자율주행 자동차 제작 보고서

IoT 시스템 설계 및 분석

정보융합학부

2019204076 장지원

2019204082 김지훈

2019204018 김상수

2018610005 이관준

목차

1. 알고리즘

1-1. 학습 데이터 수집 알고리즘

1-2. 데이터 학습 알고리즘

2. 데이터 수집

2-1. 카메라 위치 조정

2-2. 데이터 수집 방법론

2-3. 방향전환 방법론

2-4. 데이터 수집에 활용한 코스

3. 이미지 처리 방법

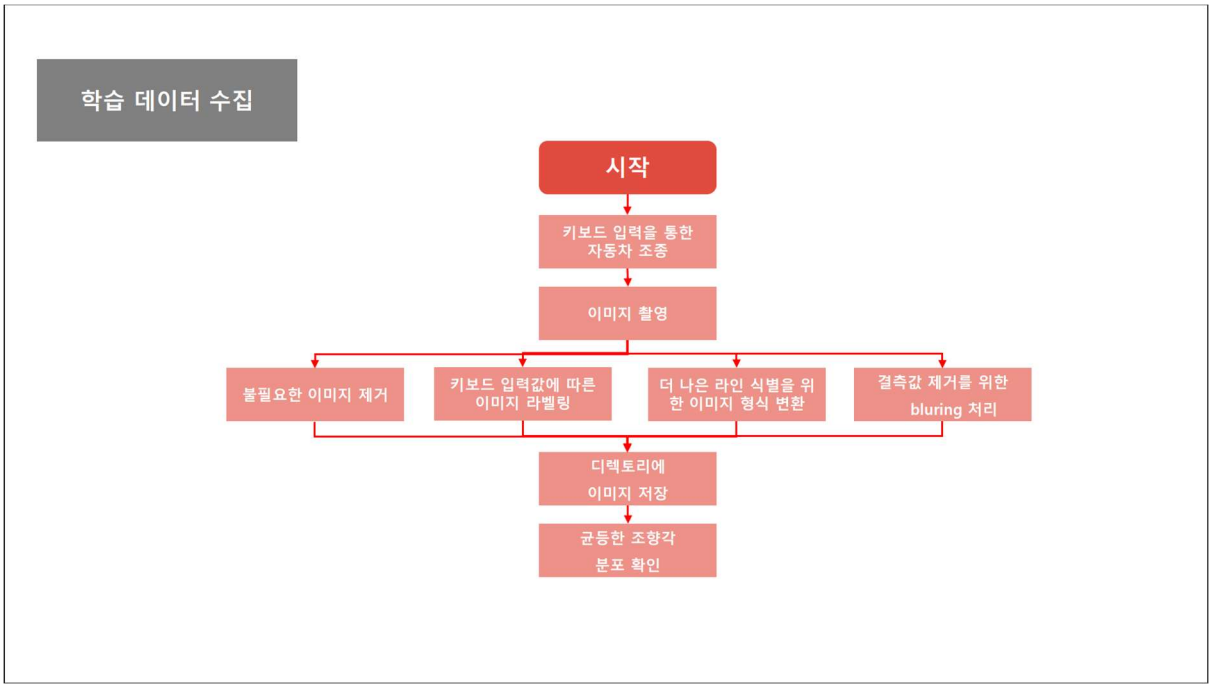
4. 시행착오

5. 조원의 역할 및 기여도

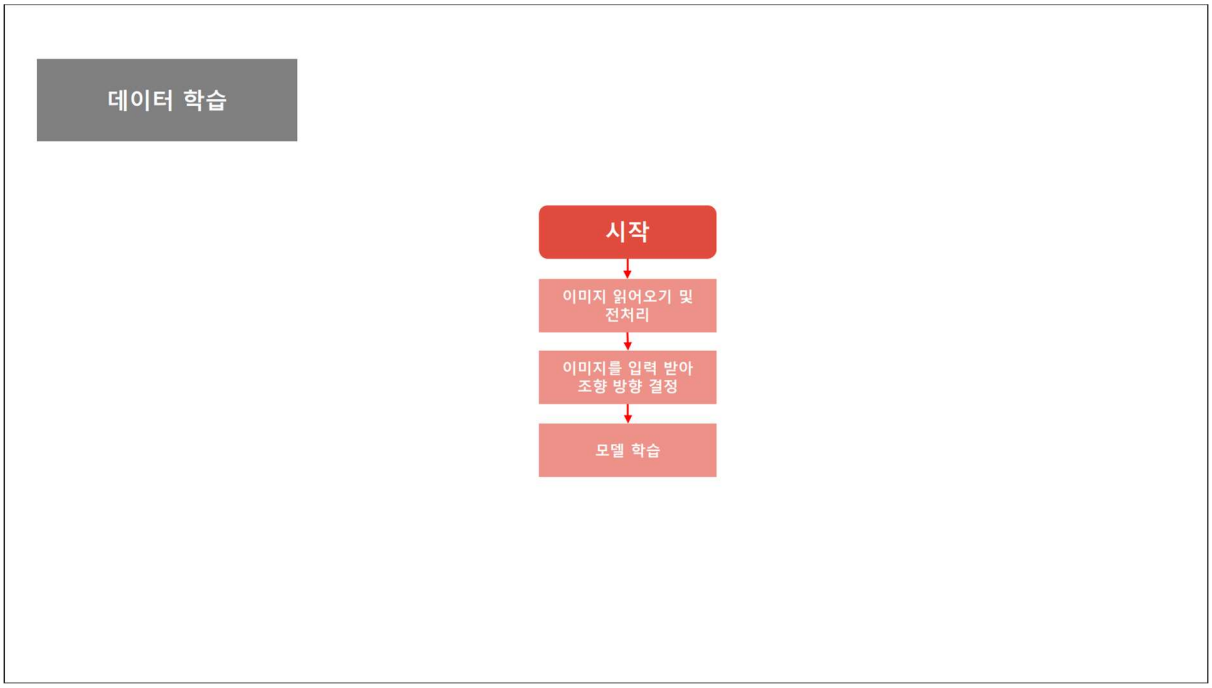
6. 주행 영상 및 활동 사진

1. 알고리즘

1-1. 학습 데이터 수집 알고리즘



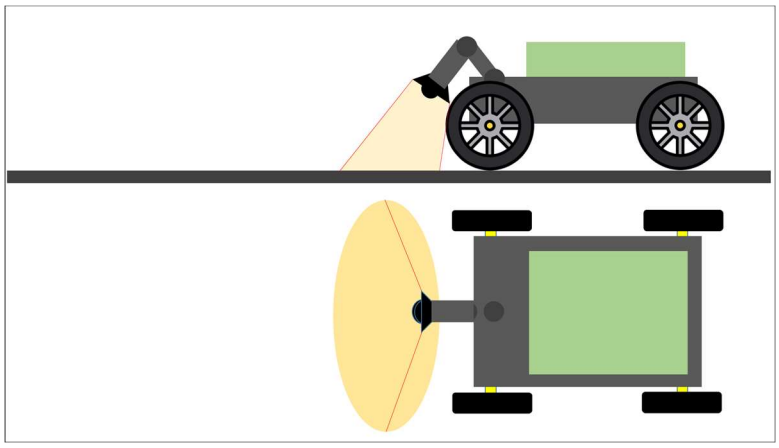
1-2. 데이터 학습 알고리즘



2. 데이터 수집

2-1. 카메라 위치 조정

① 카메라가 아래를 향함



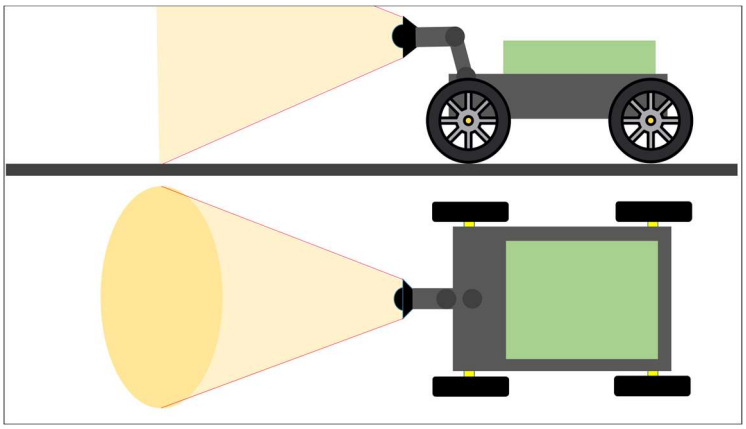
장점

- 주행결과, 직진 구간에 대한 정확도와 안정성이 높았다.

단점

- 장애물 인식에 어려움을 겪었다.
- 학습하는 라인의 길이가 상대적으로 짧아지게 되어 복잡한 트랙에 대한 주행의 정확도와 안정성이 낮음을 확인했다.

② 카메라가 정면을 향함



장점

- 장애물 인식에 강점을 가졌다.

단점

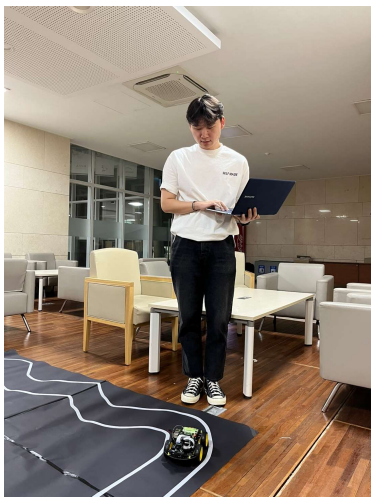
- 자동차 바로 앞 라인이 어떻게 형성되어 있는지 알지 못한 채로 주행하기 때문에 전체적인 주행의 정확도와 안정성 모두 낮았다.

결론

위와 같은 일련의 시행착오를 통해 적절한 카메라 각도가 학습 데이터를 쌓는 데에 중요할 것이라 판단하였
트랙을 주행해보았고, 가장 높은 정확도와 안정성을 보이는 카메라 각도를 채택하여 모델을 만들었다.

2-2. 데이터 수집 방법론

데이터 수집은 2가지 방법으로 나누어 수행해 보았다. 첫 번째로 사람의 눈을 통해 직접 트랙을 보면서 주행하며 이미지 데이터를 수집하였고, 두 번째로는 자율주행 자동차의 카메라를 통해 보여지는 트랙의 이미지 데이터를 수집하였다. 여기서 우리는 위의 2가지 데이터 수집 방법론 중 더 나은은 결과를 보이는 방법을 채택하여 사용하기로 하였다.




각각 사람의 눈을 통해 직접 트랙을 보면서 주행하는 모습과 자동차의 카메라로 얻어지는 영상을 통해 주행하는 모습이다.

① 직접 트랙을 보면서 주행하는 방법

첫 번째 데이터 수집 방법에 대한 가설을 다음과 같이 정의 하였다.

‘사람의 눈을 통해 직접 트랙을 보면서 주행하며 데이터를 모으는 방법은 데이터의 질이 상대적으로 낮다.’

해당 데이터 학습 방법은 데이터 수집을 위한 주행이 더 쉽다는 장점이 있어 적은 시간 안에 데이터의 볼륨을 많이 확보할 수 있다. 하지만 데이터 학습은 카메라를 통해 얻어지는 이미지로 이루어지기 때문에 운전자의 눈을 통해 트랙을 보면서 주행하는 방법은 데이터의 질이 상대적으로 낮을 것이라고 생각하였다. 이에 대응하여 볼륨이 큰 데이터를 사용하여 모델을 구축하면 사용 가능한 모델이 구축될 수 있을 것이라 판단했다.

모델 구축 결과)

- 전체적인 주행 정확도와 안정성 모두 기준 미달이었다.
- 데이터의 질이 낮은 단점을 볼륨이 큰 데이터가 보완해줄 수 있을 것이라 생각했으나 전체적으로 실패한 모델이 구축되었다.

② 자동차의 카메라로 얻어지는 영상을 통해 주행

두 번째 데이터 수집 방법에 대한 가설을 다음과 같이 정의 하였다.

‘자동차의 카메라로 얻어지는 영상을 통해 주행하며 데이터를 모으는 방법은 데이터의 질이 상

대적으로 높다.'

해당 데이터 학습 방법은 데이터 수집을 위한 주행이 더 어렵다는 단점이 있어 데이터 확보에 많은 시간이 소요된다. 그래서 데이터의 볼륨이 작을 수밖에 없는 단점을 높은 데이터의 질로 보완하여 모델을 구축하기로 했다.

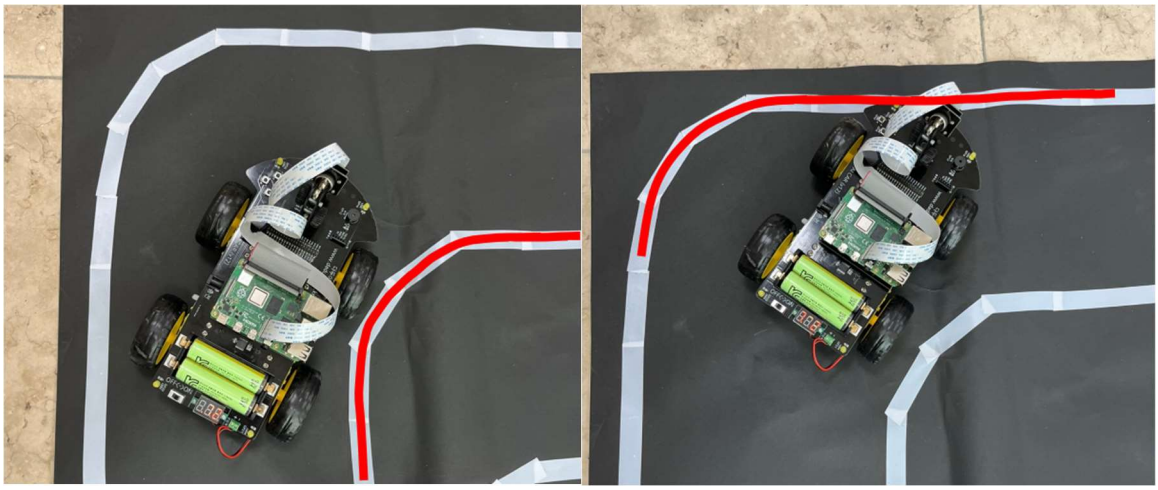
모델 구축 결과)

- 전체적인 주행 정확도와 안정성이 준수함을 확인함
- 데이터의 볼륨이 작은 단점을 높은 데이터의 질로 보완하여 어느정도 성능을 보이는 모델을 구축함

결론

위와 같은 일련의 시행착오를 통해 자동차의 카메라로 얻어지는 영상을 통해 주행하며 학습 데이터를 수집하

2-3. 방향전환 방법론



각각 안쪽 라인을 따라 주행하는 모습과 바깥쪽 라인을 따라 주행하는 모습이다. 우리 조는 주행의 안정성을 라인을 침범하지 않고 트랙의 중앙을 얼마나 잘 유지하는가로 결정했다.

① 안쪽 라인을 기준으로 방향전환

사전에 자동차를 움직일 때 안쪽 라인을 기준으로 방향전환을 수행하면 트랙의 중앙으로 차체의 중심이 유지됨을 확인했다. 그러므로 처음에는 안쪽 라인을 기준으로 방향전환을 한 데이터로 모델을 구축했다. 해당 모델은 차체 회전 반경이 작은 트랙에서만 정상적으로 주행하는 모습을 보였다.

② 바깥쪽 라인을 기준으로 방향전환

바깥쪽 라인을 기준으로 방향전환을 하면 안쪽 라인을 기준으로 방향전환을 수행할 때보다 차체 회전 반경이 커야만 하는 트랙에서 선 밖으로 나가는 일이 적어졌다.

결론

차체의 중심이 트랙의 중앙에 위치하기 위해서는 안쪽 라인을 기준으로 방향 전환을 해야 하나, 해당 방법은
확인할 수 있었다. 따라서 차체 회전 반경이 큰 트랙에서도 정상적으로 주행하는 모델을 만들기 위해 바깥쪽

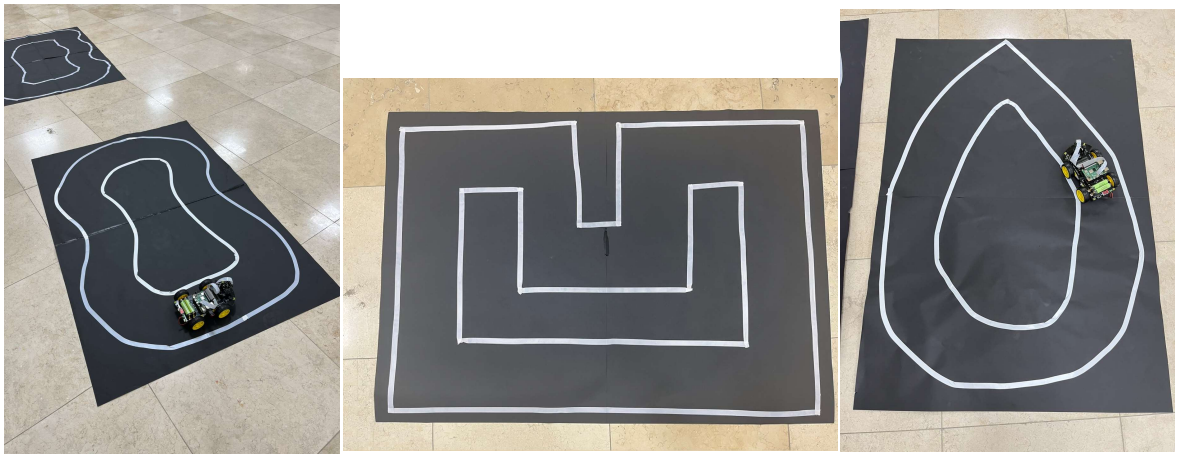
2-4. 데이터 수집에 활용한 코스



꼬불꼬불 골목길 - 트랙의 폭이 좁음

D자 트랙 - 차체의 회전 반경이 큼

F1 트랙 - 180도 회전 코스가 있음



8자 코스 - 차체의 회전이 완만함

직각 B자 코스 - 직각 회전이 많음

물방울 코스 - 회전반경이 큼

3. 이미지 처리 방법

우리 조는 이미지 처리를 여러 방법을 사용해본 후 가장 나은 성능을 보이는 처리 방법을 사용하기로 결정했다. 사용해 본 이미지 처리 방법들은 3가지가 있으며 다음과 같다.

① 이미지 YUV + Blur 처리리 후 이미지

차선 인식은 되나 햇빛에 의한 노이즈가 있어 햇빛이 조금이라도 있으면 차선의 인식이 어려움을 확인했다.

② threshold 130 적용 후 이미지

차선 인식이 잘 되고 적당한 노이즈가 존재하는 임계값이 130임을 확인했다. 하지만 여전히 노이즈가 존재하여 차선 인식에 어려움이 있어 개선의 필요성이 있었다.

③ 이미지에서 색을 제거하고, 이진화 하여 흰색과 검은색으로 구분

모폴로지 연산을 적용하고 흰색과 검은색으로 구분한다.

결론

방법 ③을 통한 이미지 전처리 방식이 가장 노이즈에 강건한 모습을 보여줌을 확인했다. 그러므로 해당 방

4. 시행착오

문제 1) 딥러닝을 통한 직각코스 주행의 어려움

직각코스는 자율주행 자동차 입장에서 완만한 코스에 비해 회전 방향의 판단 기준이 모호하여 주행에 어려움이 있다. 이 문제를 직각코스에 대한 더 많은 데이터 학습을 통해 해결하고자 하였으나 해당 방법을 통한 성능 개선은 실패하였다.

카메라의 렌즈를 통해 직각 코스를 바라보게 되면 주어진 카메라 렌즈의 한정된 화각으로 인해 차선을 아예 보지 못하는 경우가 생긴다. 이에 대한 해결책으로 본래 직선으로 주행해야 하는 조향각에서 직선이 아닌 오른쪽으로 약간의 회전을 주며 주행하는 방법을 고안했다. 해당 방법을 통해 카메라는 항상 오른쪽 차선을 인식할 수 있게 되어 직각 코스에 대해 더욱 강건하게 동작하였다.

```
if self.steering_angle >= 85 and self.steering_angle <= 95:
    print("go")
    self.motor_go(90,50)
```

수정된 부분: (90,90) -> (90,50)

문제 2) 무거운 모델로 인해 지연 시간 발생

프로젝트를 진행하면서 무거운 모델로 인해 지연 시간이 발생하여 조향각을 산출하는 시간에 딜레이가 발생했다. 이는 곧 차체가 라인을 벗어나는 결과를 초래했다. 우리는 이 문제를 해결하기 위해 TFLite라는 오픈소스 딥러닝 프레임 워크를 사용하였다.

TFLite는 모바일 및 임베디드 기기의 제한된 자원과 계산 능력을 고려하여 최적화된 딥러닝 모델 실행 환경을 제공한다. 또한 모델 크기를 축소하고, 메모리 사용량과 연산량을 최소화하여 효율적인 실행이 가능하도록 하는 TFLite는 하드웨어 가속을 지원하여 딥러닝 모델을 효율적으로 실행할 수 있어 CPU, GPU, DSP 및 네이티브 하드웨어 가속기와 같은 다양한 플랫폼에서 최적화된 실행을 지원한다.

해당 딥러닝 프레임 워크를 사용하기 전에는 조향각 계산을 5번 수행하는 시간이 평균적으로 1.6초가 걸리는 것을 확인했다. 이후 TFLite를 적용하고 난 뒤 수행 시간이 평균적으로 5배 가량 줄어들었고, 이는 차체가 라인을 벗어나지 않고 더 안정적으로 주행하도록 만들었다.

문제 3) 물체 탐지

우리는 물체 탐지에 YOLO와 PyTorch, 그리고 OpenCV DNN과 MobileNet-SSD를 사용한 물체 탐지 아키텍처를 사용해보았다. YOLO는 실시간 객체 탐지에 특화되어 있으며, 전체 이미지 컨텍스트를 활용하여 다양한 객체를 탐지할 수 있다. PyTorch는 동적 그래프와 사용자 친화적인 API를 제공하여 모델 개발과 실험을 편리하게 수행할 수 있으며, 활발한 커뮤니티와 생태계를 통해 다양한 지원과 자원을 활용할 수 있다. 하지만 해당 두 아키텍처는 좋은 성능을 내지 못해 가장 나은 성능을 보인 OpenCV DNN과 MobileNet-SSD를 사용한 물체 탐지를 사용하였다.

성능강화를 위한 시도) hidden layer를 한 층 더 쌓는 심층 신경망을 추가함

심층 신경망은 더 깊은 구조를 가지므로 더 복잡하고 추상적인 패턴을 학습할 수 있다. 우리는 이에 착안하여 심층 신경망을 추가하면 더 나은 성능을 보여줄 수 있을 거라 생각했다. 하지만 수정된 모델은 무게가 매우 커 라즈베리파이 실행 환경에서 수행할 수 없었다. 이 문제를 해결하기 위해 우리는 해당 모델을 경량화하여 수행해보기도 하였으나 원본 모델보다 더 떨어진 성능을 보여 신경망의 추가를 통한 성능강화를 위한 시도는 실패로 결론지었다.

5. 조원의 역할 및 기여도

이름	역할	기여도
이관준	학습 모델 구축, 모델 평가	27%
김상수	학습 모델 구축, 모델 평가	24.4%
김지훈	학습 모델 구축, 보고서, 모델 평가	24.4%
장지원	데이터 학습, 코스 제작, 보고서, 모델 평가	24.4%

6. 주행 영상 및 활동사진

곡선 코스 주행 영상: https://youtu.be/kocXg6vgz3s

직각 코스 주행 영상: https://youtu.be/8xnh0wM1N20