게임 엔진 기반 Deep Drive 플랫폼 개발

김혁주, 김건하, 이은성, 곽동신 (인하대학교 전기공학과)
김수영 (인하대학교 전자공학과)
임성비 (인하대학교 조선공학과)
e-mail:power940509@naver.com

Development of Game Engine Based Driving Simulator Platform

Hyeok-Ju Kim, Geon-Ha Kim, Eun-Sung Lee, Dong-Shin Kwak,
Su-Young Kim, Seong-Bi Lim

요 약

Deep Drive 플랫폼 개발 기술인 HILS(Hardware In the Loop Simulation)는 시스템 모델에 근간을 둔 실시간 시뮬레이션 기법과 H/W를 접목시킨 실시간 해석 기법으로, 플랫폼 개발으로 현재는 난해성이 있는 자율 주행 자동차 실차의 테스트 베드를 구현하여, 여러 가지 실험을 통해 실제 차의 운행에 대한 피드백을 제공, 운전자로부터 안전을 보장하기 위한 기술 보장. 기술의 상용화, 양산화 연구에 도움을 줄수 있음.

Ⅰ. 서론

최근 자동차 산업의 주요 핵심 트랜드는 '커넥티비티'와 '자율주행차'이며 완성차 업계 뿐만 아니라 ICT기업 또한 자율주행차 기술개발에 나서고 있다. 기존의 엔진 + 파워트레인 기술 및 규모의 경제논리로 자동차 시장의 우월한 지위를 유지하던 전통 완성차 업체는 자동차 산업의 패러다임 변화(융복합, 대체 동력, 공유경제, 가전화)에 좀 더 유연하게 대처하는 ICT기업의 도전을 받고 있는 상황이며, 인공지능 분야에서 혁신적인 발전을 가져온 딥러닝 알고리즘의 개발 및 기계 학습 기술이 자율주행차의 다양한 범주에 적용이 가능하고 또 예측하기 힘든 파급력을 보여 주고 있다. 이로 인해 자율주행 자동차 업체 수요는 증가하고 있음을 알 수 있다.

그러나 자율주행 자동차를 설계하고 시험하는 데에 있어 실차의 테스트 베드를 구현하는데 기술적으로 어려울 수 있고, 테스트 과정에서 자율주행 차량의 사고 발생 시 현재는 책임 대상자의 대한 명확성이 없다. 또한 개발과정에 있어 ECU 오류검사 과정과 비용의 증가는 불가피하다.

위와 같은 어려움을 해결하고자 Deep Drive 플랫폼을 개발하게 되었다. Deep Drive 플랫폼은 PC의 게임과 모형의 스티어링 휠, 엑셀레이터, 브레이크를 이용하여 가상의 현실에서의 테스트 환경을 구현하고 이를 자율주행차량 개발자들에게 제공하는 방안을 마련하게 되었다. 실차를 직접 구현하는 것에 비해 적은 비용이 요구될 것이고, 사고 역시 가상 현실(PC 게임)내에서 발생하므로 안전문제와 책임에 관한 문제 역시 해결할 수 있다.

Ⅱ.Deep Drive 플랫폼의 특징 및 장점

기존 Logitech사의 G29 게이밍 휠의 프레임 만을 사용하여 하드웨어 조이스틱 자체 MCU 구동 제작을 하고 사용자의 운전 동작 내용을 개별적 저장장치에 저장한다.

자율주행 환경구성을 위해 actuator에 제어 모터를 설치하여, 신호에 맞게 제어 가능하고
Handle 부 모터 제어는 내장된 모터와 엔코더 사용, Pedal부는 외부 서보모터 사용하여 각도를 제어한다.

플레이 백 구동 시, 저장되어있던 내용을 재생시켜 하드웨어 동작.

영상처리 후 차량의 이동에 맞는 신호를 생성하여 Handle, Pedal을 물리적으로 구동시킨다.

게임엔진으로는 GTA5(Grand Theft Auto5)를 사용하였다.

Ⅲ. Deep Drive 플랫폼의 기능과 구동모드

1.Deep Drive 플랫폼의 적용 기능

가. 조작 레코딩 기능 : 가상환경 내 사용자가 조작한 커맨드 값을 기록 후 플레이 하는 기능.

나. 경로 추적 기능 : 장애물이 없는 경로트랙에 목표 좌표를 입력하면 해당 좌표로 자동 이동하는 기능.

다. 자율주행 Level 1 기능 : 신호등 및 차선 인식 후 사용자의 운전을 보조 하는 기능.

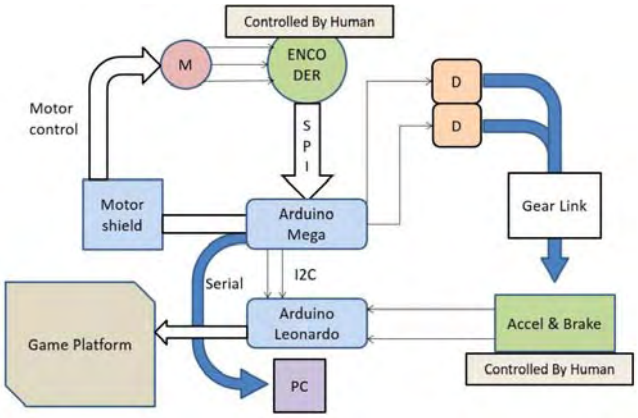
라. 시뮬레이션 기능 : 운전자의 가상 주행 시뮬레이션을 통해 사고 위험성을 판단하는 기능.

마. 데이터 수집 기능 : 다수의 운전자의 데이터를 수집해 가상 환경 내 운전자 프로필을 등록, 예측하는 기능.

2.Deep Drive 플랫폼의 구동모드

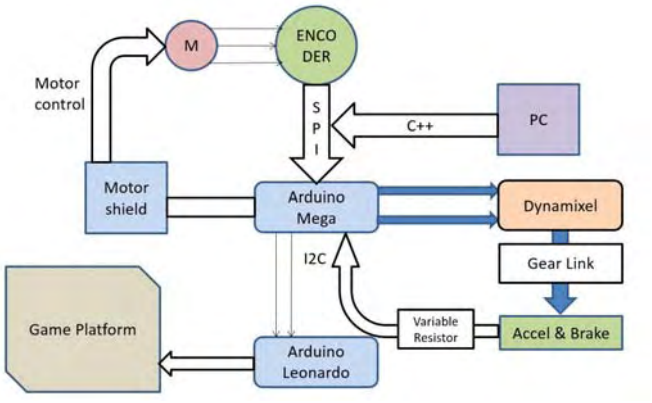
가. 플레이 모드 : 1차 목표인 플레이모드는 기존의 게임 컨트롤러가 아닌 자체 제작을 통한 컨트롤러 구현으로 이후 연구에서 사용자 설정으로 인한 추가 사항을 쉽게 수정 가능한 환경을 부여하는 의미가 있다. 그리고 추가적으로 컨트롤러 MCU에 신호를 부여함으로서, 사람의 직접적인 조작이 아닌 신호로서 컨트롤러를 제어할 수 있는 환경을 구현.

나. 레코딩 모드 : 자율주행 연구를 위하여 사용자의 주행 습관 등의 데이터를 기록하고 분석하기 위해, 운전자의 실제 운전 데이터를 실시간으로 기록하여 데이터화 하는 모드. PC를 이용하여 저장.



[그림 1] 레코딩 모드의 하드웨어 알고리즘

다. 플레이 백 모드 : 레코딩 모드를 통해 기록된 운전자의 운전 데이터를 분석, 또는 연구하기 위해 시뮬레이션 상에 재생하는 모드, 기본적으로 이 모드를 실행하게 되면, 이전에 운전자가 운전하여 PC에 저장되었던 데이터내의 경로와 파라미터들이 동일하게 입력되어, 시뮬레이션 상의 개체가 움직이게 된다.



[그림 2]플레이 백 모드의 하드웨어 알고리즘

라. 자율주행 모드 : 기존의 플레이 모드, 레코딩 모드, 플레이 백 모드에 시뮬레이션 내의 환경을 영상처리 하여 시뮬레이션 내의 여러 오브젝트를 인식하고, 필요한 파라미터를 생성하는 소프트웨어를 구현하여 주행에 필요한 data를 Serial 통신을 통하여 하드웨어에 전달 받고 해당 값에 대하여 즉각적으로 Hardware Actuator를 제어하는 모드이다.

이후에 시뮬레이션 내에 추가적인 값들을 받아 처리하면, 실제 환경에서 제한적인 연구 방법까지 실현 가능하다.

최종 모드이며, 자율주행 연구의 테스트베드가 될 모드이다.

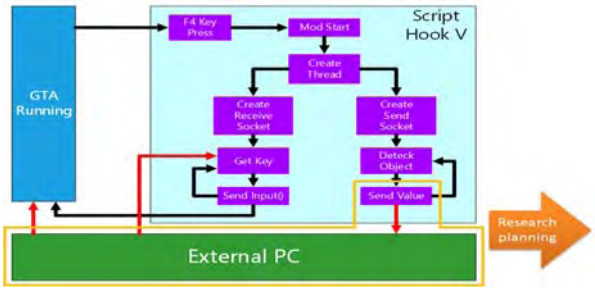
[그림 3] 자율주행 모드의 최종 시스템 구성

Ⅳ. Deep Drive 플랫폼의 하드웨어 구조 및 제어

상기 플랫폼의 하드웨어는 기본 MCU장치로 Arduino Mega, Leonardo를 사용하고 있으며, 스티어링 휠 내의 DC모터(24V,2개), 엑셀레이터, 브레이크, Encoder, 그리고 엑셀레이터와 브레이크를 기계적으로 제어할 다이나믹셀(AX-18A)를 포함하고 있다.

[그림 4]Arduino Mega

[그림 5]하드웨어 구성



[그림 6] HILS 알고리즘

1. 하드웨어 구조

가. MCU to GAME

동작하고 있는 Actuator의 모션에 따른 위치값을 GAME PC로 전달하여 GAME 상에서 작동이 되도록 설계 하였다. 먼저 Actuator의 값을 MCU로 사용하고 있는

Arduino MEGA2560 으로 읽어 들이도록 하였다. 이 때, Arduino MEGA는 게임 시뮬레이션과는 별개의 PC에 연결하였다. 그리고 읽어 들인 값을 GAMING PC에 전달하기 위하여 Arduino Leonardo를 HID 장치로 인식하게끔 설계하였다. Arduino MEGA에서 Leonardo로 신호를 전달하고 이를 게임으로 전달할 수 있도록 구현하였다.

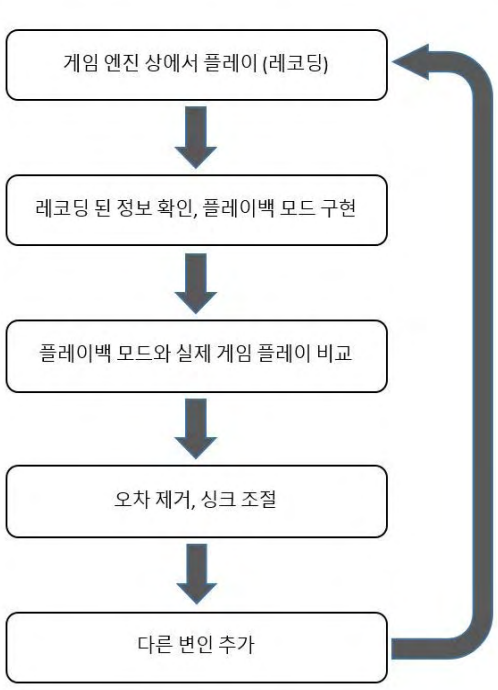
나. wheel

휠을 제어하기 위하여 외부에서 힘을 가해주는 것이 아닌, 휠에 내장되어 있는 모터를 이용하였다. 이 모터는 Magnetic Encoder를 장착하고 있어 SPI 통신을 통하여 MCU에 휠의 위치값이 들어오도록 설계되었다.

다. accel, brake

accel과 brake를 제어하기 위하여 ROBOTIS 사의 Dynamixel ax-18a 모터를 이용하였다. 각각의 accel과 brake에 ax-18a를 3d 프린팅으로 만든 링크를 이용하여 부착하였다. accel과 brake의 위치값은 가변저항을 통해 직접적으로 MCU에 들어오도록 설계되었다.

2. Actuator 모터제어



[그림 7] 제어 과정 도식도

게임 환경으로부터 도로 정보(차선, 신호, 보행자 등)를 받아와 그에 맞게 Actuator를 자동 제어하도록 하였다. 도로정보를 읽어 들이고 그에 맞게 움직여야 하는 wheel, accel, brake 모터의 위치 기댓값이 계속적, 실시간으로 들어오도록 하였다. 들어온 위치 기댓값에 맞게 Actuator를 제어하기 위하여,

먼저 wheel 부분은, 내부 모터에 장착되어 있는 엔코더 값을 위치 기댓값과 계속적으로 비교하도록 하였다. 엔코더 값, 즉 현재 휠의 위치값을 위치 기댓값에 근접시킴으로써 자동 제어한다.

그리고 accel, brake 부분은, accel과 brake 페달에 장착되어있는 가변저항을 통해 현재 위치값을 읽는다. 마찬가지로 도로정보에 따른 accel과 brake의 위치 기댓값에 현재 위치값을 근접시킴으로써 자동제어한다.

Ⅴ. Deep Drive 플랫폼의 사용 기술

1. 통신

가. SPI 통신을 이용하여 wheel의 각도와 회전수에 대한 정보를 얻고, 레코딩 모드와 플레이백 모드를 구현하기 위해 사용할 정보들을 수치화 시켜서 다양한 방법으로 인식한다.(ex. text 파일)

나. I2C 통신을 활용하여 게임 엔진이 장치로 인식할 수 있는 Arduino Leonardo에 수치화된 정보(wheel, accel, brake)를 전송, 수신된 정보를 통해서 Actuator 구동시키고, 게임 엔진 상에서 동작할 수 있도록 사용한다.

다. PC를 사용하여 Serial Monitor에 레코딩 모드의 정보를 저장한 뒤, 플레이 백모드에 구현되도록 활용한다. 저장되고 활용되는 정보는 텍스트 파일로서 수치화된 정보를 의미한다.

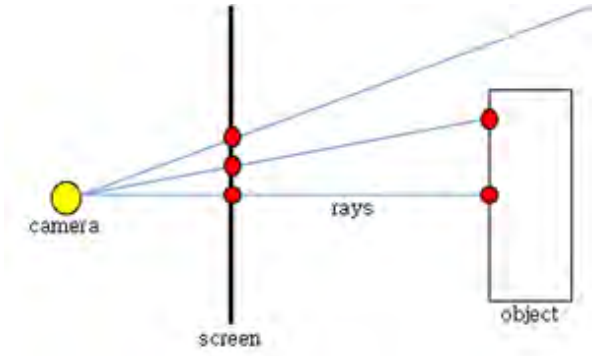
2. Ray Casting

주변 Object 데이터의 취득 방법으로 게임 내 구현된 Ray Cast 관련 Native Function를 이용한다. Ray Cast란 [그림 8]과 같이 임의의 두 점을 인자로 주었을 때, 두 점을 잇는 보이지 않는 레이져를 발사하는 것을 말한다. 이렇게 발사된 레이저에 Detect된 Object의 ID를 가져올 수 있고, 해당 ID를 통해 Object의 여러 정보를 얻을 수 있다. Algorithm 1은 Ray Casting을 통한 Obejct 정보 획득의 예시이다.

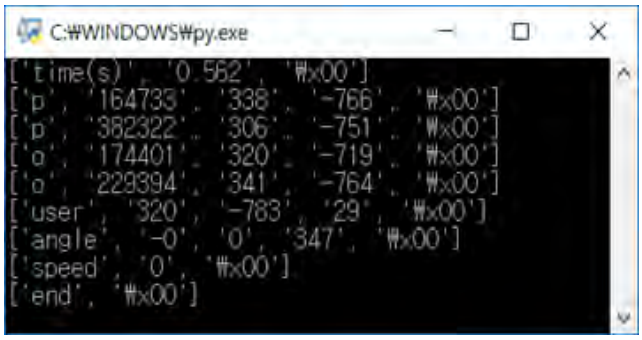
3. Open CV

Ray Casting을 통해 주변 Object에 대한 Truth Value를 획득 할 수 있었다면, Open CV를 통해 게임 화면 속 가상 현실에 대한 Vision Data를 획득 할 수 있다. 게임 화면을 캡쳐하는 방법으로 MFC Programming, Python Package, 외부 PC의 Web Cam을 이용한 방법 등이 있다. 본 연구에서는 Python의 Python Imaging Library(PIL)를 이용하여 GTA V의 화면을 캡쳐하여 사용하였다.

원본 게임화면 및 OPENCV 적용화면



[그림 8] Ray casting의 원리



Ray casting으로 얻어낸 주변 오브젝트 truth data.

4. Inter-Process Communication (IPC)

데이터를 처리하는 방법으로 게임 내에 Thread를 추가로 생성하여 데이터를 처리하는 방법과 Multi Processing을 이용하여 데이터를 처리하는 방법 두 가지가 있다. 본 연구에서는 Multi Processing을 이용하였는데, thread를 이용할 경우 안정성의 위험과, 게임 내에 연산량이 많아져 원활한 게임 진행이 어렵다는 문제가 있었기 때문이다. Multi Processing을 이용할 경우 process간 데이터 공유에 관한 문제가 발생하는데 이는 IPC를 이용하여 해결이 가능하다. IPC의 종류로는 Shared Memory, PIPE, FIFO, Socket 등이 있고 본 연구에서는 PIPE를 통한 IPC를 구현하였다.

5. Key Event

PIPE를 통해 전송 받은 데이터 및 Vision 처리를 통해 얻은 여러 데이터를 종합하고, 처리한 결과는 게임에 Key Value로 전송되어진다. 이때 게임에 Key Event를 발생시키기 위해 WINAPI에서 제공해주는 SendInput Method를 사용하였다.

Ⅵ. 결론

본 하드웨어 플랫폼은 Rockstar Advanced Game Engine를 기반으로 DLL Injection. 및 Ray Casting으로 주변 사물에 대한 Truth Value 취득과 OpenCV를 이용해 화면에 대한 Vision Data를 취득하기 위해 제작되었다. 이 자료는 앞으로 제작할 ADAS 알고리즘 제작을 위한 HIL-System의 주변 환경 모델링 데이터취득의 기반 인프라로 사용할 것이다.

현재 레코딩모드로 저장된 데이터를 플레이백 모드로 시연 시 약간의 타임스탬프 딜레이 현상이 발생하고 있고 이를 보안하는 작업을 진행중에 있으며, 다양한 시뮬레이션 주행환경 데이터를 생성중에 있다.

또한 또한 단계적으로 Software, Hardware, Vehicle을 통합하여 자율주행차 전체 시스템 개발에 단계적/통합적으로 적용 가능한 Software-In-the-Loop System(SILS), Hardware-In-the-Loop System)(HiLS),

Vehicle-In-the-Loop System(VILS) 개발 환경을 구현중 에 있다.

Ⅶ. 참고문헌

[1] 이건용, (2010), “실차 환경 시험의 제약 극복 : 다양한 용도로 실용화되는 HILS”, AUTOMOTIVE, 서울

[2] 홍태욱, 권재준, 박기홍, (2012), “로봇 조향 기반 EPS HILS 시스템의 개발 및 검증”, 한국정밀공학회지, 제 30권 1호, pp.85-95

[3] A. Filipowicz, J. Liu, A. Kornhauser. Using Virtual Worlds, Specifically GTA5, to Learn Distance to Stop Signs, Princeton Specialty Conference, Boston. 2016

[4] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, et al. End to end learning for self-driving cars. arXiv preprint arXiv:1604.07316, 2016.

[5]B. C. Russell, A. Torralba, K. P. Murphy, and W. T. Freeman, “Labelme: a database and web-based tool for image annotation,” International Journal of Computer Vision, vol. 77, no. 1-3, pp. 157 - 173, May 2008.

[6]M. Johnson-Roberson. C. Barto. S. Sridhar “Driving in the Matrix: Can virtual worlds replace human-generated annotations for real world tasks?”. arXiv:1610.01983v1 [cs.CV]. Oct 2016.

언리얼 엔진기반
Rockstar Advanced Game Engine

GTA5 엔진은 위 두문구로 대체