돌발 상황을 대비한 자율주행 시스템 구현*

이정민, 장세희, 윤용익
숙명여자대학교 ICT융합공학부 IT공학전공

ljmpink9966@sookmyung.ac.kr, sehuiui@sookmyung.ac.kr, yiyoon@sm.ac.kr

Autonomous driving system for emergency situations

Jung-Min Lee, Se-Hui Jang, Yong-Ik Yoon
Dept. of Information Technology Engineering, Sookmyung Woomen's University

요 약

자율주행 기술이 고도화됨에 따라 사용자가 주행 상황을 실시간으로 모니터링하고 주행을 제어할 수 있는 자율주행 서비스가 필요하다고 생각했다. 또한, 돌발 장애물을 고려하며 정해진 경로로 주행하는 자율주행을 구현하고자 해당 시스템을 설계하게 되었다. 해당 시스템은 차량, 서버, 애플리케이션으로 구성되어있으며 구성요소 간의 실시간 통신을 통해 차량 주행 상황 및 사용자 제어 명령을 자유롭게 전달하고자 했다. 차량의 자율주행 알고리즘을 구현하기 위해 이미지 데이터 처리에 효과적인 CNN을 활용하여 장애물 회피 모델과 라인 트레이서 모델을 구현하여 해당 모델들을 하나의 솔루션으로 통합하였다. 해당 솔루션 구현을 통해 차량이 마주할 수 있는 돌발 상황에 대처하는 자율주행의 안전성을 높이고자 했으며 자율주행 환경에서 사용자 조작을 용이하게 하고자 하였다.

키워드 자율주행, 돌발 장애물, CNN, 실시간 통신

1. 서론

자율시스템 운행 시 교통사고 발생에 대한 우려, 사용자 조작의 어려움 등 다양한 걸림돌이 있다. 이러한 걸림돌을 해소하기 위해 차량은 직면한 돌발 상황을 회피하는 자율주행을 수행하며, 사용자가 편리하게 자율주행 상황을 실시간으로 모니터링할 수 있는 시스템이 필요하다고 생각했다.

이러한 취지에서 차량이 마주한 돌발 상황을 회피하고, 정해진 라인을 따라 주행하는 자율주행 시스템을 구현하였다. 또한, 서버, 차량, 사용자 조작 애플리케이션 간의 통신을 구현하여 서버를 통해 차량과 애플리케이션이 실시간으로 정보를 주고받을 수 있도록 하였다. 이러한 기능은 사용자가 실시간으로 차량의 주행 상황을 확인하고 간단한 조작을 통해 차량을 제어함으로써 안전성을 높을 수 있다.

본 연구에서는, 이러한 자율주행 시스템에 대한 소개와 개발 과정 및 내용을 소개하고자 한다. 논문의 구성은 다음과 같다. 제2장에서는 현재 자율주행 기술에 관해서 설명하며 제3장에서는 본 시스템인 '돌발 상황을 대비한 자율주행'의 개발 전반적인 부분을 설명한다. 제4장에서는 본 시스템의 구현 환경과 구현 기능에 관해서 설명하며 제5장에서는 본 연구의 결말을 맺는다.

2. 기술 동향

본 장에서는 현재 개발 중인 자율주행 기술에 관해서 설명한다. 2.1절에서는 웨이모의 자율주행 택시를, 2.2절에서는 현대자동차의 '아이오닉 5 로보택시'를 소개한다.

* 이 논문은 2018년도 정부(교육부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임 (No. 2018R1D1A1B07047112)

2.1 현대자동차 자율주행

그림 1. 자율주행 기술의 6단계
(출처 : 현대모터그룹 Tech)

현대자동차는 '아이오닉 5 로보택시'를 공개했으며 해당 차량에 적용된 자율주행 기술은 SAE 기준 레벨 4 수준으로 개발 중이다. 즉, 완전한 무인 주행이 가능한 자율주행차를 출시예정이다.

2.2 웨이모의 자율주행 택시

구글의 웨이모는 Guidehouse에 따르면 전 세계 자율주행기술에서 가장 앞서 있다고 평가를 받고 있다. 현재 샌프란시스코에서 자율주행 택시를 시범 운영 중이며 가속, 감속, 차선변경 등 자동차의 모든 움직임을 차가 스스로 조작하는 완전자율주행 차량이다. 자율주행 미국 SAE 기준 레벨 4단계까지 구현했다고 보고 있으며 2023년에 상용화를 할 수 있을 것이라는 평가도 나오고 있다.

3. 상황인지 기반 자율주행 시스템 개발

그림 2. 자율주행 차량 및 모바일을 통한 차량 조작

기존의 자율주행 기술을 참고하여, 이 연구의 자율주행 시스템 개발은 여기에서 더 나아가 돌발 장애물에 대한 대처 알고리즘을 추가하여 더 안전한 자율주행 서비스를 개발하는 것이다.

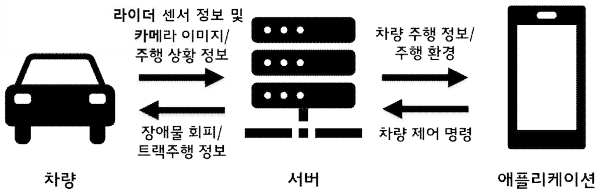

그림 3. 시스템 구성도

전체적인 시스템 구성은 (그림 3)와 같다. 차량이 수집한 이미지 정보 및 라이더 센서 정보를 기반으로 장애물 회피 모델, 라인 트랙 주행 모델을 구성하였다. 서버는 장애물 회피 모델, 라인 트랙 주행 모델의 결과를 바탕으로 자율주행 알고리즘을 구성하여 차량을 제어한다. 이를 통해, 차량은 돌발 장애물을 회피하는 자율주행을 수행할 수 있다. 사용자와 차량은 서버를 중심으로 통신을 주고받는다. 이를 통해 사용자가 자율주행 차량을 조작하고 차량이 마주한 돌발 상황과 이를 회피하는 현 주행 상황을 확인할 수 있다.

3.1 상황인지 기반 자율주행 알고리즘

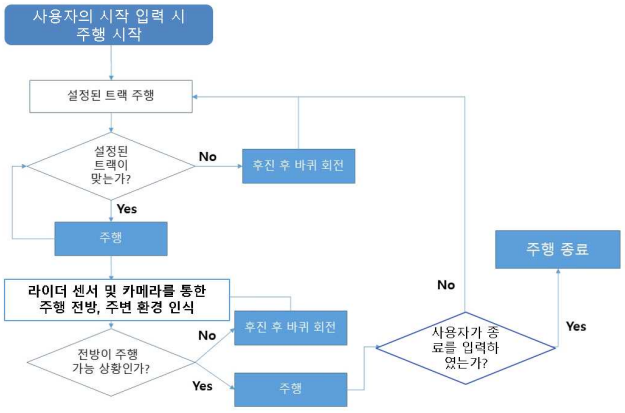

그림 4. 알고리즘 명세

장애물 회피 모델과 라인 트랙 주행 모델의 결과로 구성한 자율주행 알고리즘은 (그림 4)와 같다. 사용자가 속도와 함께 시작을 입력할 경우 주행을 시작한다. 차량은 설정된 속도로 트랙을 따라 주행하고, 설정된 트랙이 맞으면 계속 주행한다. 설정된 트랙이 아니라면 후진 후 바퀴 회전으로 방향을 전환한다. 주행 전방 인식 후, 장애물이 없는 진행 가능 상황일 경우 계속 주행한다. 장애물이 있는 주행 불가 상황일 경우에는 후진 후, 바퀴 회전으로 방향을 전환한다. 사용자가 종료를 입력할 경우 차량의 주행은 종료된다.

그림 5. 장애물 회피 모델과 라인 트랙 주행 모델을 반영한 자율주행 차량 주행

장애물 회피 모델은 학습한 결과를 바탕으로 장애물이 있을 확률을 0에서 1 사이의 값으로 반환한다. 장애물 판단은 먼저 라이더 센서를 통해 장애물과의 거리를 1차로 확인한 뒤 2차로 카메라를 통해 확인한다. 반환받은 값이 0.5 이하일 경우 주행 가능한 상황으로 차량을 계속해서 주행하고, 0.5 이상일 경우 주행 불가한 상황으로 차량의 바퀴 steer 값을 조정하여 후진하여 장애물 회피 주행을 구현하였다. 라인 트랙 주행 모델의 경우 라인을 인식한 이미지와 진행할 지점의 좌표를 이용하여 라인 트랙 주행을 구현하였다.

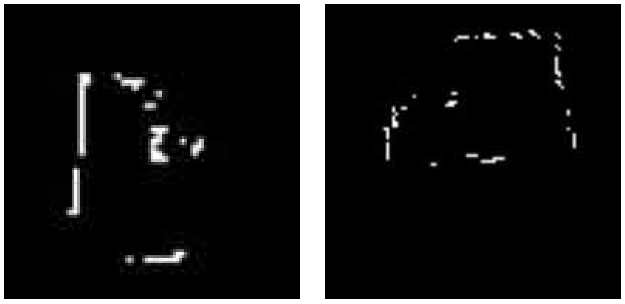

그림 6. 라이더를 이용한 지도

장애물 회피 모델과 라인 트랙 주행 모델이 동시에 적용되며, 모델의 정확도를 높이기 위해 과도한 코너링 상황, 라인을 벗어난 주행 상황 등 다양한 상황의 이미지 데이터 및 라이더 센서 데이터를 수집하였다. 장애물 회피의 경우 주행 가능한 상황, 주행 불가한 상황 두 경우로 나누어 데이터셋을 분류하였다. 장애물을 인식하는 범위의 정확도를 높이고자 카메라뿐만아니라 라이더 센서를 활용하여 다양한 장애물 인식 상황의 데이터를 수집하여 모델에 적용하였다.

3.2 통신을 통한 사용자 명령 전달

차량과 사용자의 실시간 통신을 위해 서버와 애플리케이션 간 통신은 소켓 통신을 활용하여 이를 구현하였으며, 서버와 차량 간의 통신은 무선 연결을 활용하여 구현하였다.

3.3 모바일을 통한 제어 및 주행 확인

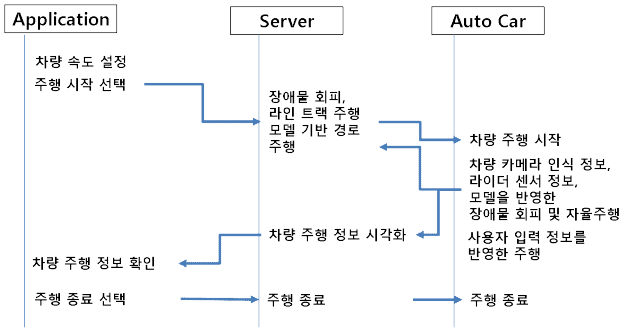

그림 7. 시스템 프로세스

(그림 7)은 시스템의 전반적인 실행 프로세스를 나타낸 것이다. 사용자는 차량의 속도를 설정한 뒤 주행 시작 버튼을 누르면 서버에서 자율주행 알고리즘이 동작하도록 차량에 명령을 전달한다. 명령에 맞춰 차량의 주행이 시작되고, 차량은 주행 중 카메라 정보를 서버에 실시간으로 전달하며 서버는 이를 받아 애플리케이션에 전달해 사용자가 차량 주행 정보를 확인할 수 있다. 주행을 종료하고자 할 때 주행 종료 버튼을 누르면 해당 명령이 서버에 전달되고 서버는 이를 차량에 전달해 주행이 종료된다.

사용자는 모바일을 통해 차량의 속도를 입력하고 주행의 시작, 정지, 종료를 조작할 수 있다. 차량의 현재 주행 상황은 모바일과 시스템의 소켓 통신을 통해 실시간으로 파악할 수 있으며 사용자 명령을 모바일을 통해 받도록 했다.

4. 구현 내용 및 분석

4.1 개발 환경

'돌발 상황을 대비한 자율주행 시스템'의 개발 환경은 다음과 같다. 차량 서버는 Ubuntu 기반의 ROS를 사용하였다. 모바일 통신은 Socket.io 라이브러리를 사용, 서버 내에서는 Python으로 구현한다. 안드로이드 IDE를 설치하여 모바일 애플리케이션을 구축하고 Android 4.2를 이용하여 갤럭시M 12에 적용해 개발한다.

4.2 서버 구현

그림 8. 서버에서 확인되는 차량 전방 상황

서버는 차량의 제어 및 사용자 애플리케이션 간의 중심 역할을 한다. Python을 기반으로 학습한 장애물 회피 모델, 라인 트랙 주행 모델의 결과를 기반으로 차량을 제어한다. (그림 8)과 같이 서버에서 차량이 인식하는 실시간 차량 전방 상황이 확인되고, 이 화면을 사용하여 라인 트랙 주행 모델의 진행 지점, 장애물이 있을 확률이 결정된다. 서버는 사용자와 차량의 소켓 통신의 서버로 연결을 주관하는 역할을 도맡는다.

4.3 클라이언트 구현

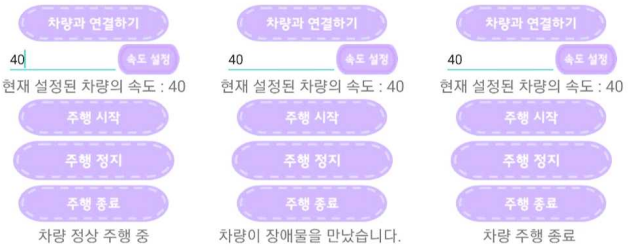

그림 9. 애플리케이션 UI

(그림 9)는 사용자에게 보이는 안드로이드 화면을 나타낸 것으로 차량과 연결하기 버튼으로 차량과 연결이 되고, 속도를 설정할 수 있다. 주행 시작, 정지, 종료 버튼으로 차량을 조작할 수 있으며, 화면 하단에는 차량이 정상 주행 중인 상황, 장애물을 만난 상황, 주행 종료한 상황 등을 표시하여 차량이 마주한 돌발 상황을 사용자가 확인할 수 있도록 하였다.

5. 결론

본 연구에서는 차량 제어 애플리케이션으로 클라이언트의 명령에 맞춰 차량의 자율주행을 손쉽게 조작하고, 돌발 장애물에 대비한 자율주행 알고리즘을 구현하여 차량에 적용함으로써 차량 주행의 안전성을 높이는 데 중점을 두었다. 본 연구에서 알 수 있듯이 클라이언트 애플리케이션을 통해 실시간으로 차량의 주행 정보 확인이 가능하며 차량은 정해진 경로를 따라 주행하는 과정에서 돌발 장애물을 회피하며 안전한 주행을 한다. 이는 자율주행 시스템의 안정성을 높이는데 이바지할 것으로 기대하고 있다.

향후 애플리케이션을 통해 차량의 주행 정보를 더욱 자세히 확인할 수 있도록 차량의 위치 정보 및 주행 영상을 실시간 스트리밍함으로써 자율주행의 안전성 확보하고, 자율주행 알고리즘에 실시간 도로 교통 정보 및 목적지 설정 기능을 더함으로써 자율주행의 정확성을 확보할 수 있을 것으로 판단된다. 더 나아가 장애물을 예측하고 이를 우회할 수 있는 알고리즘을 개발하는데 기초가 될 것으로 생각한다. 이를 위해 추가적인 연구 및 기술개발이 필요할 것으로 예상한다.

참고문헌

[1] 자율주행이 만드는 새로운 변화 - 삼정KPMG 경제연구원, 삼정 Insight 제69호

[2] Automated Driving Systems - Guidehouse Insights

[3] AIoT AutoCar Guide Book - Hanback Electronics

[4] Hyundai Motor Group Tech , “자율주행” - https://tech.hyundaimotorgroup.com/kr/mobility-device/autonomous/