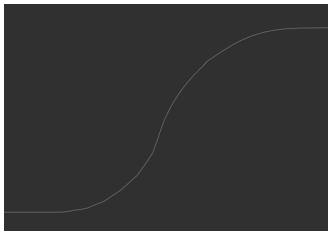
(a) 교차로 구간 벡터 맵



(b) 포인트 맵과 동기화한 교차로 구간 벡터 맵

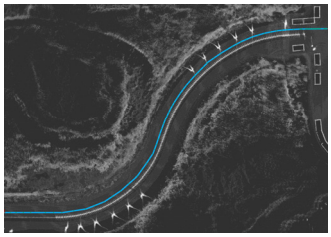


(c) 교차로 구간의 경로 계획 결과 화면



(d) 곡선로 구간 벡터 맵

(e) 포인트 맵과 동기화한 곡선로 구간 벡터 맵



(f) 곡선로 구간의 경로 계획 결과 화면

그림 8 제안하는 지도 변환 도구의 결과물 예들

포인트 맵은 GPS 좌표계와 다른 로컬 좌표계를 갖는다. 따라서 GPS 좌표를 사용하는 벡터 맵 상의 좌표들을 로컬 좌표계의 포인트 맵 상에 매핑하기 위해서는 좌표 변환 행렬이 필요하다. 둘째, 포인트 맵은 그 포인트 맵이 대상으로 하는 실제 물리 공간의 기하학적 비율을 정확하게 반영한다고 보장하지 않는다. 예를 들어, 포인트 맵의 어떤 구간은 물리 공간의 1 평방미터를 같은 크기의 정사각형으로 모델링할 수도 있지만, 다른 구간은 조금 찌그러진 사각형으로 모델링할 수도 있다. 그 이유는 3차원 포인트 맵을 생성하는 NDT 매핑 알고리즘이 전역적인 기준점들 없이 단순히 라이더가 제공한 포인트 집합을 점진적으로 단순 누적시키는 방식으로 3차원 물리 공간을 모델링하기 때문이다. 그 결과, 구간별로 기하학적 비율이 다른 3차원 포인트 맵이 만들어지는데, 이러한 포인트 맵을 대상으로 GPS 좌표를 상기 변환 행렬을 이용하여 단순 변환하면 포인트 맵 안에서 그 좌표가 올바른 지점에 매핑될 수가 없게 된다. 결과적으로 SHP 안에 GPS 좌표로 표현된 주행 유도선을 상기 변환 행렬을 이용해 좌표 변환하는 것만으로는 Autoware가 사용하는 포인트 맵과 동기화할 수가 없는 것이다. 따라서, 저자들이 개발한 지도 변환 도구는 CSV 형식으로 주행 유도선을 변환하는 과정에서 Autoware가 사용할 포인트 맵과 동기화할 목적으로, 실제 차량이 타겟 지역의 차로들을 주행하면서 얻은, 차량 이동 좌표들의 시퀀스 파일을 추가로 사용한다. 그 결과, 제안하는 도구는 지도 변환의 결과물로서 point.csv, node.csv, dtlane.csv, lane.csv 이상 4가지 파일들을 출력한다.

개발한 변환 도구는 4단계로 동작한다. 단계 1은 경로점들을 약 1미터의 등간격으로 조정하는 단계이다. 이 단계는 곡선로 상의 경로점들이 간격이 짧을 때 방향 벡터의 순간 변화율이 커짐으로 인해 발생하는 조향휠의 급회전을 방지한다. Autoware는 또한 최단 경로를 탐색할 때 경로점들의 개수로 경로의 길이를 판단하기 때문에 등간격 조정은 필수적이다. 단계 2는 직선로 상의 경로점들의 방향 벡터를 평탄화하는 단계이다. 이 또한 조향휠의 급회전을 방지하기 위함이다. 단계 3은 교차로에서 미연결 상태인 주행 유도선들을 상호 연결하는 단계이다. 이 때 교차로 상에서 좌회전 또는 우회전을 통해서 서로 연결되는 주행 유도선들에 대해서는 곡선 유도선을 끼워넣어서 연결한다. 이를 위해서 개발한 도구는 라인 샘플링(line sampling) 기법[17]를 통해서 완만한 곡률을 갖는 곡선 유도선을 생성한다. 단계 4는 벡터 맵의 주행 유도선을 포인트 맵 상의 동일한 경로에 대해 실제로 차량이 주행한 궤적에 잘 매치되도록 주행 유도선을 수작업으로 편집하는 단계이다. 이렇게 포인트 맵 안에서 측정된 궤적들을 기준 삼아 동기화된 주행 유도선들은 최종적으로 CSV 형식의 벡터 맵으로 출력된다.

그림 8은 제안하는 지도 변환 도구의 결과물 예시들을 보여준다. 그림 8a은 자율주행 실험도시 K-City의 한 교차로 구간에 대해서 단계 1부터 4까지를 적

용한 벡터 맵을 보여준다. 그림 8b는 이 벡터 맵을 포인트 맵과 동기화한 것이고, 그림 8c는 최종적으로 Autoware 안에서 주행 경로를 계산한 결과 화면이다. 그림 8의 하단 그림들은 단계 1부터 4까지를 적용한 곡선로 구간 벡터 맵(그림8d)에 대해서 포인트 맵과 동기화하고(그림8e) 최종적으로 Autoware 안에서 주행 경로(파란 선으로 표시됨)를 계산한 결과 화면(그림8f)를 보여준다. 최종적으로 얻어진 벡터 맵은 Autoware가 이식된 i30 자율주행 차량에 적용하여 실도로 주행이 가능함을 확인하였다.

5. 결론 및 향후 연구

본 기고문은 Autoware를 실차량에 이식하고 구동할 때 고려해야 할 주요 문제들에 대해서 기술하였다. 요약하면, Autoware는 맵기반 자율주행만을 지원하기 때문에, 고정밀 포인트 맵과 벡터 맵이 필요하고, 타겟 차량을 위한 제어 프레임워크가 필요하다. 다행히도 차량 제어 프레임워크를 제작하는 것은 자율주행을 연구하는 많은 대학 연구실이 이미 축적하고 있는 역량과 스킬로 충분히 가능할 것이다.

그러나, 대학 연구실 차원에서 임의의 도로 구간에 대해서 고정밀 포인트 맵과 벡터 맵을 제작하는 일은 쉽지 않다. 포인트 맵은 NDT 매핑 알고리즘을 통해서 쉽게 얻을 수 있지만, 이것에 동기화된 벡터 맵을 만드는 것은 이 목적에 맞는 맵 편집 도구를 개발해도 많은 수작업을 요구한다는 것을 확인하였다. 아마도 이런 이유로, Autoware 사이트에서는 고정밀 지도를 제작하기 위해서 회사를 고용할 것을 권고한다.

자율주행 연구에 Autoware를 자유롭게 사용하기 위해서는 Autoware가 요구하는 고정밀 지도를 쉽게 제작하는 방법이 개발되어야 한다. 최근에 고정밀 지도 제작을 전문으로 하는 벤처 회사들이 등장하기는 했지만, 저자들의 경험에 의하면 그 회사들이 자율주행 기술을 잘 이해하지 못하기 때문에 Autoware에 꼭맞는 지도를 제작해주는데 현재로서는 아쉬운 점들이 많다.

이에 저자들은 Autoware 플랫폼을 어느 연구 그룹이나 쉽게 사용할 수 있도록 고정밀 지도 제작 서비스를 개발하고 있다. 고정밀 지도 제작 기술은 Autoware 연구와는 또 다른 연구를 요구하지만, 서비스가 가용해지는 대로 Autoware 연구자들이 활용할 수 있는 기회를 만들고자 한다.

참고문헌

[1] Baidu. “Apollo Open Platform,” http://apollo.auto

[2] Autoware Foundation. “Autoware,” http://autoware.org

[3] P. Biber and W. Straber, “The Normal Distributions Transform: A New Approach to Laser Scan Matching,” in Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), vol. 3, 2003, pp. 2743-2748.

[4] J. Redmon et al. “You Only Look Once: Unified, Real-Time Object Detection,” in Proc. of 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), vol. 1, 2016, pp. 779-788.

[5] R. B. Rusu. “Semantic 3D Object Maps for Everyday Manipulation in Human Living Environments,” Institut für Informatik der Technischen Universität München, Jul. 2009.

[6] TIER4. “Autoware Tools: Vector Map Builder,” https://tools.tier4.jp/feature/vector_map_builder

[7] M. Werling, J. Ziegler, S. Kammel, and S. Thrun, “Optimal Trajectory Generation for Dynamic Street Scenarios in a Frenet Frame,” 2010 IEEE International Conference on Robotics and Automation.

[8] R. C. Coulter, “Implementation of the Pure Pursuit Path Tracking Algorithm,” Techincal Report, Carnegie Mellon University Robotics Institute, 1992

[9] “Gazebo,” http://gazebosim.org

[10] “LGSVL Simulator,” https://www.lgsvlsimulator.com

[11] 이영재 등 3명, “자율 주행 소프트웨어를 위한 차량 제어 프레임워크 구현 및 평가”, 정보과학회: 컴퓨팅의 실제 및 레터, 2018

[12] EtherCAT Technology Group. “EtherCAT - the Ethernet fieldbus,” http://www.ethercat.org/pdf/ethercat_e.pdf

[13] “On-board Diagnostics,” https://en.wikipedia.org/wiki/On-board_diagnostics

[14] Jarrod M. Snider. “Automatic Steering Methods for Autonomous Automobile Path Tracking,” Technical Report CMU-RI-TR-09-08, Carnegie Mellon University Robotics Institute, 2009

[15] “국토지리정보원,” https://www.ngii.go.kr

[16] “World Geodetic System 1984 (WGS84),” https://en.wikipedia.org/wiki/World_Geodetic_System

[17] Atsushi Sakai. “State Space Sampling of Feasible Motions for High-Performance Mobile Robot Navigation in Complex Environments,” Journal of Field Robotics, vol. 25, no. 6-7, pp 325-345, 2008

약 력

최 규 진

2019 숭실대학교 스마트시스템소프트웨어학과 졸업(학사)
2019 현대자동자 자율주행 경진대회 AVC 참가 (차량 제작 및 차량 제어 담당)
2019~현재 숭실대학교 지능시스템학과 재학(석사 과정)
관심분야 : 자율주행, 차량 제어
Email : qjinchoi@soongsil.ac.kr

이 효 은

2019 숭실대학교 스마트시스템소프트웨어학과 졸업(학사)
2019 현대자동자 자율주행 경진대회 AVC 참가 (Autoware 담당)
2019~현재 숭실대학교 지능시스템학과 재학(석사 과정)
관심분야 : 자율주행, 실시간 시스템
Email : hyoeunlee@soongsil.ac.kr

백 한 나

2019 숭실대학교 정보통신전자공학부 졸업(학사)
2019 현대자동자 자율주행 경진대회 AVC 참가(고정밀지도 담당)
2019~현재 숭실대학교 지능시스템학과 재학(석사 과정)
관심분야 : 자율주행, 고정밀지도
Email : vlvl5@ssu.ac.kr

구 성 우

2012~현재 숭실대학교 스마트시스템소프트웨어학과 재학(학사과정)
2019 현대자동자 자율주행 경진대회 AVC 참가(고정밀지도 담당)
관심분야 : 자율주행, 고정밀지도
Email : rntjddn@soongsil.ac.kr

김 강 희

1996 서울대학교 컴퓨터공학과 졸업(학사)
1998 서울대학교 전기·컴퓨터공학부 졸업(석사)
2004 서울대학교 전기·컴퓨터공학부 졸업(박사)
2004~2010 삼성전자 무선사업부 책임연구원
2010~현재 숭실대학교 스마트시스템소프트웨어학과 부교수
관심분야 : 실시간 시스템 가상화, 실시간 소프트웨어 분석과 설계, 자율주행 시스템
Email : khkim@ssu.ac.kr