OpenCV를 이용한 딥러닝 기반 아두이노 자율주행자동차

프로젝트 보고서

정보컴퓨터공학부
프로젝트 팀명 : 화이팅
팀원
201424459 박재현
201124455 박지훈

목 차

1. 과제 목표

1.1 과제 목표

본 졸업과제는 아두이노를 이용한 RC자동차에 대해 딥러닝에 의한 자율주행시스템을 구축하는데 있다. 즉 RC자동차를 무인 주행할 수 있게 하기 위한 제반 하드웨어 및 소프트웨어 시스템을 구축한다. 구체적으로 첫째, RC자동차를 위한 하드웨어 운영환경을 구축한다. 둘째, 딥러닝에 의한 자율주행을 위한 컴퓨터 프로그램을 개발한다. 프로그램은 파이썬, OpenCV, 아두이노 언어 등을 사용한다. 마지막으로 개발한 다층퍼셉트론(MLP)을 학습시켜 자율주행 자동차를 구현한다.

2. 대상 문제 및 요구조건 분석

2.1 문제 분석

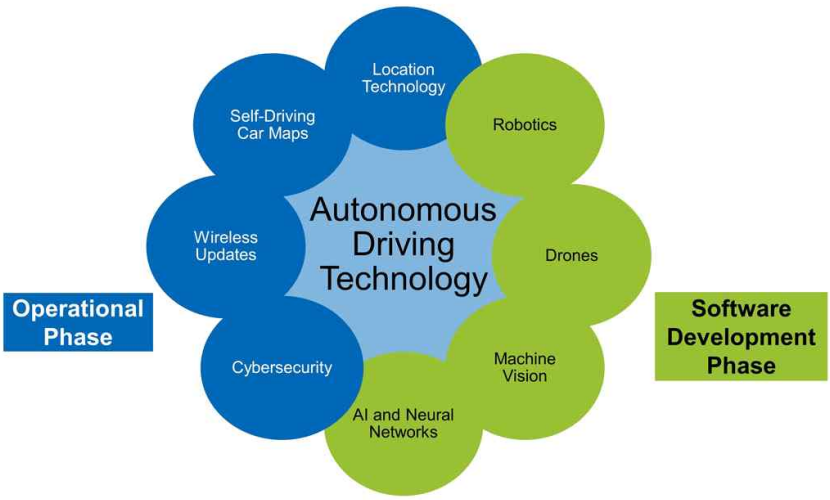
최근 자율주행 자동차를 개발하기 위해 다양한 노력이 이루어지고 있다. 4차 산업혁명의 가장 파괴적이고 혁신적인 자율주행 자동차는 경제, 사회 전반에 걸쳐 가장 광범위한 변화를 이끌 것으로 평가되고 있다. 따라서 세계의 주요 IT기업, 자동차 기업은 자율주행 자동차 시장을 선도하기 위해 노력하고 있다. 그 선두에 Google의 자율주행 자동차가 있다.

<그림 1> Google의 자율주행자동차

자율주행 자동차 기술은 빅 데이터, 인공지능, 사물인터넷, 지능형 로봇 등 다양한 첨단 기술을 필요로 하고 있다. 자율주행이 사람을 대신하여 운행한다는 것은 사람의 편리성을 극대화할 수 있고 시간의 절약효과를 볼 수 있다. 운전자의 부담을 줄일 수 있고, 운전상황을 보면서 개인적인 일을 할 수도 있게 한다.

하지만 안전이 보장되는 자율주행 자동차를 활용하기 위해서는 다양한 노력이 이

루어져야 한다. 먼저, 자율주행이 가능할 수 있는 기술 확보가 되어야 한다. 이러한 기술에는 인공지능 기술, 시각인식 기술, 다양한 센서처리 기술, 딥러닝 기술 등 다양한 기술이 요구되고 있다.



<그림 2> Google의 자율주행 기술

또한 자율주행 자동차가 도로를 돌아다니기 위해서는 자율주행을 위한 법적, 제도적 기반도 마련되어야 한다.

2.2 요구조건 분석

본 과제에서는 자율주행을 할 수 있는 RC자동차를 개발하려고 한다. 이를 위한 요구조건은 다음과 같다.

① 카메라에서 인식되는 도로(루트)는 정확하게 인식되어야 하고 경로를 벗어나면 안 된다.
② 주행 중 신호등이 있으면 이를 인식하고, 빨간불이면 정지 파란불이면 계속 운행을 한다.
③ 딥러닝을 위한 충분한 트레이닝데이터를 가지고 있어야 한다.
④ 자동차의 운행은 안전을 최우선으로 하여야 한다.
⑤ 여러 각도나 빛의 밝기에 따라서도 정상적으로 운행되어야 한다.
⑥ 서버와 주기적으로 통신이 되어야 한다.

2.3 현실적 제약사항 분석 결과 및 대책

자율주행 자동차를 개발하기 위해서는 다양한 기술이 필요하며 법적, 제도적 제도 마련도 필요하다. 그러나 본 졸업과제에서는 다음과 같은 제약조건하에서 자율주행을 할 수 있는 RC자동차를 개발하려고 한다.

① 도로에는 RC자동차 한 대만이 운행을 한다.

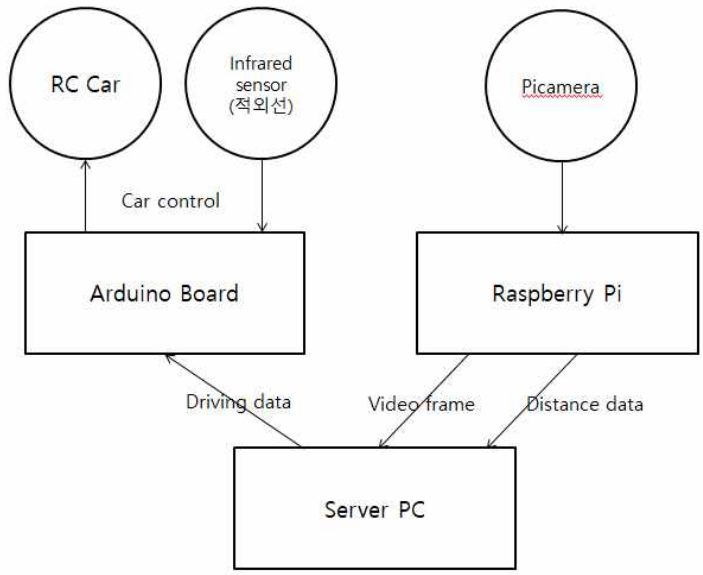
② 자동차가 주행하는 도로는 직접 만든 모형 도로에서 운행한다.
③ 자동차는 운행 중 신호등, 장애물을 만나는 것을 가정한다.

3. 시스템 설계

이 장에서는 자율주행자동차 개발을 위한 시스템 구성과 시스템 작동 단계에 대해 기술한다.

3.1 전체 시스템 구성

본 과제에서 개발할 RC자동차에 대한 자율주행 시스템 구성은 다음 <그림 3>과 같다.



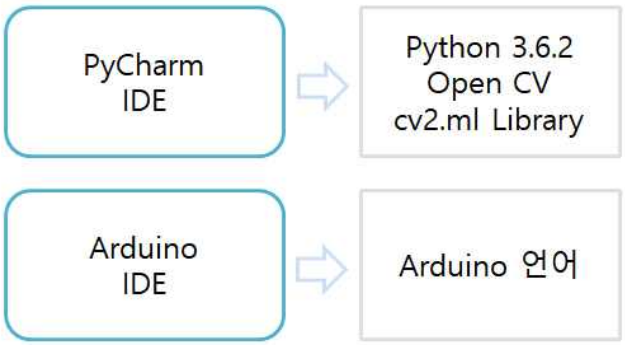
<그림 3> 자율주행 시스템의 구성

1) 하드웨어 구성

① RC 자동차(RC Car) : 모형 도로 상에서 건전기의 전원에 의해 실제 주행할 자율주행 자동차이다.
② 아두이노 보드(Arduino Board) : RC자동차를 제어하기 위한 마이크로컨트롤러 보드이다. 자동차에 대해 전진, 좌측, 우측, 후방, 정지 등을 제어한다.
③ 라즈베리 파이(Raspberry Pi) : 파이카메라를 탑재하고 있으며 운행 시 이로부터 실시간 비디오 프레임을 서버에 전송한다.
④ 서버 PC : 샘플데이터를 생성하거나 다층퍼셉트론(MLP)을 트레이닝하고 학습된 데이터를 이용하여 RC자동차의 운행을 제어한다. 이를 위해 아두이노 컨트롤러와 블루투스 통신을 한다.

2) 개발 소프트웨어 환경

자율주행 RC자동차를 개발하기 위해 PyCharm IDE(Integrated Development Environment)와 Audino IDE를 사용한다.



<그림 4> 소프트웨어 개발 환경

딥러닝에 의한 자율주행 프로그램을 작성하기 위해 PyCharm IDE에서 파이썬, OpenCV를 사용하여 프로그램을 작성한다. 특히 신경망학습을 위해서는 CV2.ml 라이브러리를 활용한다. 또한 아두이노에서 RC자동차를 제어하기 위한 코드는 Arduino IDE에서 Arduino언어로 작성한다.

3.2 시스템 작동 단계

RC자동차에 대한 자율주행 시스템의 작동은 크게 3단계로 구분하여 생각할 수 있다.

1) 인지 단계

라즈베리파이 보드에 장착된 파이카메라, 적외센 센서를 이용하여 차량 외부의 환경을 인지한다. 파이카메라를 통해 입력된 비디오 스트림은 라즈베리파이에서 비디오 프레임으로 변환하여 주기적으로 서버 PC에 전달한다. 적외선 센서를 통해 입력된 데이터는 아두이노에서 영상에 따른 주행보다 우선적으로 주행을 제어하여 충돌을 방지한다.

2) 판단 단계

라즈베리파이로부터 수신한 비디오 프레임과 장애물과의 거리 데이터로부터 서버 PC에서는 이들 데이터들을 분석하여 다층퍼셉트론(MLP)에 의해 학습된 데이터를 기반으로 주행상황을 판단하고 RC자동차를 제어할 정보를 블루투스 통신을 통해 아두이노 마이크로컨트롤러에 전달한다. 이 때 비디오 프레임은 자동차의 운행 방향을 제어하기 위해 학습된 신경회로망으로부터 사용된다. 그리고 장애물과의 거리

데이터를 RC자동차를 정지를 제어하기 위해 사용한다.

또한 푸른 신호등과 빨간 신호등은 자동차의 전진과 정지를 제어하기 위해 사용된다.

3) 제어 단계

아두이노 마이크로컨트롤러는 블루투스 통신을 통해 서버 PC로부터 전달받은 제어정보를 이용하여 RC자동차를 제어한다. 제어는 전진, 중지, 왼쪽, 오른쪽 등의 제어정보를 자동차에 보냄으로써 수행된다.

4. 시스템 개발

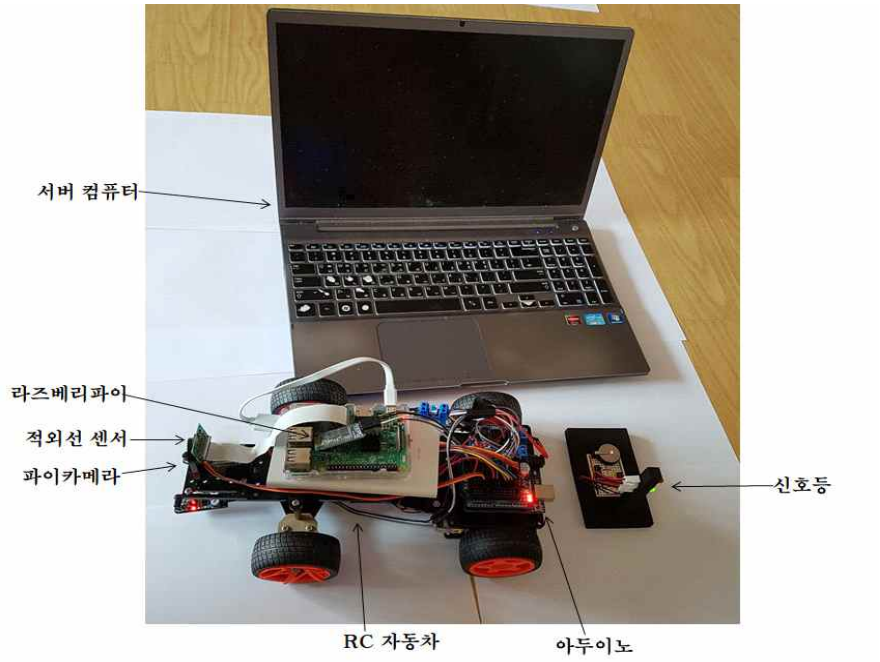
이 장에서는 자율주행자동차를 개발한 단계와 개발에 사용된 주요 알고리즘과 개발 결과에 대해 기술한다. 즉 다층프셉트론(MLP, Multilayer Perceptron)을 이용한 딥러닝에 의한 자율주행 자동차는 다음의 단계로 개발되었다.

- 하드웨어 구성
- 신경회로망 학습 모듈 개발
- 학습데이터를 이용한 자율주행 프로그램의 개발
- 자율주행 실험

4.1 하드웨어 구성

1) 하드웨어의 조립 및 서버 컴퓨터

자율주행자동차를 개발하려면 먼저 자율주행시스템을 구성하는 장치들을 구입하여 조립하는 과정을 거쳐야 한다. 즉 RC자동차, 아두이노 보드, 라즈베리파이 보드, 신호등, 파이카메라, 적외선 센서 등 필요한 하드웨어 장비들을 구입하여 이들이 제대로 작동할 수 있도록 조립하였다. 조립된 하드웨어 시스템의 구성은 <그림 5>와 같다.



<그림 5> 조립된 하드웨어 시스템

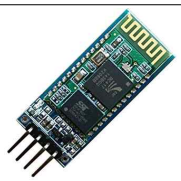
조립 단계에서의 주요 작업은 RC 자동차의 조립, Arduino 보드와 RC자동차의 연결, Raspberry Pi와 Picamera의 연결, 장애물 인식을 위한 적외센 센서 2개 장착 및 아두이노와 서버간 통신을 위해 블루투스 모듈 HC-06 사용, 신호등의 조립 등이다. 그리고 본 과제에서 사용된 주요 부품들을 보면 <표 1>과 같다.

2) Raspberry Pi 와 Server(PC) 간의 socket 통신

러닝에 필요한 샘플 데이터를 얻기나 자율주행에 필요한 영상을 보내기 위해 Raspberry Pi와 PC(server)를 TCP/IP socket 통신을 사용하여 실시간으로 영상 스트리밍을 하기 위한 파이썬 코드를 구현하였다.

3) 아두이노 코드 구현

주행을 위한 모터 드라이브 제어, 방향 제어를 위한 서보모터 등 기본적인 세팅을 구현하고 아두이노와 PC간의 통신을 위해 HC-06 블루투스 모듈을 사용하여 블루투스 통신을 구현하였고 통신되는 우리가 정한 문자에 따라 주행 방식을 다르게 하여 모터 드라이브를 제어하였고 적외선 센서 2개를 앞에 장착하여 앞에 장애물이 존재할 경우, 즉 적외선 센서가 감지되는 동안에는 통신되는 문자에 상관없이 주행을 멈추게 구현하였다. 이렇게 영상처리 주행보다 장애물 감지 주행에 우선순위를 주어서 충돌 방지가 가능하도록 하였다.

Pi camera		자동차가 주행을 하면서 샘플데이터 수집 및 학습모델에 따른 주행에 필요한 input 데이터를 얻기 위해서 카메라 모듈로 Raspberry Pi와 호환이 좋은 Picamera를 사용하였다.
DC Motor Drive Module		RC자동차에서 사용하는 모터 드라이브다. 모터 드라이브를 사용함으로서 보다 편하게 모터를 제어할 수 있기에 모터 드라이브를 사용하여 Real Gear 모터 2개를 장착 및 제어하였다.
Tower Pro Micro Servo MG995		아두이노와 서보모터를 연결하여 자동차 주행에 있어서 턴(회전)을 제어한다. 즉, 주행 방향을 물리적으로 제어한다.
Step-down DC-DC Converter module		입력전압을 낮은 전압으로 바꾸는데 사용된다. 전압변환과정에서 코일을 사용하는데 코일은 이론적으로 손실이 0이므로 변환효율을 높이기 위해 사용하였다.
Sensor Shield		아두이노에서 모듈들을 연결할 때, 부족한 핀 수를 보충하기 위하여 센서 쉴드를 사용하였고 여기에 블루투스, 컨버터, 서브모터, 모터 드라이브, 적외선 센서 등을 연결하였다.
HC-06 Bluetooth		HC-06 Bluetooth은 직렬포트 통신모듈로써 PC에서 아두이노로 주행을 제어하기 위하여 사용하였다.
Obstacle Avoidance Module		아두이노 자동차 전방에 2개를 좌우로 설치하여 전방에 장애물이 감지하여 충돌 방지를 위해 사용하였다.
Arduino		마이크로 컨트롤러보드를 사용하여 만들어진 개발보드로서 자동차의 제어를 직접적으로 조작하는데 사용하였다. 전력으로는 18650 3.7V건전지 2개를 사용하였다.
Raspberry Pi		OS의 기능을 싱글보드 컴퓨터로서 파이카메라를 사용하여 주행 시에 입력되는 영상 데이터들을 PC(server)로 통신하는데 사용하였다. 외부전력을 공급받기 위해 휴대폰 보조배터리(2.1A)를 사용하였다.

<표 1> 사용된 부품들

4) Putty 설정

라즈베리파이는 하나의 OS로서 사용하기 위해서는 키보드나 마우스의 연결이 필요하다. 하지만 주행하는 자동차에 앞의 장치들을 부착할 수는 없기에 원격 제어가 필요하다. Raspberry Pi3는 기본적으로 와이파이 기능이 제공되기에 putty를 통하여 PC(Server)에서 Raspberry Pi에 연결하여 원격 제어를 통해 영상 스트림을 통신하기 위한 코드를 수행한다.

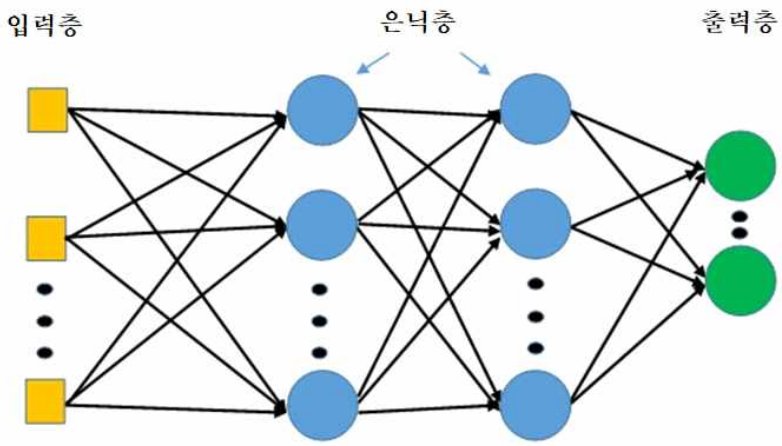
4.2 신경회로망 학습 모듈 개발

하드웨어 조립이 완료되면 이들을 구동할 프로그램 모듈들을 개발하여야 한다. 본 과제는 딥러닝에 기반한 자율주행자동차를 개발하기 때문에 먼저 자율주행에 사용할 신경망 모델을 설정하였다. 다음으로 설정한 신경망을 학습할 모듈을 개발하였다.

1) 딥러닝을 위한 신경망 모델의 생성

본 과제에서는 자율주행 학습을 위해 다층퍼셉트론을 사용한다. 그리고 다층프셉트론을 지원하는 OpenCV의 cv2.ml 라이브러리 함수를 활용하여 학습 모듈을 구현하였다. 그리고 구현한 학습 모듈을 이용하여 기계학습을 수행하였다.

본 과제에서 사용한 다층프셉트론(MLP, Multilayer Perceptron)의 구조는 다음 <그림 6>과 같다.



<그림 6> 다층퍼셉트론(MLP)

다층프셉트론은 하나의 입력층과 여러개의 은닉층(hidden layer) 및 하나의 출력층을 가지고 있다. 따라서 여러 층으로 구성되며 은닉층을 가진 구조이므로 심층

신경망 구조로 딥러닝으로 학습한다. 심층 신경망은 층마다 다른 층위의 특징이 학습된다. 낮은 층위의 특징은 단순하고 구체적이며 높은 층위의 특징은 더욱 복잡하고 추상적이다.

본 과제에서는 다층프셉트론을 구성하기 위해 OpenCV의 cv2,ml 라이브러리를 사용한다. OpenCV의 cv2,ml 라이브러리는 기계학습을 위한 클래스와 함수들의 집합, 특히 다층프셉트론(MLP)을 구현할 클래스와 함수를 제공한다. 다층프셉트론을 구성하는 함수는 다음과 같다.

```
cv2.ml.ANN_MLP.create(layerSizes[, activateFunc[, fparam1[, fparam2]]])
```

ANN_MLP.create 함수는 매개변수로 주어지는 모델을 갖는 다층프셉트론을 구성한다. 여기서, layerSizes는 정수 벡터로 주어지며 각 정수 값은 각 층의 뉴런 개수를 지정할 수 있다.

학습의 속도와 정확성을 중심으로 자율주행에 적합한 다층프셉트론 구조를 도출하기 위해 은닉층을 1개, 2개로 수정하며 검토하고, 뉴런의 개수도 다양하게 검토한 결과 본 과제에서 최종적으로 사용한 다층프셉트론은 입력층의 뉴런의 개수가 38,400개이며, 은닉층의 뉴런의 개수는 64개이며, 출력층의 개수는 4개로 구성되어 있다. 그리고 출력층의 각각은 자동차의 전진(forward), 왼쪽(left), 오른쪽(right) 정지(stop) 등에 해당한다.

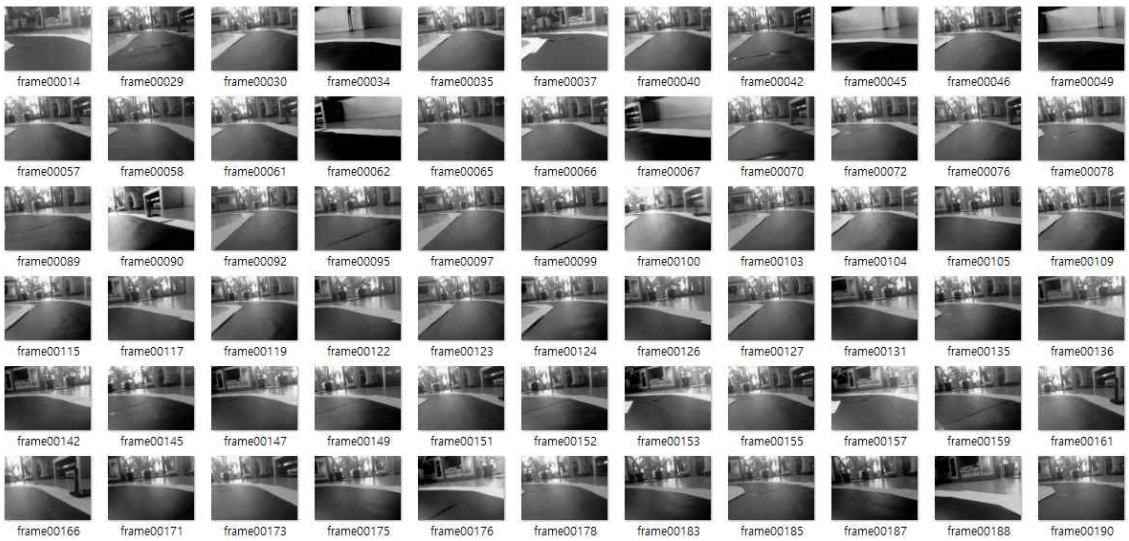
본 과제에서 MLP 모델구조를 사용한 것은 자율주행에서 설정한 출력의 개수가 적고 이미지로 입력되는 입력의 개수가 아주 많고 자율주행에 대한 간단한 제어동작으로 구성되어 있기 때문에 은닉층을 많이 사용할 필요가 없었기 때문이다. 은닉층을 많이 사용할 경우 학습하는데 오래 걸린 노력에 비해 주행 정확도는 향상되지 않았기 때문이기도 하다.

2) 기계학습 단계

인공신경망을 기계 학습하기 위해 파이카메라로 입력되는 샘플데이터를 수집하여야 한다. 라즈베리 파이가 파이카메라로부터 비디오 스트림을 입력하여 프레임들을 서버 PC에 전달하면 학습모듈에서는 수동으로 RC자동차의 움직임을 제어하여 훈련 데이터와 검사 데이터를 생성한다.

① 샘플 데이터 수집

데이터학습에 필요한 샘플 데이터들을 RC자동차를 주행하면서 영상 상황별 키보드를 사용하여 주행 방식을 저장하고 이를 토대로 샘플 데이터들을 파이썬 Numpy를 이용하여 npz파일로 저장한다. 저장된 비디오 프레임의 일부는 다음 <그림 7>과 같으며 이는 기계학습 시에 사용된 프레임을 보여준다.



<그림 7> 수집된 샘플 비디오 프레임

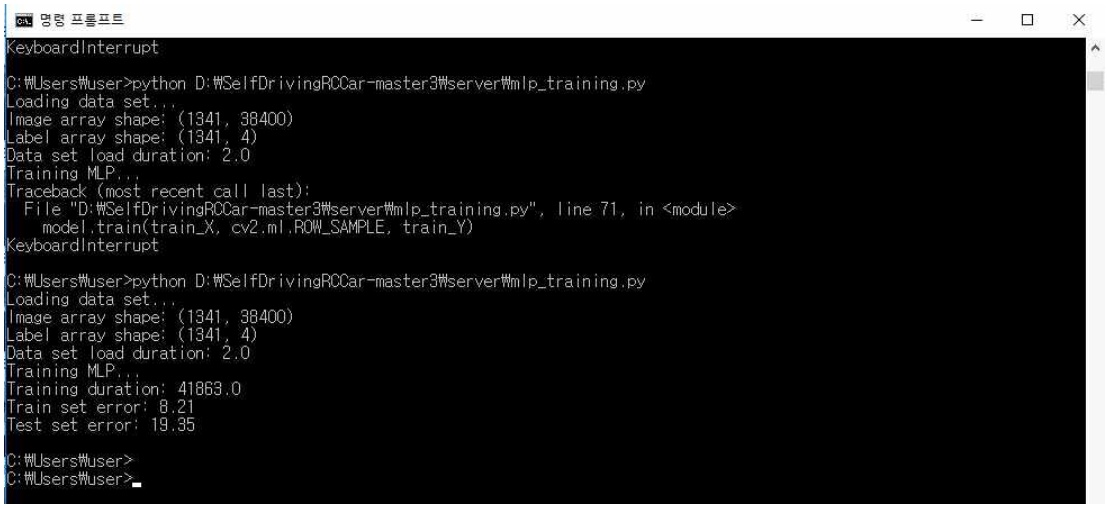
샘플데이터를 생성하는 구체적 방법은 라즈베리파이로부터 TCP/IP 통신으로 비디오 스트림을 받고 그 데이터에 키보드 제어를 활용하여 상황에 따른 주행 학습 데이터를 생성한다. 샘플데이터 npz에서 비디오 프레임은 학습시 입력값이 되며 키보드 제어값은 출력값이 된다.

② 기계학습

기계학습을 위한 데이터를 준비한 다음 기계학습을 수행한다. 학습은 지도학습(Supervised learning)으로 이루어지고 학습한 데이터의 에러 확률을 줄이기 위하여 역전파(BFS)알고리즘을 이용하였다. 그리고 학습을 위해 OpenCV의 cv2,ml 라이브러리에 있는 다음과 같은 ANN_MLP.train 함수를 사용하였다.

```
cv2.ml.ANN_MLP.train(inputs, outputs, sampleWeights[, sampleIdx[, params[, flags]]])
```

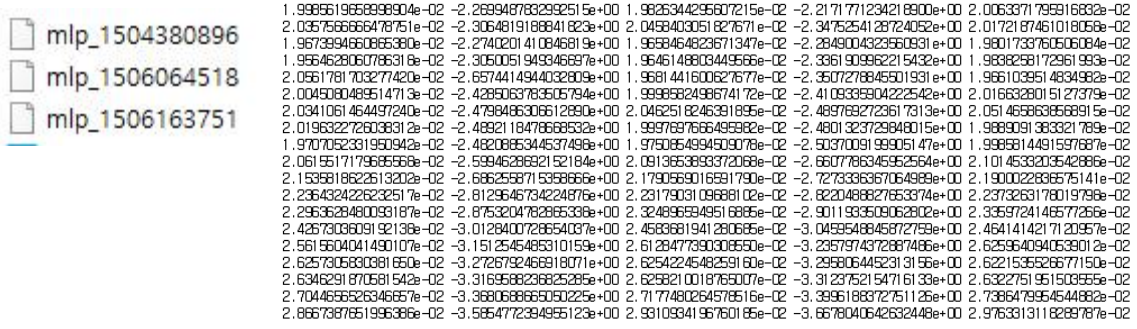
ANN_MLP.train 함수는 수집된 훈련 데이터와 검사 데이터를 이용하여 기계학습을 수행한다. 이 함수의 inputs 파라미터는 비디오 프레임, outputs는 키보드 제어값이 된다. 학습은 npz파일을 읽어서 진행하면서 500회 반복하거나 오차율이 0.001보다 낮으면 학습을 종료하도록 설정하였고 학습이 종료되면 입력에 따른 주행 학습결과를 xml 파일로 학습모델을 저장하도록 하였다.
다음 <그림 8>은 샘플데이터의 npz들을 이용하여 다층퍼셉트론에 기계학습한 결과를 보여주고 있다.



<그림 8> 기계학습 결과

학습 결과 훈련 오류가 8.21%로 비교적 높은 수치이나 RC주행자동차의 차선 이탈에는 영향을 미치지 않는 수준이었다.

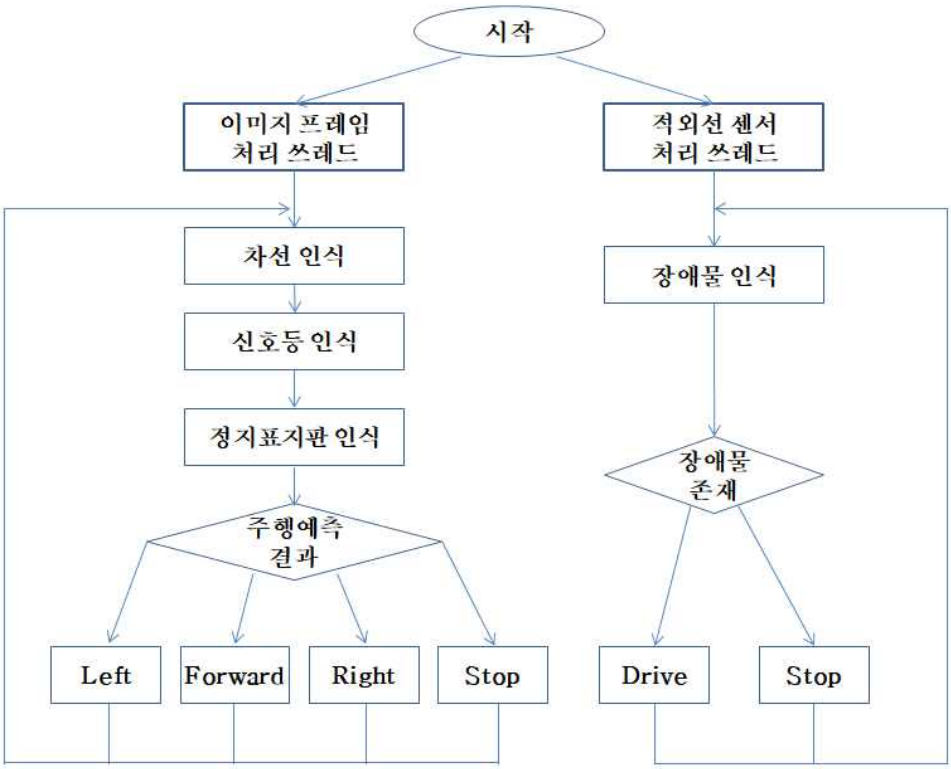
학습한 결과에 대한 데이터는 xml 파일을 생성하여 저장한다. 학습 후 생성된 xml 파일의 실례는 다음 <그림 9>와 같다.



<그림 9> 학습 결과 데이터

4.3 자율주행 프로그램 개발

자율주행을 위해 서버 PC에서 수행할 자율주행 프로그램의 흐름도는 다음 <그림 10>과 같다.



<그림 10> 자율주행 서버 프로그램의 흐름도

자율주행 프로그램의 기본 구조는 멀티쓰레드 구조를 갖는다. 하나의 쓰레드는 파이카메라로부터 들어오는 프레임을 처리하며, 다른 하나의 쓰레드는 적외선 센서로부터 오는 장애물의 존재를 처리하는 쓰레드이다.

서버는 주기적으로 입력되는 비디오 프레임에서 주행결과를 예측하여 RC자동차의 움직임을 컨트롤하고 또 영상뿐만이 아니라 장애물의 인식을 위해서 적외선 센서를 이용하여 장애물 감지하여 일정거리에 있으면 움직임을 멈추라는 신호를 보내주며 RC자동차를 컨트롤한다.

입력되는 비디오 프레임으로부터 주행을 예측하기 위해 OpenCV의 cv2,ml 라이브러리에 있는 ANN_MLP.predict 함수를 사용하였다.

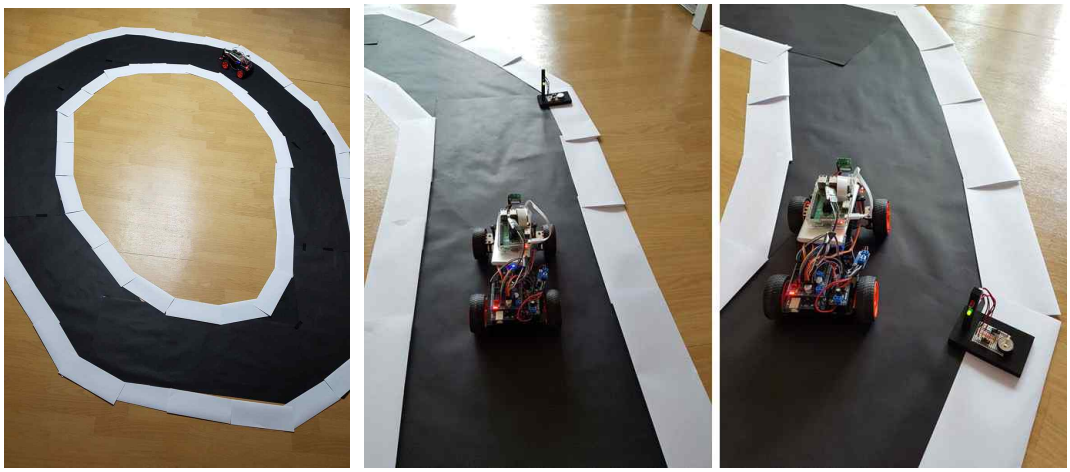
```
cv2.ml.ANN_MLP.predict(inputs[, outputs]) → retval, outputs
```

ANN_MLP.predict 함수는 라즈베리파이의 파이카메라로부터 입력되는 비디오 프레임을 이용하여 주행을 위한 출력 결과를 예측한다.

또한 적외선 센서는 RC자동차의 전방에 장애물이 존재할 경우 이를 탐지하여 RC자동차를 정지하게 하는 역할을 수행한다.

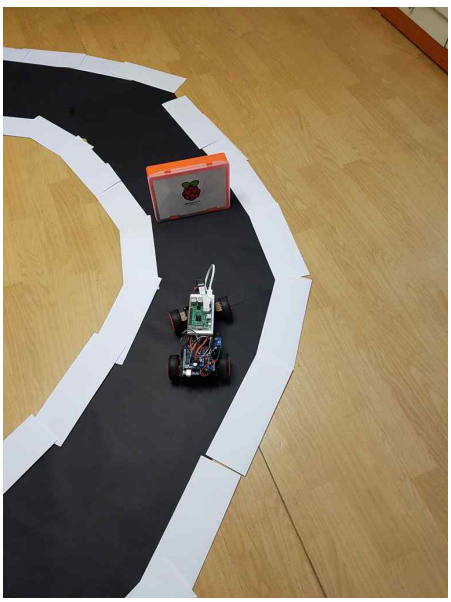
4.4 자율주행 실험 결과

자율주행은 파이카메라와 적외선 센서로부터 오는 정보를 이용하여 학습단계에서 학습한 mlp.xml 학습 데이터들로 실시간으로 라즈베리 파이에서 Socket 통신되는 영상입력 데이터에 따라 주행을 예측하여 주행정보를 아두이노에 블루투스 통신을 이용하여 보냄으로써 이루어진다. 그리고 이때 적외선 센서에서 장애물이 감지될 경우에는 통신에 관계없이 모터를 정지시킴으로서 충돌을 방지한다. 다음 <그림 11>은 자율주행을 위한 차선을 따라 RC자동차가 자율주행하는 모습을 보여주고 있다.



<그림 11> 실험한 차선을 따라 자율주행 중인 RC자동차

다음 <그림 12>는 자율주행차가 장애물을 만났을 때 정지한 모습을 보여주고 있다.



<그림 12> 장애물을 만나 정지한 RC자동차

자율주행 결과는 차선인식은 3바퀴 회전에 1회의 실패율을 보였으면 장애물 인식에 대해 오류는 없었다.

5. 결론

이 장에서는 본 과제를 수행한 결과에 대한 결론을 내리고 향후 계획에 대해 간략히 기술하고 끝으로 역할분담을 보여준다.

5.1 과제 결과

본 과제에서는 OpenCV를 이용한 딥러닝 기반 자율주행 자동차를 개발하였다. 이를 위해 현실 세계가 아닌 인위적으로 만든 실험 공간에서 딥러닝을 통한 자율주행 기능을 구현하였다. 개발한 기능은 차선인식, 신호등 인식, 장애물 인식으로 제한하였다. 그리고 차선인식의 실패율은 3바퀴 회전에 1회의 실패율을 보였으며 장애물 인식의 오류는 없어 실험은 성공적으로 평가된다. 따라서 본 과제는 다층 퍼셉트론을 이용한 기계학습에 기반하고 있으며, 제한된 환경 속에서 목표한 대로 비교적 잘 작동하였다.

5.2 향후 과제

향 후 본 과제에서는 포함되지 않은 GPS 센서를 통한 위치 정보 파악을 통한 목표 지점 설정과 주행 위치 파악 기능을 구현하면 보다 완전한 자율주행자동차를 구현할 수 있을 것이다.

5.3 역할 분담

학번	이름	구성원별 역할
201424459	박재현	신경회로망을 이용한 학습구현 학습에 따른 주행 판별 모터 및 센서 조작
201124455	박지훈	장애물 판별 영상처리 , 동작 제어 블루투스, TCP/IP 통신 아두이노 모델 제작 학습 샘플데이터 구현