Journal of the Korea Academia-Industrial cooperation Society
Vol. 25, No. 4 pp. 299-308, 2024

https://doi.org/10.5762/KAIS.2024.25.4.299
ISSN 1975-4701 / eISSN 2288-4688

오픈소스 및 AMR을 이용한 GNSS 기반의 자율주행 시스템

강효련, 정성엽*
한국교통대학교 기계공학과

GNSS-based Autonomous Driving System using AMR and Open Source

Hyo Ryeon Kang, Seong Youb Chung*
Division of Mechanical Engineering Korea National University of Transportation

요 약 본 논문에서는 오픈소스와 하드웨어 플랫폼을 이용해 GNSS 기반의 자율주행 시스템을 구현하였다. 자율주행의 개발에는 많은 시간과 비용이 소모되므로 이를 극복하고자 오픈소스 소프트웨어를 이용한 연구가 활발히 진행되고 있다. 논문에서는 Ubuntu 20.04, ROS Noetic의 오픈소스 소프트웨어 플랫폼을 사용하며 AMR(Autonomous Mobile robot)과 GNSS, IMU, LiDAR, 카메라 등의 센서를 하드웨어 플랫폼으로 사용한다. GNSS 데이터의 보정과 위치 및 자세 추정, SLAM(Simultaneous Localization and Mapping), 네비게이션 등의 기능을 가진 오픈소스를 적용하고 시스템을 구현하였다. 그리고 웨이포인트를 작성하여 자율주행 시스템에 경로로 제공한다. 실제 환경에서의 테스트 결과, 구현한 자율주행 시스템이 평지 주행, 언덕 주행에 대한 두 개의 경로에서 성공적으로 동작함을 확인하였다. 또한, 웨이포인트의 스무딩에 따른 주행의 정확성을 평가하기 위해 평지 주행 경로에서의 RMSE(Root Mean Square Error)를 계산하였다. RMSE는 0.5649 m에서 웨이포인트의 스무딩을 거친 후에는 0.2067 m로 감소함을 확인했다. 이를 통해 오픈소스를 이용한 자율주행 시스템의 제작이 가능한 점을 확인하였다.

Abstract In this paper, we implemented a GNSS-based autonomous driving system using open-source software and hardware platforms. Developing autonomous driving systems requires significant time and cost, so research utilizing open-source software has been actively pursued to overcome these challenges. In our study, we utilized the open-source software platforms of Ubuntu 20.04 and ROS Noetic. As for hardware platforms, we employed sensors such as AMR (Autonomous Mobile Robot), GNSS, IMU, LiDAR, and cameras. We applied open-source solutions that encompass functionalities like GNSS data correction, position and attitude estimation, SLAM (Simultaneous Localization and Mapping), and navigation to implement the system. Additionally, we created waypoints to provide paths for the autonomous driving system. The real-world testing results confirmed that our implemented autonomous driving system successfully operated on two different routes: flat terrain and hilly terrain. Furthermore, we evaluated the accuracy of waypoint-based navigation by calculating the RMSE (Root Mean Square Error) on the flat terrain route. After applying waypoint smoothing, the RMSE decreased from 0.5649 meters to 0.2067 meters. This demonstrates the feasibility of creating autonomous driving systems using open-source tools.

Keywords : Autonomous Driving, Localization, SLAM, Navigation, Open Source

본 논문은 2023년도 교육부의 재원으로 한국연구재단의 지원을 받아 수행된 지자체-대학 협력기반 지역혁신사업의 결과임 (2021RIS-001(1345370811)).
*Corresponding Author : Seong-Youb Chung(Korea National University of Transportation)
email: sychung@ut.ac.kr
Received January 24, 2024 Revised March 8, 2024
Accepted April 5, 2024 Published April 30, 2024

1. 서론

자율주행은 다양한 분야에 활용이 가능하며 연구가 활발하게 진행되고 있다. 그중 자율주행 자동차는 주변 환경을 인식, 주행 상황 판단 및 차량의 제어를 실행함으로 스스로 정해진 목적지까지 주행한다. 자율주행 자동차의 개발은 운전자에 편의성을 주며 부주의한 운전으로 인한 사고를 줄이며 효율적인 경로 선택으로 인한 교통 체증 감소, 연비의 향상으로 차량의 효율성을 높일 수 있다[1,2].

하지만, 자율주행 자동차는 안전 문제, 기술적 한계로 인한 완벽한 자율주행이 불가한 문제 그리고 제작 및 유지보수 비용으로 인한 구매 비용이 높다는 문제가 있다[3]. 이로 인해 자율주행의 개발에는 많은 시간과 비용이 소모된다. 이러한 비용을 줄이기 위해 개발자들은 오픈소스를 이용한 연구를 진행하고 있다. 오픈소스 소프트웨어는 코드가 공개되어 누구나 수정, 배포가 가능한 소프트웨어를 의미하고 누구나 접근이 쉽다는 장점에 의해 소프트웨어에 대한 의견, 오류 등을 공유하며 개발 비용, 시간 혹은 예상치 못한 오류들을 발견할 수 있는 장점이 있다.

오픈소스를 사용한 자율주행 시스템 혹은 자율주행의 인지, 판단, 제어에 관련된 연구는 활발하게 진행되고 있다. [4]는 저가형 UGV(Unmanned Ground Vehicle)와 2D LiDAR(Light Detection and Ranging) 하나만을 사용한 자율주행 플랫폼과 오픈소스 SLAM 알고리즘을 기반으로 한 자율주행 시스템을 제시하였으며 원활한 동작이 가능함을 확인했다. [5]는 저가의 카메라, LiDAR, IMU(Inertial Measurement Unit), GNSS(Global Navigation Satelite System) 기반으로 한 자율주행 플랫폼을 구성하고, 오픈소스로 구성된 자율주행 시스템을 구현하였다. 실외 환경에서 주변 장애물에 따른 LiDAR의 감지에 따라 불안정한 모습을 보였으나 IGVC-2019 (Intelligent Ground Vehicle Competition 2019)의 실내자율주행의 AutoNav 코스를 완주하였다. [6]는 카메라, LiDAR, IMU, GNSS 각 1대를 사용해 플랫폼을 구성하며 위치 및 자세 추정, 경로 계획과 회피 기동에 대한 오픈소스 알고리즘을 사용하여 실내, 실외 환경에서의 자율주행 시스템을 구현하였다. [7]는 츠쿠바 챌린지(Tsukuba Challenge)에서 GNSS 없이 카메라와 라이다를 기반으로 한 플랫폼을 토대로 자율주행 오픈소스 플랫폼인 Autoware[8]에서 CNN(Convolutional neural network)을 통해 E2E(End-to-End) 알고리즘을 보완한 실외 자율주행시스템을 구현하였다. 최종 결과로 웨이포인트에 대한 추종 정확성을 평가하기 위한 RMSE(Root Mean Square Error)는 0.1768 m이었다. 또한, [4-7] 모두 오픈소스 메타운영체제인 ROS(Robot Operating System)을 기반으로 하였다.

이처럼 자율주행에 필요한 기능들에 대한 오픈소스의 성능 향상에 관한 연구를 진행해 자율주행에 필요한 기능을 구현하는 연구가 있고, 자율주행에 필요한 전체 기능을 종합해 오픈소스로 제공하는 연구도 활발히 진행되고 있다.

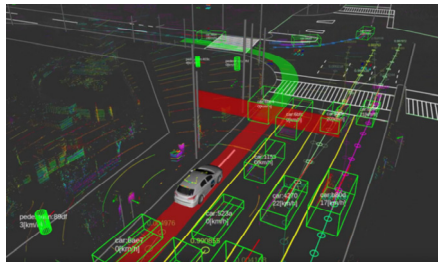


Fig. 1. Autoware[8]

Fig. 2. Hardware platform of F1TENTH[9]

Fig. 1은 앞의 [7]에서 언급되었던 일본의 TIERIV사가 개발한 ROS기반의 자율주행 오픈소스 플랫폼인 Autoware이다[8].

Fig. 2는 자율주행 자동차 경주 대회, 연구를 위한 오픈소스 플랫폼인 F1TENTH의 하드웨어 플랫폼이다. 실제 차량의 1/10 크기의 RC 차량을 이용해 실내 환경에서의 경주 및 자율주행 알고리즘 개발에 사용되며 저렴하고 안전한 실험 환경의 제공과 사용을 위한 튜토리얼도 제공해 사용자들의 손쉬운 접근과 의견 교환을 기대할 수 있다[9].

이처럼 자율주행의 개발을 위한 각 기능을 가진 오픈소스나 하나로 통합된 형태의 오픈소스들이 배포되고 있으며, 오픈소스의 적용이 가능한 하드웨어 플랫폼들도 판매되고 있어 사용자들은 이를 이용해 자유로운 자율주행 시스템의 구현이 가능하다.

본 논문에서는 자율주행 시스템을 개발하는데 자유로운 사용이 가능하고 여러 사람의 의견을 구할 수 있으며 범용성이 높은 오픈소스를 이용하여 기본 알고리즘을 작성하고 특정 알고리즘의 추가 혹은 변경이 가능한 자율주행 시스템을 구현한다. 그 후 테스트를 통해 자율주행 알고리즘의 취약점을 파악 후 향후 연구에 해당 내용을 추가하여 알고리즘을 보완하고자 한다. 자율주행 시스템을 구성하는데 필요한 인지, 판단, 제어 등의 기능을 가진 오픈소스를 사용하며 구현한 자율주행 시스템을 시뮬레이션과 실제 환경에 적용하여 시스템을 수정 및 보완하였다.

2. 본론

자율주행 시스템의 필요한 기능은 크게 인지, 판단, 제어 세 가지로 볼 수 있다. 이 외에도 통신, 보안 등이 있으나 앞의 세 가지가 자율주행에 있어 큰 축을 담당하기에 본 논문에서는 플랫폼의 구성, 자율주행 알고리즘 개발과 최종 알고리즘으로 나누어 다루고자 한다.

2.1 플랫폼 구성

기본적인 소프트웨어 플랫폼은 접근성과 범용성이 좋으며 커뮤니티화가 활발한 Ubuntu와 ROS를 사용하며. Ubuntu는 Linux 기반의 오픈소스 운영체제이며 다양한 프로그래밍 언어, 개발 도구 등의 사용이 가능해 개발자들이 사용한다. ROS는 로봇 응용프로그램을 개발할 때 필요한 기능 구현 및 관리와 개발 환경에 필요한 라이브러리 등을 제공하는 오픈소스 메타운영체제이며 시뮬레이터를 통한 시스템의 테스트와 디버깅 또한 가능하다.

하드웨어 플랫폼은 AMR(Autonomous Mobile Robot), GNSS, IMU, LiDAR, 카메라 각 1대씩을 사용하여 구성한다. Fig. 3은 제작한 하드웨어 플랫폼의 사진이며 사용된 플랫폼의 모델은 Table 1과 같다.

Table 1. Specification of Hardware, Software Platform

Spec.		Company	Model
Soft.	OS	Canonical Ltd.	Ubuntu 20.04
	Meta OS	Open Robotics	ROS Noetic
Hard.	AMR	AgileX Robotics	Hunter-SE
	GNSS	Ublox	C099-F9P-0
	IMU	VectorNav	VN-100
	LIDAR	Velodyne	VLP-16
	Camera	Logitech	C920

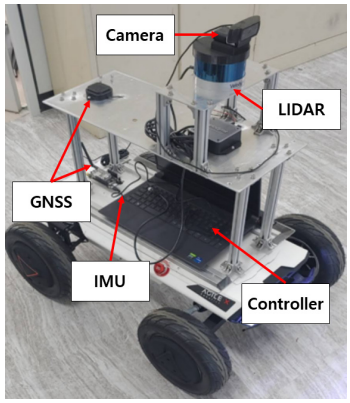


Fig. 3. Hardware platform for experimentation

2.2 자율주행 알고리즘 개발

2.2.1 인지

자율주행에 있어 인지는 센서들을 통해 차량의 위치와 주변 환경에 대해 파악하며 이를 토대로 주행 경로의 계획, 장애물 회피 및 주행의 안전성을 유지하는 역할을 한다. 센서들의 데이터는 주변 환경이나 센서의 성능 및 기타 요인으로 인한 노이즈(noise)가 발생할 수 있다. 이러한 노이즈는 자율주행의 성능을 감소시키는 원인이 되기에 이를 완화하고자 GNSS의 데이터 보정과 GNSS, IMU 및 AMR의 엔코더(encoder)의 센서 융합을 진행했다.

GNSS 데이터의 보정을 위해 2007년 Tomoji Takasu가 소개한 실시간으로 정확한 위치 정보를 얻기 위한 오픈소스 기반의 GNSS 네비게이션 및 위치 결정 라이브러리인 RTKLIB을 사용한다[10]. RTKLIB은 다중 위성 GNSS 데이터의 처리와 RTK(Real-Time Kinematic), DGPS(Differential GPS) 등의 다양한 측위 방법을 지원해 연구 및 교육 목적으로 활용되고 있다[11]. RTKLIB의 RTK 기능 중 하나의 GNSS를 사용하며 인터넷을 통해 물리적, 환경적 요인에 의한 노이즈의 실시간 보정이 가능한 NTRIP-RTK(Networked Transport of RTCM via Internet Protocol-RTK)를 사용한다.

NTRIP-RTK의 사용에는 이동하는 GNSS의 데이터를 보정할 기준국이 필요한데 이는 GNSS 데이터 통합센터를 통해 충주기준국을 이용하였다[12].

Fig. 4은 출발지와 도착지를 정한 후 약 2 km 이동 후 보정에 따라 생기는 오차를 비교한 것이다. 보정을 하지 않은 경우의 오차는 4.54 m이었으며 보정을 거친 경우 오차는 0.14 m이었다.

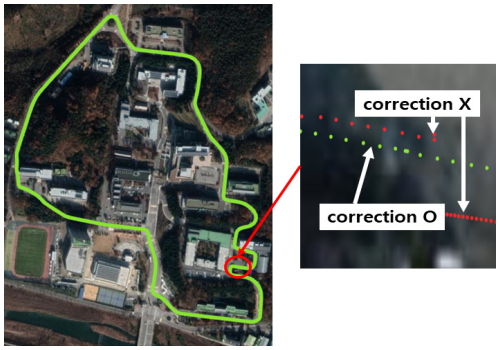


Fig. 4. Error comparison with and without GNSS correction

보정한 GNSS 데이터, IMU, 엔코더 데이터의 센서 융합을 위해 robot_localization을 사용한다. robot_localization은 로봇의 위치 및 자세 추정 작업을 지원하는 패키지(package)이며 확장 칼만 필터(EKF: Extended Kalman Filter, 이하 EKF), 무향 칼만 필터(UKF: Unscented Kalman Filter, 이하 UKF)를 사용하는 위치 추정 노드를 제공한다[13].

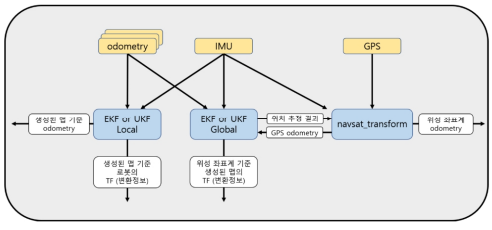


Fig. 5. Schematic of robot_localization using GNSS data

Fig. 5은 robot_localization의 GNSS 데이터의 융합을 진행하는 navsat_transform 노드를 사용한 경우의 개략도이다. 위치 추정 노드는 오도메트리(odometry) 정보와 IMU 데이터를 수신하고 navsat_transform 노드는 IMU와 GPS 데이터를 수신한다. 이를 통해 robot_localization은 로컬(local) 위치에서의 차량의 오도메트리, TF(TF: TransForm, 이하 TF)와 글로벌(global) 위치에서의 차량의 오도메트리, TF 데이터를 생성한다.

해당 과정에서 IMU 데이터, AMR의 엔코더 데이터, 보정된 GNSS 데이터를 EKF를 통해 각 센서의 오차를 보정한다. 이는 로컬과 글로벌 위치에서의 오도메트리, TF로 계산된다.

navsat_transform 노드는 GNSS의 데이터를 통해 로봇의 오도메트리 정보를 지구 좌표계에 맞게 보정하는 기능 또한 포함하고 있다. 하지만, 일반적으로 GNSS에서 사용하는 좌표계인 WGS84(World Geodetic System 1984)는 지구를 타원체로 정의하므로 2D 맵에 있어 정확성이 낮아질 수 있다. 이는 내부에 포함된 기능인 UTM(Universal Transverse Mercator) 좌표계 변환을 통해 해결할 수 있다.

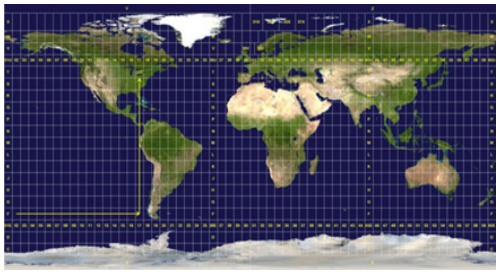


Fig. 6. UTM coordinates system[14]

UTM 좌표계는 Fig. 6과 같이 전 세계를 직사각형으로 나누어 각각에 특정 투영법을 적용한 좌표체계이다. UTM 좌표계는 지구를 평면으로 투영하여 계산하기 때문에 지구 곡률에 따른 왜곡을 줄일 수 있어 WGS84의 정확성 문제를 해결할 수 있다. UTM 좌표계는 각각의 구역(zone)을 정해 좌표를 특정하는데 대한민국은 51S, 51T, 52S, 52T에 해당한다[14].

SLAM은 로봇이 자신의 위치를 파악함과 동시에 주변 환경의 지도를 작성하는 기술이다. GNSS, IMU, LiDAR 등의 센서들과 알고리즘을 사용해 로봇의 위치와 주변 환경의 구조를 추정 및 업데이트하며 주변 환경의 정보를 맵 형태로 저장한다. SLAM은 차량의 위치 결정, 정적 및 동적 장애물 구분, 로봇 경로 계획에 필수적인 기술이며 크게 3D SLAM과 2D SLAM으로 나뉜다.

3D SLAM은 2D SLAM보다 더 많은 데이터를 처리하기에 계산 비용이 높고, 3D 정보를 포함한 3D 맵은 용량이 크지만 복잡한 환경의 공간 구조를 정확하게 파악하여 위치를 정확히 파악할 수 있고, 물체의 입체적인 정보를 토대로 장애물의 뚜렷한 인식이 가능하다. 2D SLAM은 계산 비용이 비교적 낮아 빠른 수행이 가능하고 맵의 용량이 작지만, 2D 데이터의 한계로 인해 비슷한 환경에서 위치를 혼동할 수 있고 2D 맵은 높이 정보를 포함하지 않아 수직 방향 장애물의 감지가 어려울 수 있다.

본 논문에서는 3D SLAM을 사용해 정확한 위치 추정과 함께 네비게이션에서의 계산속도를 높이기 위해 생성된 3D 맵을 투영시킨 2D 맵을 사용한다.

LIO-SAM(LiDAR Odometry and Mapping with Switchable Constraints)은 3D SLAM 오픈소스이며 라이다의 포인트 클라우드(Point Cloud) 데이터와 IMU의 데이터를 활용해 로봇의 위치 추정과 주변 환경을 실시간으로 확인한다[15].

또한, LIO-SAM은 GNSS 데이터를 이용한 보정이 가능하며 LOAM, LIOM 등의 알고리즘에 대해 장거리 주행에서의 정확성을 비교하였을 때 준수한 결과를 보이는 것을 확인하였다[16].

OctoMap은 옥트리(octree) 기반의 라이브러리이며 옥트리는 3D 공간을 격자 형태로 분할해 표현한 자료 구조이다. OctoMap은 3D 맵을 격자 형태로 분할, 각 격자 셀(cell)에 대해 확률 기반의 활성화 상태를 유지하며 이를 이용해 로봇 주변 환경을 모델링하며 장애물과 빈 공간을 구분한 표현이 가능하다[17].

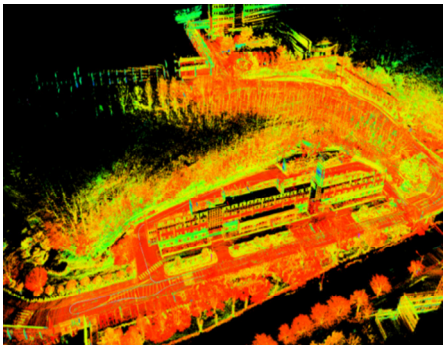
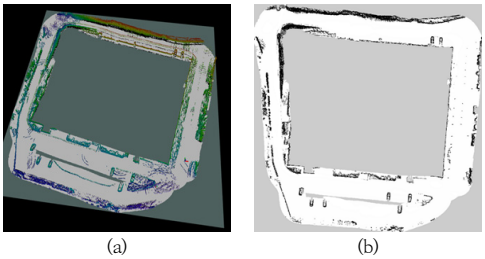


Fig. 7. 3D Map created with LIO-SAM



(a) (b)

Fig. 8. 2D Map created with OctoMap
(a) 3D map by OctoMap
(b) 2D map with noise removed

Fig. 7은 교내의 일부 구역을 LIO-SAM을 통해 3D 맵으로 형성한 것이며 Fig. 8은 LIO-SAM을 통해 생성한 3D 맵을 Octomap을 이용하여 2D 맵으로 투영한 후 노이즈를 제거한 것이다.

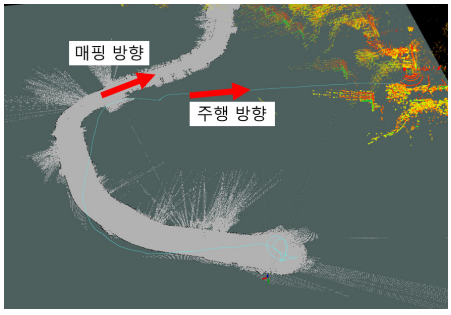


Fig. 9. Error of LIO-SAM when GNSS correction

하지만, LIO-SAM의 고정된 맵 프레임을 GNSS 데이터로 보정하는 경우 Fig. 9와 같이 SLAM 도중 보정된 위치로 매핑이 다시 진행되어 맵이 뒤틀리는 경우가 발생한다. 이는 OctoMap에서 제공하는 reset 기능을 이용하는 것으로 해결할 수 있는데, SLAM을 진행하기 전에 GNSS에 의한 보정을 선행함으로써 해당 오류에 대처할 수 있었다.

2.2.2 판단 및 제어

자율주행에 있어 판단은 인지를 통해 얻은 정보를 토대로 차량의 주행 경로를 계획하고, 주행 중발생하는 상황에 따라 결정을 내리는 역할을 수행한다, 이는 주행의 안정성과 효율성을 유지한다

본 논문에서는 자율주행의 판단 기능의 수행을 위해 ROS에서 기본적으로 제공되는 Navigation Stack[18]과 teb_local_planner[19]를 이용한다.

Navigation Stack은 move_base 기반의 ROS에서 제공되는 오픈소스 패키지이다. 센서로부터 받는 데이터와 이동하고자 하는 지역의 맵, 로봇의 오도메트리 정보를 토대로 목적지까지의 경로 계획과 충돌 회피를 계산하여 로봇을 목적지까지 안전하게 도착하도록 제어한다.

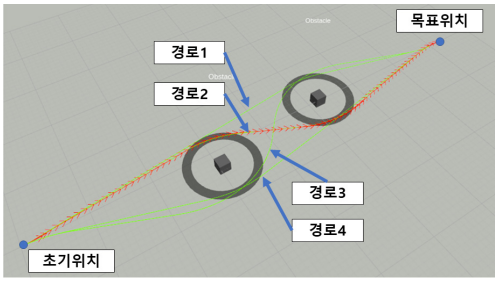


Fig. 10. Example of teb_local_planner

teb_local_planner는 TEB(Timed Elastic Band)의 알고리즘을 통해 지역 경로 계획을 하는 패키지로 복잡한 환경에서 가속도나 속도 제한 같은 로봇의 동적 제약과 충돌 회피를 기대할 수 있다[19]. Fig. 10은 teb_local_planner의 예제를 실행한 것이며 초기 위치에서 목표 위치까지의 로봇의 가상 경로를 계산한다. 이때 장애물과의 거리, 로봇의 동적 특성을 고려하여 계산한 경로 중 가장 적절한 경로를 선택한다. 또한, teb_local_planner는 다른 오픈소스들에 비해 차량과 같은 아커만 조향(Ackermann Steering) 형식의 제어 코드를 제공한다.

teb_local_planner를 Navigation stack에 적용하여 사용하는 경우 내부 파라미터에 AMR의 크기, 한계 조향각, 장애물 인식 범위, 안전거리 등을 설정해야 한다. 이 외에도 사용하는 AMR, GNSS 등의 프레임(frame) ID와 topic 명에 대해 일치시켜줘야 하며 로봇의 오도메트리 topic 명은 LIO-SAM의 오도메트리인 “/lio_sam/mapping/odometry”로 설정해야 한다. 또한, Navigation Stack은 기본적으로 AMCL(Adaptive Monte Carlo Localization)이 기본적으로 탑재되는데 이는 LIO-SAM과 충돌을 일으켜 정확성을 낮추므로 알고리즘에서 실행되지 않게 제외해야 한다.

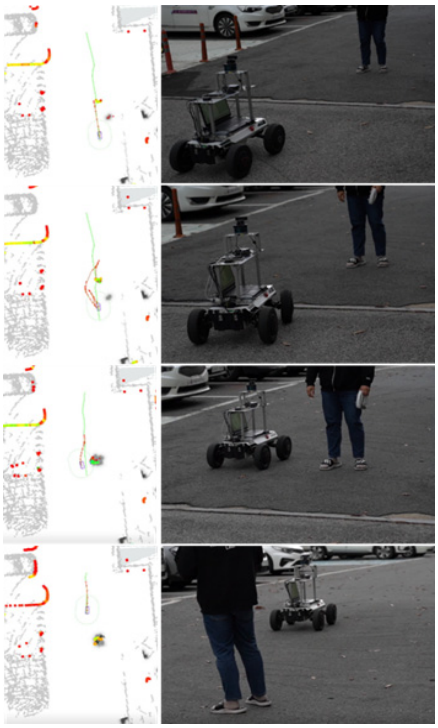


Fig. 11. Collision avoidance test

Fig. 11은 장애물을 인지한 후 Navigation Stack과 teb_local_planner에 의한 회피 기동을 실제 환경에서 정상작동하는지 테스트한 것이며 정상적으로 장애물을 회피하는 것을 확인할 수 있다.

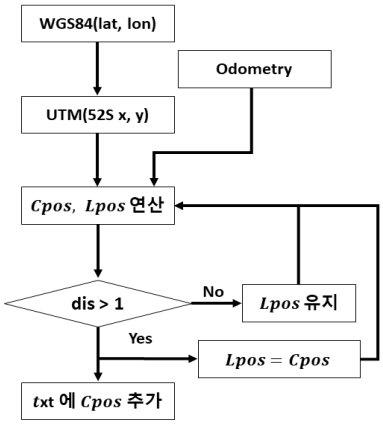


Fig. 12. Waypoint algorithm

Navigation Stack에 연속적인 목표 위치인 웨이포인트(waypoint)를 주기 위해 관련 오픈소스를 사용하려 하였으나 해당 오픈소스는 ROS Noetic에서는 호환되지 않아 직접 알고리즘과 코드를 작성하였다. 알고리즘은 Fig. 12와 같다. $Lpos$는 GNSS 데이터의 마지막 위치를 뜻하며 $Cpos$는 현재 위치를 뜻한다. 웨이포인트는 GNSS의 WGS84의 좌표를 UTM 좌표로 변경한 위치 데이터와 LIO-SAM의 오도메트리에서 차량의 헤딩을 추출하여 1 m의 간격마다 갱신하며 이를 텍스트로 저장한다.

작성한 웨이포인트의 노이즈를 줄이기 위해 스무딩(smoothing)을 통해 보정을 진행하였으며 스무딩 방식은 이동평균(Moving Average Method)이다. 본 논문에서 사용한 이동평균의 수식은 Eq. (1)과 같다. w는 웨이포인트를 의미하며 m은 이동평균을 구할 웨이포인트의 개수를 의미하며 m의 값에 따라 현재 웨이포인트인 w_i의 이전, 이후 웨이포인트를 불러와 계산한다. 이를 통해 생성된 웨이포인트는 Fig. 13과 같으며 스무딩을 진행하는 경우 웨이포인트가 부드럽게 이어지는 것을 확인할 수 있다.

$$w_i = \frac{w_{i-m} + \ldots + w_{i+m}}{2m} = \frac{1}{2m}\sum_{i+m}^{i-m} w_i \qquad (1)$$

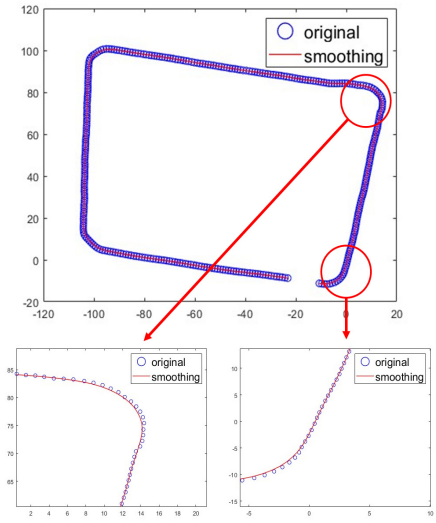


Fig. 13. Comparison waypoint with and without smoothing

2.2.3 최종 알고리즘

본 논문의 GNSS 기반 자율주행 알고리즘은 위치 추정과 맵 제작, 네비게이션과 속도 조절, 카메라를 통한 차선 인식 및 객체 탐지 세 가지 부분으로 나뉘며 이는 Fig. 14와 같다.

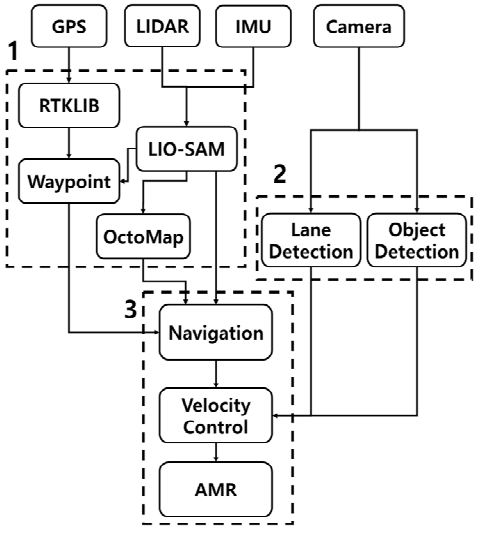
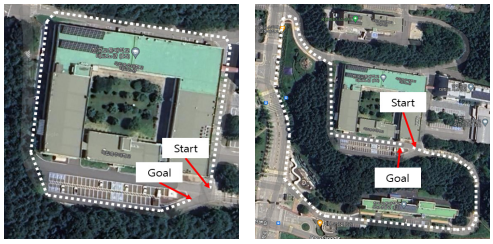


Fig. 14. Algorithm of autonomous driving based on GNSS

RTKLIB를 통한 GNSS 데이터 보정, LIO-SAM으로 3D SLAM 수행, OctoMap의 2D 맵 생성과 웨이포인트는 보정 GNSS와 오도메트리를 기반으로 생성하여 환경을 인지한다. 네비게이션은 웨이포인트, OctoMap의 2D 맵, LIO-SAM의 오도메트리 정보를 기반으로 판단을 진행하여 차량의 속도인 cmd_vel을 생성하고 이를 통해 AMR을 제어한다.

3. 결과 및 고찰

자율주행의 최종 알고리즘을 두 개의 경로에 대해 실제 환경에서 테스트를 진행하였다.



(a) (b)

Fig. 15. Paths for the final algorithm test
(a) Path 1, Mainly flat land
(b) Path 2, Includes hill driving

Fig. 15에서 경로 1은 평지 주행, 약 400 m의 경로이고, 경로 2는 언덕 주행이 포함된 약 800 m의 경로이다. 두 경로에서 일정 수준의 자율주행이 가능하였으며 Fig. 16은 Fig. 15 (b)의 언덕 주행이 포함된 경로에서의 주행 사진이다. 이를 통해 오픈소스를 사용하여 구현한 자율주행 시스템은 실제 환경에서 일정 수준의 주행이 가능함을 확인했다.

웨이포인트의 스무딩에 따른 주행의 정확성을 비교하기 위해 Fig. 15의 (a)인 경로 1에서 스무딩 전, 후의 웨이포인트를 이용한 주행에서의 RMSE를 계산하였다. 오차(error)는 웨이포인트와 웨이포인트에 도달했을 때의 차량 위치의 거리 차이로 계산하였다. 오차에 대한 그래프는 Fig. 17, Fig. 18와 같다. Fig. 17은 0.5 m의 간격으로 저장한 웨이포인트의 주행 오차이며 Fig. 18는 0.5 m의 간격을 1.0 m으로 변경한 후 스무딩을 거친 웨이포인트의 주행 오차이다. RMSE를 계산한 결과 Fig. 17에서의 RMSE는 0.5649 m이었고, Fig. 18에서의 RMSE는 0.2067 m이었다. 이를 통해 웨이포인트에 스무딩을 거친 후 주행하는 경우 RMSE가 감소하였으며 주행의 정확성이 향상함을 확인하였다.

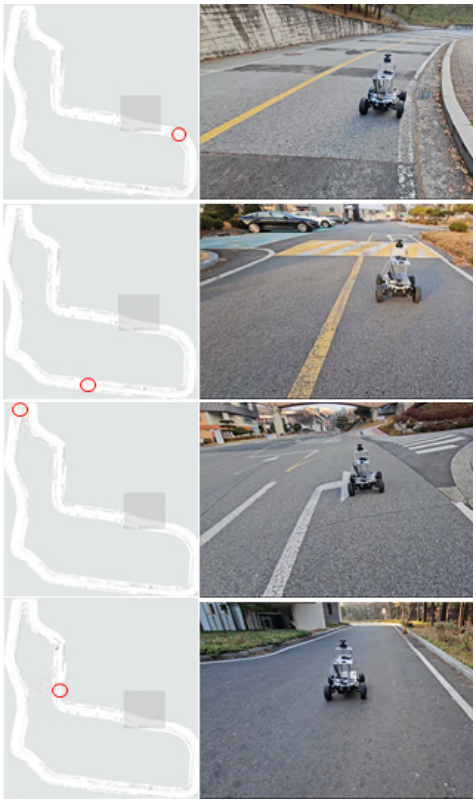


Fig. 16. Autonomous driving in rela environment

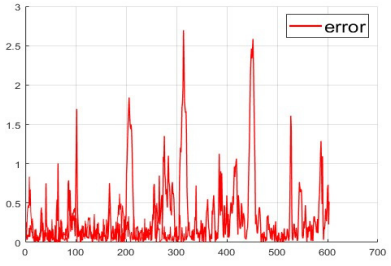


Fig. 17. Error graph before waypoint smoothing

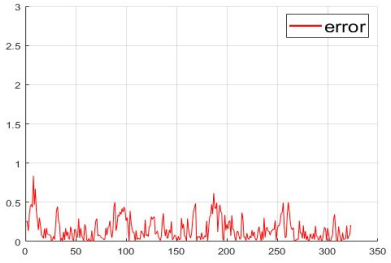


Fig. 18. Error graph after waypoint smoothing

RMSE를 통한 비교를 통해 웨이포인트의 스무딩에 따라 자율주행의 정확성이 증가함을 확인할 수 있었다. [7]의 GNSS가 없는 실외 자율주행 결과의 RMSE인 0.1768 m와 비교하였을 때 본 논문의 결과인 RMSE 0.2067 m은 큰 차이가 없다고 판단되며 추후 카메라를 이용한 객체 탐지 및 차선 유지 등의 기능을 추가하거나 다른 알고리즘들을 비교 후 변경하는 방식을 통해 자율주행의 정확성을 높일 수 있을 것으로 생각된다.

4. 결론

본 논문에서는 자율주행을 위한 하드웨어, 소프트웨어 플랫폼을 구성하고 오픈소스를 이용하여 GNSS 기반의 자율주행 시스템을 구현 및 실제 환경에서 실험함으로써 다음과 같은 결과를 얻었다.

1. 사용자가 자유롭게 수정하거나 원하는 기능만을 종합할 수 있는 범용성이 높은 오픈소스를 사용하여 자율주행 알고리즘을 구현했다.
2. 구현한 자율주행 알고리즘은 평탄한 경로와 언덕 주행이 포함된 경로인 두 경로에 대해 일정 수준의 자율주행이 가능함을 확인했다..
3. 웨이포인트의 스무딩을 통해 자율주행의 주행성능이 향상됨을 확인하였고 추후 카메라를 통한 추가적인 기능이나 다른 오픈소스의 사용으로 자율주행의 정확성을 높일 수 있을 것이다.
4. 센서의 노이즈, 오픈소스들의 적용 및 호환에 어려움이 있었고, 코드의 추가 작성이 필요한 때도 있었지만 이를 극복함으로 오픈소스를 이용한 자율주행 시스템의 개발이 가능하다.

향후 연구로 차선 인식, 객체 탐지의 기능 추가와 인지, 판단, 제어와 관련된 오픈소스들에 대해 비교 및 테스트를 진행해 더 정확하고 안전한 주행이 가능한 자율주행 시스템을 개발하고자 한다.

References

[1] K. H. An, S. W. Lee, W. Y. Han, J. C. Son, "Technology Trends of Self-Driving Vehicles", *Electronics and Telecommunications Trends*, Vol.28, No.4, pp.35-44, Aug. 2013.
DOI: https://doi.org/10.22648/ETRI.2013.J.280404

[2] S. H. Park, Global Trends in Autonomous Vehicle

Industry, Research Report, KDB Industrial Bank Future Strategy Research Institute, Korea, Vol.801, pp.50-66

[3] I Barabás, A Todoruţ, N Cordoş and A Molea, "Current challenges in autonomous driving", *IOP conference series: materials science and engineering*, Vol.252, No.1, Nov. 2017.
DOI: https://doi.org/10.1088/1757-899X/252/1/012096

[4] E. Teskeredzic and A. Akagic, "Low cost UGV platform for autonomous 2D navigation and map-building based on a single sensory input", *2020 7th International Conference on Control, Decision and Information Technologies*, CoDIT, Prague, Czech Republic, pp. 988-993, Jun. 2020.
DOI: http://dx.doi.org/10.1109/CoDIT49905.2020.9263975

[5] N. Jain, A. A. Shah, H. Bollamreddi and M. Kothari, "Development of a Low Cost Autonomous Ground Vehicle", *2022 IEEE International Conference on Autonomous Robot Systems and Competitions* , ICARSC, Santa Maria da Feira, Portugal, 2022, pp. 154-160, Apr. 2022.
DOI: http://dx.doi.org/10.1109/ICARSC55462.2022.9784787

[6] F. Marques, P. Santana, M. Guedes, E. Pinto, A. Lourenço and J. Barata, "Online self-reconfigurable robot navigation in heterogeneous environments", *2013 IEEE International Symposium on Industrial Electronics*, IEEE, Taipei, Taiwan, pp. 1-6, May. 2013.
DOI: http://dx.doi.org/10.1109/ISIE.2013.6563831

[7] A. Carballo, S. Seiya, J. Lambert, H. Darweesh, P. Narksri, L. Morales, N. Akai, E. Takeuchi, and K. Takeda, "End-to-end autonomous mobile robot navigation with model-based system support", *Journal of Robotics and Mechatronics*, Vol. 30, No. 4, pp.563-583, Aug. 2018.
DOI: https://doi.org/10.20965/jrm.2018.p0563

[8] S. Kato, Y. Maruyama, S. Maeda, M. Hirabayashi, Y. Kitsukawa, A. Monrroy, T. Ando, Y. Fujii, T. Azumi, "Autoware on Board: Enabling Autonomous Vehicles with Embedded Systems", *2018 ACM/IEEE 9th International Conference on Cyber-Physical Systems*, ICCPS, Porto, Portugal, pp. 287-296, Apr. 2018.
DOI: http://dx.doi.org/10.1109/ICCPS.2018.00035

[9] M. O'Kelly, H. Zheng, D. Karthik, R. Mangharam, "F1TENTH: An Open-source Evaluation Environment for Continuous Control and Reinforcement Learning", *Proceedings of Machine Learning Research*, Vol.123, pp.77-89, Apr. 2020.

[10] T. Takasu, N. Kubo, A. Yasuda, "RTKLIB: Open Source Program Package for RTK-GPS", *FOSS4G 2009 Tokyo*, FOSS4G, Tokyo, Japan, Nov. 2009.

[11] C. Lim, Y. Lee, A. Cho, and B. Park, "A Review on the Usage of RTKLIB for Precise Navigation of Unmanned Vehicles," *Journal of Positioning, Navigation, and Timing*, Vol. 10, No. 4, pp.243-251, Dec. 2021.
DOI: https://dx.doi.org/10.11003/JPNT.2021.10.4.243

[12] Global Navigation Satellite System Integrated Data Center, Always-on Observatory Guidance [Internet], Global Navigation Satellite System Integrated Data Center, c2015[cited 2015 December 10], Available From: https://gnssdata.or.kr/cors/getCorsView.do (Accessed Mar. 2, 2024)

[13] T. Moore, robot_localization 2.7.5 Documentation [Internet], ROS, c2023[cited 2023 December 12], Availa ble From: https://docs.ros.org/en/noetic/api/robot_localization/html/index.html (Accessed Dec. 14, 2023)

[14] Wikipedia, Universal Transverse Mercator coordinate system [Internet], Wikipedia, c2023[cited 2023 December 10], Available From: https://en.wikipedia.org/wiki/Universal_Transverse_Mercator_coordinate_system (Accessed on: Dec. 14, 2023)

[15] T. Shan, B. Englot, D. Meyers, W. Wang, C. Ratti and D. Rus, "LIO-SAM: Tightly-coupled LiDAR Inertial Odometry via Smoothing and Mapping", *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, Las Vegas, NV, USA, pp.5135-5142, Oct. 2020.
DOI: http://dx.doi.org/10.1109/IROS45743.2020.9341176

[16] M. Kim, M. Zhou, S. Lee and H. Lee, "Development of an Autonomous Mobile Robot in the Outdoor Environments with a Comparative Survey of LiDAR SLAM", *2022 22nd International Conference on Control, Automation and Systems*, ICCAS, Jeju, Korea, Republic of, pp.1990-1995, Nov. 2022.
DOI: http://dx.doi.org/10.23919/ICCAS55662.2022.10003762

[17] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss and W. Burgard, "Octomap: An efficient probabilistic 3d mapping framework based on octrees", *Autonomous Robots*, Vol.34, No.3, pp.189-206, Feb. 2013.
DOI: http://dx.doi.org/10.1007/s10514-012-9321-0

[18] ROS wiki, navigation [Internet], Open Robotics, c2020 [cited 2020 Sep 14], Available From: http://wiki.ros.org/navigation (Accessed Jun. 6, 2023)

[19] C. Rösmann, F. Hoffmann and T. Bertram, "Integrated online trajectory planning and optimization in distinctive topologies", *Robotics and Autonomous Systems*, Vol.88, pp.142-153, Feb. 2017
DOI: http://dx.doi.org/10.1016/j.robot.2016.11.007

강 효 련(Hyo-Ryeon Kang) [정회원]

- 2021년 8월 : 한국교통대학교 기계공학과 (공학사)
- 2024년 2월 : 동대학원 기계공학과 (공학석사)

〈관심분야〉
자율주행, 지능로봇 소프트웨어

정 성 엽(Seong-Youb Chung) [정회원]

- 1994년 2월 : 연세대학교 기계공학과 (공학사)
- 1996년 2월 : 한국과학기술원 기계공학과 (공학석사)
- 2005년 2월 : 한국과학기술원 기계공학과 (공학박사)
- 2005년 4월 ~ 2007년 3월 : 삼성중공업 메카트로닉스센터 책임연구원
- 2007년 4월 ~ 현재 : 한국교통대학교 기계공학과 교수

〈관심분야〉
양팔 로봇, 머신 비전, 지능로봇 소프트웨어