



HuStar 경북 AI·SW 아카데미 3 기 - 2 조

Smart BallBot

HuStar 3 기 2 조

박효정, 이진주, 김수영, 배수영, 여동훈

요약문

이번 프로젝트에서 제작한 Smart BallBot 은 이미지 프로세싱을 통한 객체 인식과 LiDAR 를 이용한 알고리즘으로 계획된 자율주행 기술을 결합한 테니스공 수거 로봇이다. 운동 경기 중 공을 수거하는 볼퍼슨이 종종 위험한 상황에 처하는 것을 보고 프로젝트의 아이디어를 얻었다.

객체 인식을 위하여 직접 1만장이 넘는 이미지 데이터셋을 수집하고 제작하여 YOLO 로 머신러닝을 진행하여 객체 인식 모델을 만들었다. YOLO 는 다른 모델에 비하여 실시간성과 정확도를 따졌을 때 합리적인 성능을 보여주어서 채택하였다.

자율주행은 시뮬레이션 환경에서 강화학습을 통하여 능동적으로 장애물을 회피하고 목적지에 도달하는 모델을 제작하려고 하였으나 시뮬레이션과 실제 환경 사이의 간극이 있어서 알고리즘을 통한 자율주행 모델로 선회하게 되었다.

로봇은 turtlebot3 를 기반으로 하여 자체적인 프레임과 컨트롤러 보드를 교체하고 탑재하여 이미지 프로세싱을 위한 성능을 높이고 공을 수거하기 위한 하드웨어를 제작하였다. 이후 모터와 보드 및 회로를 보호하기 위하여 외장을 제작하여 장착하였다.

프로젝트를 진행하면서 이미지 프로세싱과 머신 러닝에 대한 전반적인 이해와 학습 방법을 배울 수 있었다. 또한 하드웨어와 센서에서 입력되는 데이터의 형태와 전처리를 경험하면서 추후의 개선 방향을 모색할 수 있었다. Flow Chart 를 통하여 로봇의 동작을 설계하며 프로그램 설계에 대하여 알고리즘의 중요성을 이해할 수 있었다.

Contents

1. Introduction	01
1.1 프로젝트 소개	01
1.2 주제 선정 배경	01
1) 볼퍼슨의 부상	01
2) 코로나 장기화로 인한 개인 스포츠 시장 활성화	01
3) 기존 제품과의 비교	02
2. Project	04
2.1 전체 구조	04
1) Node Graph	04
2) Flow Chart	05
2.2 객체 인식	05
1) YOLO 란	06
a) YOLOv4	07
b) YOLOv5	08
2) YOLO 선정 이유	09
a) 실시간성	09
b) 정확도	10
c) 선행 연구 분석 및 결론	10
3) 학습과정	11
a) 1차 데이터셋 확보	11
b) 2차 데이터셋 확보	11
c) 3차 데이터셋 확보	12
d) 학습 결과 비교	12
2.3 자율주행	13
1) 자율주행의 트렌드	13
2) 강화학습	13

a) 강화학습이란	14
b) Q-learning 과 DQN	14
c) 강화학습 과정	15
d) 강화학습 과정 중 발생한 문제점	15
3) 구현 결과	16
a) 객체 추적	16
b) 장애물 회피	17
c) 객체 탐색 주행	18
2.4 하드웨어	20
1) 주요 부품 리스트	21
2) 하드웨어 구상도 및 규격	22
3) 제작 과정	23
2.5 결과	24
3. 기타	25
3.1 기대 효과	25
1) 볼퍼슨의 부상 예방	25
2) 개인 스포츠 활동의 질 향상	25
3) 다양한 시장에서의 성장 가능성	25
3.2 한계 및 발전 방향	26
1) 단일 개체의 경로 계획	26
2) 강화학습의 실적용	26
3) 센서 노이즈 및 노면 특성	26
3.3 프로젝트 진행 일정	27
3.4 개발 환경 소개	28
4. 팀원 소개 및 소감	29
5. 참고 문헌	30

Table Contents

[표 1] 단일 단계 검출 방법과 두 단계 검출 방법 비교	09
[표 2] 선정 모델 학습 결과 비교	12
[표 3] 객체 탐색 주행 알고리즘 별 수거 완료 시간	20
[표 4] 로봇에 사용된 보드 및 센서	21
[표 5] 프로젝트 진행 일정	27

Image Contents

[그림 1] 기존 테니스공 수거 제품 - 롤러형(좌), 카트형(우)	02
[그림 2] 기존 테니스공 수거기 - Tennibot	03
[그림 3] Node Grpah	04
[그림 4] Flow Chart	05
[그림 5] YOLO 객체 검출 과정	06
[그림 6] YOLO 아키텍처	07
[그림 7] YOLOv4 아키텍처	07
[그림 8] 객체 인식 모델 성능 비교	08
[그림 9] YOLOv5 아키텍처	08
[그림 10] YOLOv5 모델 비교	09
[그림 11] YOLO, SSD 및 다수 모델 성능 비교	10
[그림 12] 1차 데이터 수집 예시	11
[그림 13] YOLOv5s 모델 - 실시간 객체 인식 예시	12
[그림 14] 자율주행 트랜드 기술	13
[그림 15] 탐지된 객체의 예시	16
[그림 16] 장애물 회피 Flow Chart	17
[그림 17] 제자리 360도 회전	18
[그림 18] 직진 및 반사각 방향 회전 주행	19
[그림 19] 나선형 주행	19
[그림 20] 실제 실험 환경	20
[그림 21] 내부 구상도	22
[그림 22] 외부 구상도	23
[그림 23] 자율주행 테스트(좌) 및 피칭머신 테스트(우)	23
[그림 24] 구현 결과 - 내부(좌), 외부(우)	24
[그림 25] 사용 기술 목록	28
[그림 26] 팀원 소개	29

1. Introduction

1.1 프로젝트 소개

Smart BallBot 은 테니스장에서 바닥에 떨어진 테니스공을 주우며 ‘볼퍼슨’ 역할의 일부를 대신 수행하는 로봇이다. 해당 프로젝트에서 로봇을 제작하기 위하여 이용한 기술로는 크게 컴퓨터 비전과 자율주행이 있다. 컴퓨터 비전을 이용하여 테니스공을 객체로 인식하고 Depth Camera 로 인식된 공과의 거리를 측정하여 공을 확보하도록 로봇을 제어하는 파트와 LiDAR 로 주위 물체와 거리를 측정하여 장애물을 회피하는 자율주행 파트로 나누어 진행하게 되었다.

1.2 주제 선정 배경

1) 볼퍼슨의 부상

스포츠 경기가 진행되는 중에 경기의 원활한 진행을 위하여 선수를 케어하고 경기 중 경기장 외곽으로 튕겨나가는 공들을 줍는 볼퍼슨을 쉽게 볼 수 있다. 테니스장에서는 어린 볼퍼슨, 볼키즈들을 많이 볼 수 있는데, 일각에는 이것이 아동인권 침해라는 우려가 있다. 이는 종종 볼퍼슨이 선수의 공에 맞는 경우가 발생하기 때문이다.

테니스 경기에서 남성 선수의 평균 서브 속도는 200km 를 웃돌고 있으며 서브의 운동량을 전해받은 테니스공은 유리 3겹을 깨뜨릴 수 있는 위력을 가지고 있다. 가볍고 탄성 있는 테니스공임을 감안하더라도 충분히 사람이 맞을 경우 위험한 상황이 발생할 수 있다. 이번 프로젝트에서는 이러한 문제점을 해결하기 위하여 Smart BallBot 을 제작하게 되었다.

2) 코로나 장기화로 인한 개인 스포츠 시장 활성화

한편, 2021년 현재, 코로나 19 사태가 장기화되면서 소수의 인원들과 안전하게 스트레스를 해소할 수 있는 취미생활에 대한 관심이 높아지고 있고 그 중 골프, 테니스 등의 스포츠가 각광받고 있다.

SSG닷컴에 따르면 코로나 19 사태 이전인 2019년 동기 대비 테니스와 스쿼시 용품 매출이 75% 증가했으며 기타 관련 제품들의 매출도 크게 늘었다.¹ 또한 실내 테니스 연습장의 고객 이용률이 증가하였고, 회원의 70%가 2030세대인 것으로 나타났다.² 이러한 변화에 대응하여 테니스 관련한 편의 로봇을 제작한다면 개인 생활 운동의 질 향상에도 도움이 될 수 있을 것으로 기대된다.

3) 기존 제품과의 비교

테니스공 수거 제품에는 다음과 같이 롤러형, 카트형 등의 제품이 있다. 하지만 직접 손으로 끌면서 수거해야하고 부피가 크고 보관이 어렵다는 단점이 있다.



[그림 1] 기존 테니스공 수거 제품 - 롤러형(좌), 카트형(우)

이를 보완한 제품으로는 대표적으로 ‘Tennibot’ 이 있다. 테니봇은 별도의 스테이션을

¹ 매일경제. 2021.07.12. '3개월 대기는 기본, 인기제품 구하기도 어렵다', 골프? 2030은 테니스장 간다

² 이뉴스투데이. 2021.08.24. "비대면 레슨부터 1인 플레이까지" 스크린 테니스, 2030세대 여가생활로 '관심'

통해 공의 좌표 값을 생성하고 좌표를 기준으로 주행 경로를 계획하여 주행한다.



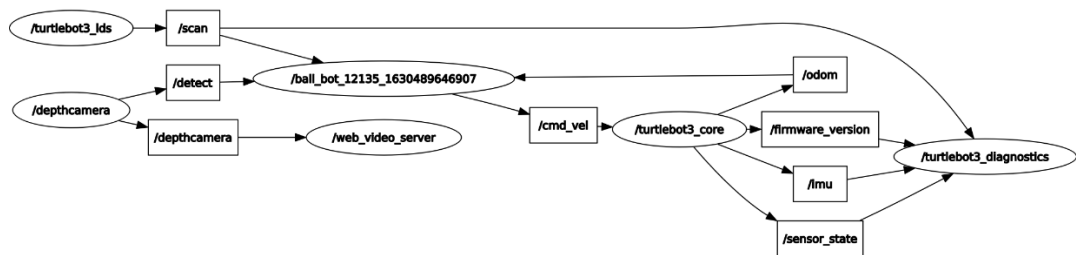
[그림 2] 기존 테니스공 수거기 - Tennibot

Tennibot 역시 부가적으로 스테이션을 휴대하고 설치하므로 다소 번거롭다는 점과 총 무게가 11kg로 무겁다는 단점이 있다. 따라서 이번 프로젝트는 'Tennibot' 보다 가볍고 별도의 카메라를 통하여 맵을 미리 생성하는 것이 아니라 필드 위에서 주행하는 로봇이 객체를 인식하도록 하여 휴대성을 개선하는데 초점을 두었다.

2. Project

2.1 전체 구조

1) Node Graph



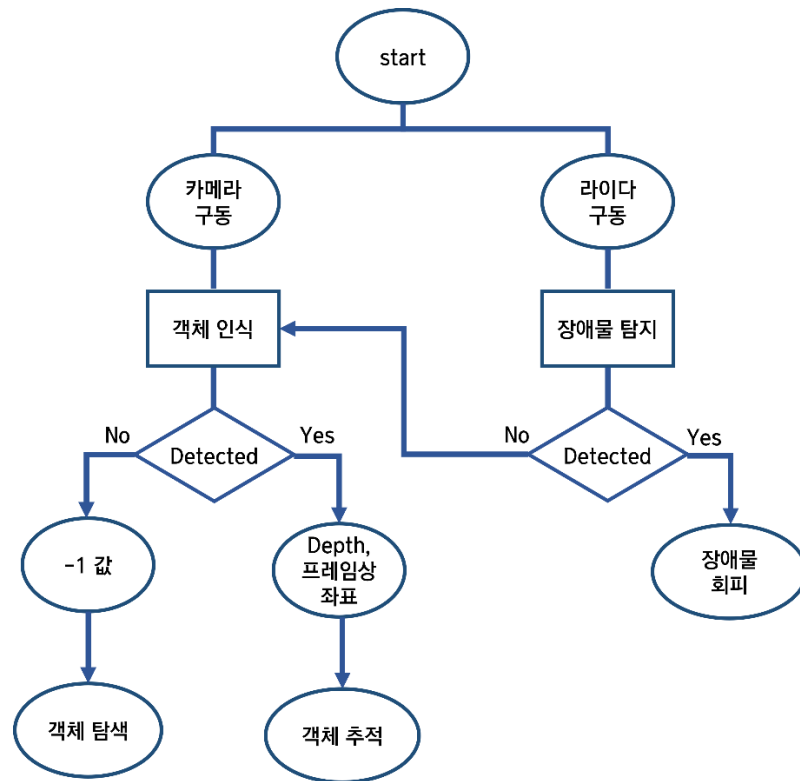
[그림 3] Node Graph

이번 프로젝트는 ROS(Robot Operating System) 이라 불리는 로봇 운영체제를 활용하였다. ROS 의 특성 중 하위 디바이스 제어, 프로세스간 메시지 패싱 등을 활용하기 위함이다. 또한 ROS 가 지원하는 자체 라이브러리는 다양한 프로그래밍 언어로 이루어진 노드와 메시지 형태로 데이터를 주고 받을 수 있게 구현되어 있어 로봇 소프트웨어의 설계에 유용했다.

로봇은 ROS 에서 다섯개의 노드를 실행하도록 구성하였다. ROS 운영체제를 위한 노드, 로봇의 상태를 받고 모터를 제어하는 노드, depth camera 에서 받은 이미지 데이터를 처리하고 필요한 값을 반환하는 노드, depth camera 의 데이터를 관찰하는 video_server 노드, 그리고 앞서 구동된 노드들에서 데이터를 받고 자율주행의 상태를 결정하는 노드로 구성되어 각자의 노드가 메시지를 통해 유기적으로 소통하며 로봇을 제어하도록 설계하였다.

turtlebot3 는 ROS 내에서 자체적으로 제공하는 자료형을 통하여 RiDAR 센서 값(scan), 위치 및 벡터 정보(odom) 등을 쉽게 얻을 수 있는 모듈을 제공하고 있다. 메시지 형태로 들어오는 정보를 해석하여 로봇의 선속도와 각속도(cmd_vel) 를 다시 메시지로 전달하여 비교적 쉽게 로봇을 제어할 수 있었다.

2) Flow Chart



[그림 4] Flow Chart

로봇의 자율주행 상태를 결정하는 노드는 [그림 4]와 같은 알고리즘으로 동작한다. LiDAR를 통한 장애물 탐지로 장애물을 회피하는 동작은 객체 인식으로 공을 추적하는 동작보다 우선하여 동작하도록 설계하였다.

장애물이 탐지되지 않는다면 객체 인식을 통하여 공의 위치 좌표를 받아서 추적하거나, 아무것도 탐지 되지 않았을 때 공을 탐색하는 동작을 하도록 설계하였다.

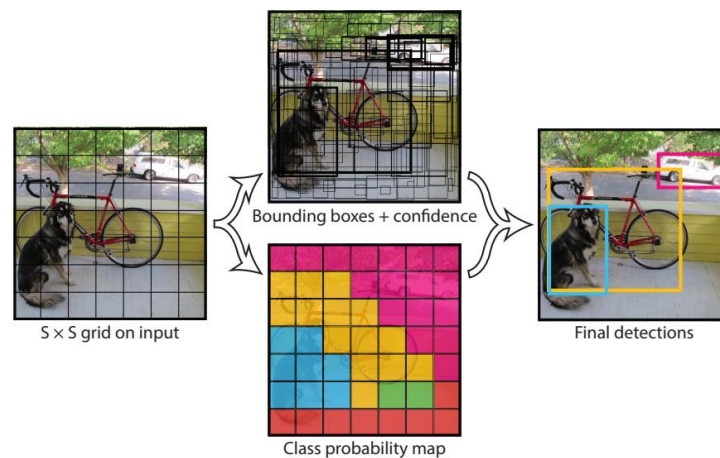
2.2 객체 인식

객체 인식은 이미지 또는 비디오에서 객체를 식별하는 컴퓨터 비전 기술이다. 자율주행을 위하여 객체를 추적할 수 있는 객체 인식이 필요하다. 이번 프로젝트에서는 테니스공을 인식하는 객체 인식 모델을 구성하는데 YOLO를 사용하였다.

1) YOLO 란

YOLO 란 실시간 객체 탐지를 지원하는 대표적인 모델로 이미지를 한번 보는 것만으로 Object의 종류와 위치를 추측하는 딥러닝 기반의 물체인식 알고리즘이다. 이미지의 픽셀로부터 bounding box 의 위치 좌표와 클래스 확률(class probabilities)을 구하기까지의 일련을 절차를 하나의 회귀 문제로 재정의 했기 때문에 이미지 내에 어떤 물체가 있고, 그 물체가 어디에 있는지를 하나의 파이프라인으로 빠르게 도출해낼 수 있다.

객체 검출 과정은 [그림 6] 과 같다. 예측하고자 하는 이미지를 SxS Grid cells 로 나누고 각 cell 마다 하나의 객체를 예측하며, 미리 설정된 개수의 boundary box 를 통해 객체의 위치와 크기를 파악한다.



[그림 5] YOLO 객체 검출 과정

YOLO 모델의 전체 구조는 총 24개의 컨볼루션 계층(convolutional layers) 과 2개의 전결합 계층(fully connected layers) 으로 구성되어 있다.

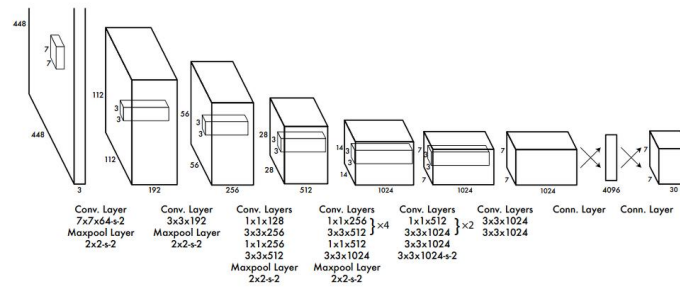


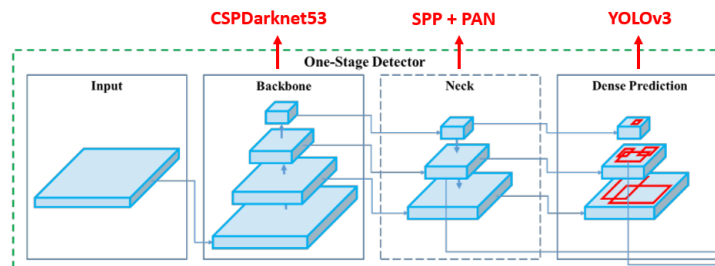
Figure 3: The Architecture. Our detection network has 24 convolutional layers followed by 2 fully connected layers. Alternating 1×1 convolutional layers reduce the features space from preceding layers. We pretrain the convolutional layers on the ImageNet classification task at half the resolution (224×224 input image) and then double the resolution for detection.

[그림 6] YOLO 아키텍처

현재 YOLO는 버전에 따라 YOLO, YOLOv2, YOLOv3, YOLOv4, YOLOv5 가 존재하며 이번 프로젝트에서는 YOLOv4, YOLOv5 를 사용하였다.

a) YOLOv4

YOLOv4 는 YOLOv3 에서 Bag of Freebies, Bag of Specials, CSPDarknet53 + SPP, PAN 과 같은 다양한 딥러닝 알고리즘이 결합된 모델로 scale 에 따라 다양한 버전이 존재한다.

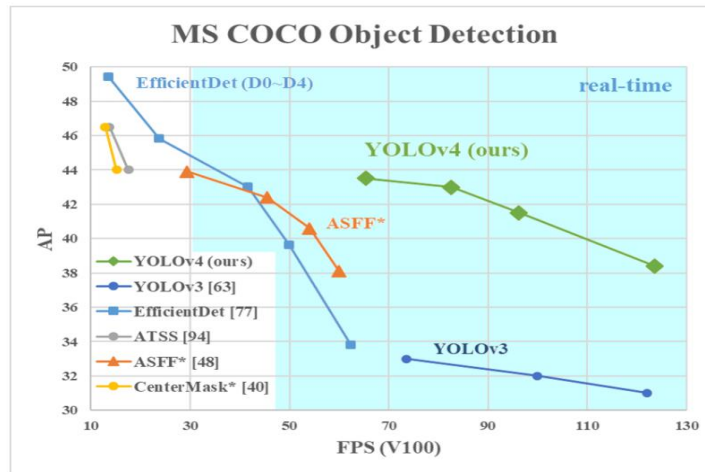


$$\text{YOLOv4} = \text{YOLOv3} + \text{CSPDarknet53} + \text{SPP} + \text{PAN} + \text{BoF} + \text{BoS}$$

↓
Path Aggregation Network

[그림 7] YOLOv4 아키텍처

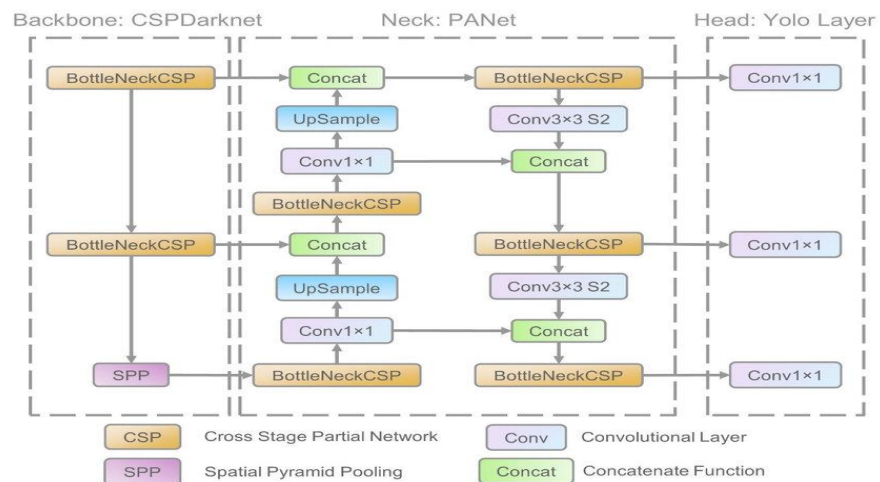
GTX 1180Ti, RTX 2080 Ti 와 같은 단일 GPU 로도 비교적 빠른 속도로 학습이 가능하며, YOLOv3 에 비해 더 정확한 객체 탐지가 가능해졌다.



[그림 8] 객체 인식 모델 성능 비교

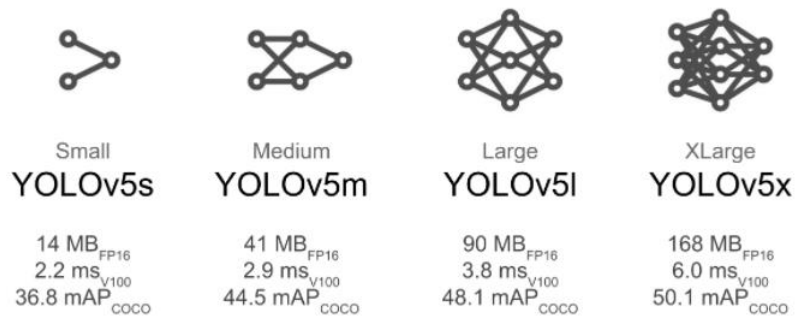
b) YOLOv5

YOLOv5 는 YOLOv4 에 비해 요구되는 자원의 양이 적고 빠른 속도를 가지고 있으며, YOLOv4 와 같이 CSPNet 기반으로 backbone 이 설계되어 있다.



[그림 9] YOLOv5 아키텍처

YOLOv5 에서 파생된 모델은 YOLOv5s, YOLOv5m, YOLOv5l, YOLOv5x 의 4 가지 버전이 있다. YOLOv5s 가 정확도가 가장 낮지만 FPS 가 가장 높고, YOLOv5x 가 성능이 제일 높지만 FPS 는 가장 낮다.



[그림 10] YOLOv5 모델 비교

2) YOLO 선정 이유

객체 인식을 자율주행에 적용하기 위하여 이번 연구는 실시간성, 정확성을 우선적으로 고려하였으며, 관련 선행 연구 분석을 진행하였다.

a) 실시간성

[표 1] 단일 단계 검출 방법과 두 단계 검출 방법 비교

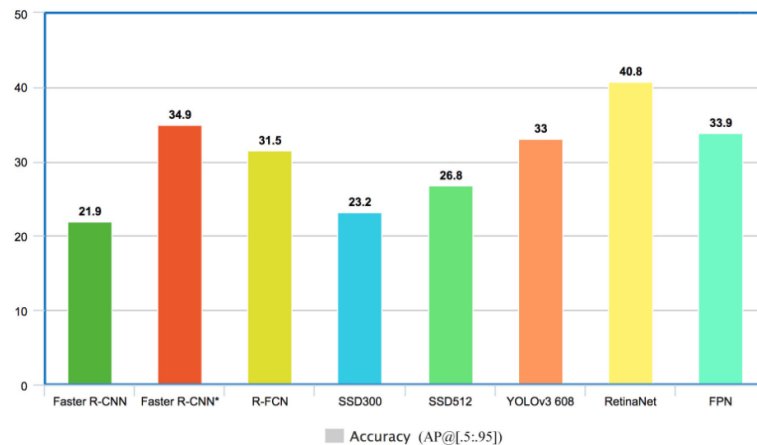
	단일 단계 검출 방법	두 단계 검출 방법
방법	모든 영역에서 Localization + Classification	1) 대략적인 Localization 2) Classification + 세밀한 Localization
정확도	합리적인 수준의 성능	매우 높은 성능
속도	매우 빠름	느림
예	YOLO, SSD	Faster-RCNN, R-FCN

객체 검출 알고리즘은 단일 단계 검출 방법과 두 단계 검출 방법으로 나뉜다.

일반적으로 단일 단계 검출이 정확도는 두 단계 검출보다 조금 떨어지지만, 처리 속도는 더 빠르다. 정확한 알고리즘이라도 동작 시간이 길어져서 실시간성이 깨지게 된다면 자율주행에 필요한 피드백에 문제가 생길 것이라 판단하여 단일 단계 검출 방법을 사용하고자 한다.

b) 정확도

대표적인 단일 단계 검출 방법인 YOLO 와 SSD 의 성능을 비교해 보았다. 처리 속도는 두 모델 모두 실시간 처리가 가능할 정도로 빨랐으며, 정확도는 COCO 데이터셋에 의하면, YOLOv3 가 SSD 계열보다 정확도가 더 높다.



[그림 11] YOLO, SSD 및 다수 모델 성능 비교

c) 선행 연구 분석 및 결론

테니스공 수집 로봇과 관련된 선행 연구들을 분석한 결과 대다수의 연구에서 객체 탐지 모델로 YOLO 를 채택되었고 특히 YOLOv3 가 주로 사용되었음을 알 수 있었다. 따라서 이번 프로젝트에서는 실시간성, 정확도, 선행 연구들을 모두 고려하여 실시간 객체 탐지 모델로 YOLO 를 선정하였으며, 그 중 가장 성능이 좋다고 판단된 YOLOv4, YOLOv5 로 테스트를 진행한 후 최종 모델을 선정하였다.

3) 학습과정

a) 1차 데이터셋 확보

1차로 수집한 데이터셋 이미지는 3600장이다. 실제 환경에서 직접 영상 촬영 후, 촬영된 영상을 약 3600여장으로 분할 후, YOLO MARK를 사용하여 확보된 데이터셋에 직접 Bounding Box를 그려 객체를 포함한 Box의 좌표를 지정하였다.

라벨링 작업 후 YOLOv4, YOLOv4tiny, YOLO5s 를 학습하였으나 실시간 객체 검출 테스트 시, 테니스공 이외의 객체도 인식되는 오류가 있었다. 동영상 분할 데이터셋의 경우 특정지역에서 촬영된 유사한 이미지이기에 다양한 환경에서의 객체 검출 오류가 존재하는 것으로 판단되어 2차 데이터 수집을 진행하였다.



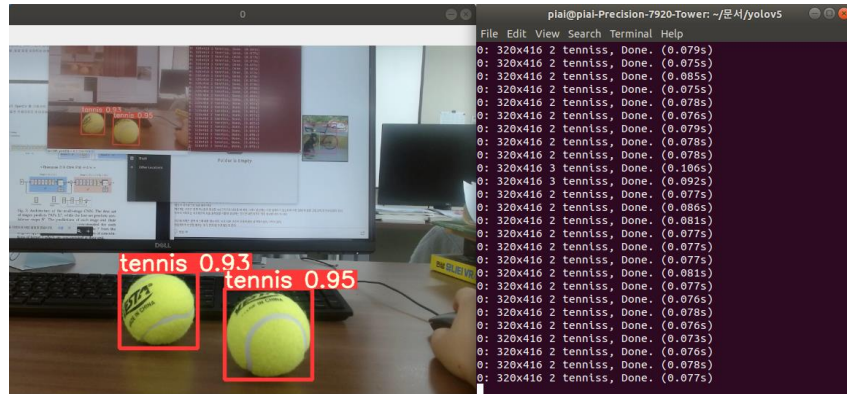
[그림 12] 1 차 데이터 수집 예시

b) 2차 데이터셋 확보

2차로 수집한 데이터셋 이미지는 기존의 3600장과 추가로 수집한 4320장을 합쳐서 총 7920장이다. 다양한 배경, 빛, 각도 등에 따른 추가 데이터셋 360장을 확보하였으며, 기존 데이터에 노이즈 추가 등을 통해 데이터 증식을 진행하였다. 변경된 데이터셋으로 재학습한 결과 인식률이 매우 높아졌지만 여전히 작은 확률로 테니스 공 이외의 객체를 인식했다. 유사 개체 또는 테니스 공이 없는 환경에서 오검출이 있다고 판단되어 3차 데이터 수집을 진행하였다.

c) 3차 데이터셋 확보

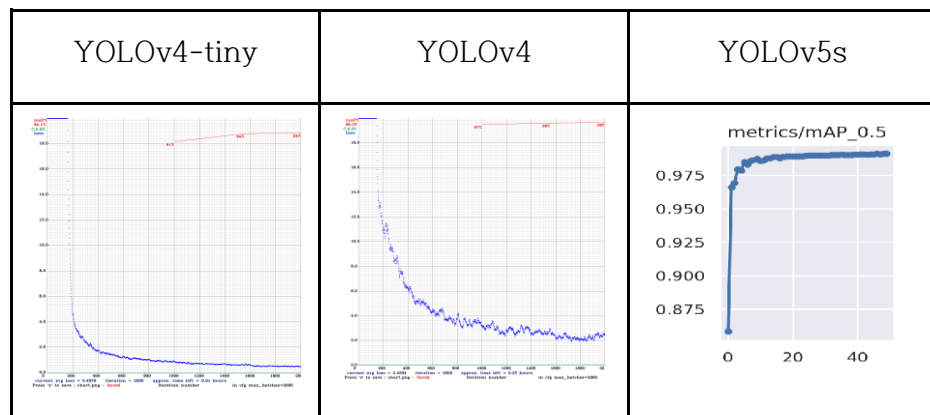
3차로 수집한 데이터셋 이미지는 기존의 7920장과 추가로 수집한 3100장을 합쳐서 총 11020장이다. 이미지 크롤링을 통해 사과, 오렌지, 시계 등 테니스 공 이외의 동그란 이미지를 3100여장 확보한 후 재학습을 진행하였다.



[그림 13] YOLOv5s 모델 - 실시간 객체 인식 예시

d) 학습 결과 비교

[표 2] 선정 모델 학습 결과 비교

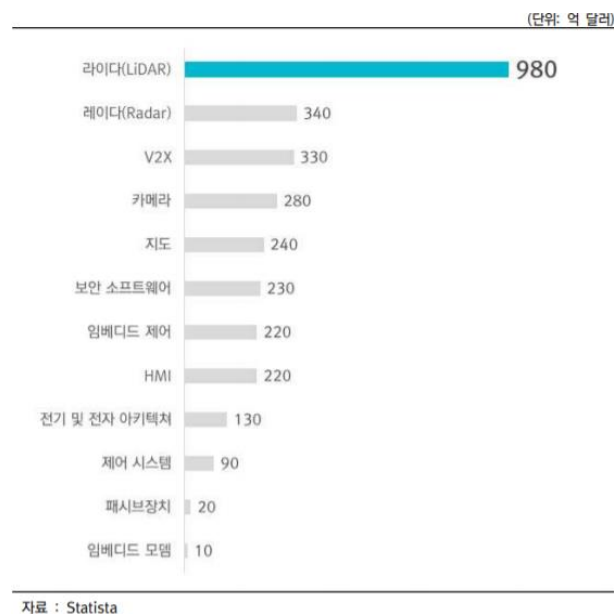


학습 후 mAP 를 비교했을 때 각 모델의 정확도가 모두 90% 이상으로 비교적 높은 성능을 나타냈다. 그 중 YOLOv4 의 성능이 가장 좋아 최종 모델로 선정하였다.

2.3 자율주행

1) 자율주행의 트렌드 및 프로젝트에 사용된 센서

데이터 분석 전문회사 스태티스타는 자율주행 산업의 발전과 함께 시장 잠재력이 높은 기술 중 하나로 LiDAR 를 뽑았다. LiDAR 는 감지할 범위에 빛을 쏘아 물체에 부딪혀 반사되는 거리, 방향 등의 정보를 확인한다. 물체와 센서와의 물리적인 위치 상태를 확인할 수 있는 LiDAR 는 완전 자율주행을 위하여 필수불가결한 요소이다. 이번 프로젝트에서는 LiDAR 를 이용하여 장애물의 회피 동작을 구현하였다.



[그림 14] 자율주행 트렌드 기술

자율주행 기술에는 LiDAR 뿐만 아니라 컴퓨터 비전을 통하여 카메라로 받아들인 이미지 데이터의 처리 및 분석 또한 중요하다. 테슬라는 Radar 와 LiDAR 없이 8개의 카메라를 이용한 컴퓨터 비전으로 자율주행을 구현하겠다고 하였다. 이번 프로젝트에서는 카메라를 이용하여 객체 탐지를 실행하고 객체의 위치를 파악하여 자율주행에 반영하는 기술을 구현하였다.

2) 강화학습

Depth Camera, LiDAR 와 같은 다양한 센서들을 활용하여 자율주행을 연구한 논문들을 조사하였을 때 DQN 을 활용한 ROS 연구가 있었다. 로봇이 장애물을 능동적으로 회피하고 목적지까지 도달하는 모델을 구현하는데 로봇의 동작을 전부 프로그래밍 하지 않고 센서 값들을 이용하여 지정된 동작을 수행하도록 하는 강화학습이 이번 프로젝트에서 필요하다고 판단하였다. 이번 프로젝트에서는 ROS 내의 Gazebo 라는 시뮬레이션 환경을 이용하여 장애물을 회피하는 모델을 학습시키고 실제 환경에 적용시키려고 시도하였다.

a) 강화학습이란

강화학습은 Agent 가 환경(Environment) 과 상호작용하며 어떤 상태(state) 에서 어떤 행동(action) 을 취하는 것이 가장 큰 보상을 받을 수 있는지 학습한다. 이러한 학습과정은 다양한 상황에서 agent 가 행동에 대하여 음, 양의 보상을 피드백 받음으로써 진행된다.

보상은 학습을 위해 만들어진 모델의 파라미터에 반영되므로 모델은 음의 보상을 피하고 양의 보상을 받기 위한 행동을 강화시킨다. Agent 는 학습 과정 중 의사결정 전략(Policy) 을 점점 발전시킨다.

b) Q-learning 과 DQN

일반적인 강화학습의 개념은 주어진 상황에 대해서만 최선의 선택을 고집하는 탐욕적 알고리즘이다. 이는 모델의 의사 결정 전략에 새로운 시도를 막게 된다. 이를 해결하기 위한 방법이 Q-learning 이다. 모델에서 도출된 행동만 선택하는 것이 아니라 임의의 확률 값으로 랜덤한 행동을 취한다. Agent 는 이를 통해서 새로운 시도를 하며 모델의 방향성과 다른 부분도 놓치지 않고 학습할 기회를 갖게 된다.

DQN(Deep Q Network) 은 Q-learning 을 테이블을 활용한 학습이 아니라

딥러닝을 이용하는 것이다. Neural Network 가 보상 값을 도출해낼 수 있도록 학습시키는 것이다.

c) 강화학습 과정

학습환경은 ubuntu 18.04, ros 1 melodic, gazebo 9 버전을 이용하여 구성되었다. 모델은 Fully Connected layer 3개로 이루어져 있다. 360도를 2D의 선형으로 탐지하는 LiDAR 센서를 0.26 라디안(15도) 을 기준으로 샘플링하여 24개의 거리값 샘플 데이터, 목표 지점과의 거리 값, 목표 지점과 로봇의 방향 벡터가 이루는 각, LiDAR 센서에서 감지된 가장 작은 거리 값, 그리고 그 거리 값의 인덱스 값까지 총 28개의 Input Data 를 만들었다.

Output 값은 0 부터 4 까지의 5개의 상태를 나타내는 값으로 출력되도록 하였다. 직진 속도는 고정되어 있으며, 각 상태 값은 로봇의 각속도로 반영되도록 하였다.

학습률은 0.00025 로 주었고, 움직이는 장애물이 있는 환경에서 시뮬레이션 하였다. 충돌 시 -200, 목적지에 도달할 시 +300 의 보상 값을 책정하고 충돌 시에는 시뮬레이션 에피소드를 리셋하도록 하였다.

d) 강화학습 과정 중 발생한 문제점

이번 프로젝트에서 강화학습을 적용하기 위하여 많은 시행착오를 겪었다. 시뮬레이션 환경에서 어느정도 장애물을 회피하여 목적지에 도달하는 모델을 구성하였지만, 실제 환경에 적용할 때 의도하지 않은 동작이 많이 발생했다. 오류가 발생한다고 추정되는 이유를 크게 세가지로 추정하고 보완하려고 하였다.

첫번째로 센서를 통해 들어오는 데이터의 노이즈로 추정하였다. 정지한 상태에서 LiDAR 를 통해 들어오는 거리 값이 5cm 이내의 비교적 큰 오차가 계속 발생하였다. cm/s 단위로 로봇의 속도 값을 변경하는데 해당 오차가 무시할 수 없을 정도로 크다고 보았다.

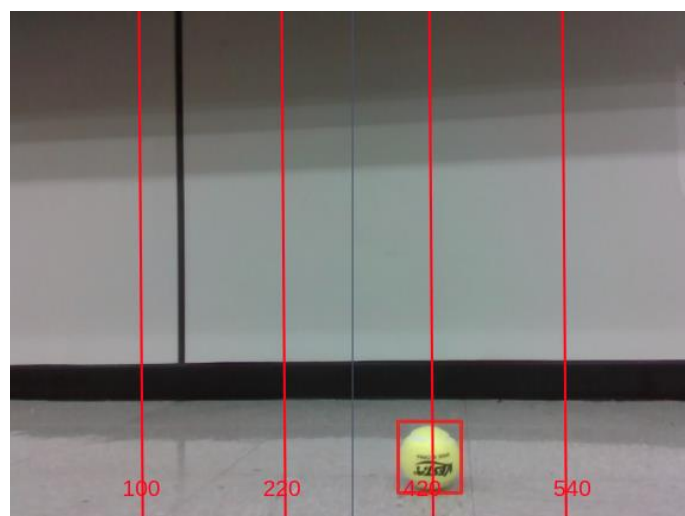
두번째로 로봇의 모터와 자이로 센서를 통해 추정하는 위치 좌표 값의 오차로 추정하였다. 모델의 output 값으로 로봇의 각속도를 지정된 값으로 설정하는데 배터리의 전압 값, 노면의 마찰 변화 등으로 모델이 의도한 동작을 로봇이 완벽하게 옮기지 못하였다.

마지막으로 학습 모델이 필요한 로봇 동작의 특성을 잘 반영하지 못하였다고 판단하였다. 학습 데이터의 분산은 모델이 학습을 거듭하면서 분산이 커지게 되고 학습 자체의 효율이 점차 감소하므로 초기 학습의 방향성이 중요하므로 적절한 모델을 찾기 위하여 다양한 환경을 구성하고 학습률, 보상에 대한 테스트를 진행하였다. 하지만 로봇의 보상으로 주어진 행동원리인 장애물 회피와 목적지 도달 중에서 목적지에 도달하기보다 장애물을 회피하는 것에 지나치게 편중되게 학습되는 것을 방지할 수 없었다.

다양한 원인으로 발생하는 통제변인의 무결성을 오차 범위 내로 유지할 수 없었기에 강화학습 대신 알고리즘을 통한 자율주행으로 방향성을 변경하였다.

3) 구현 결과

a) 객체 추적

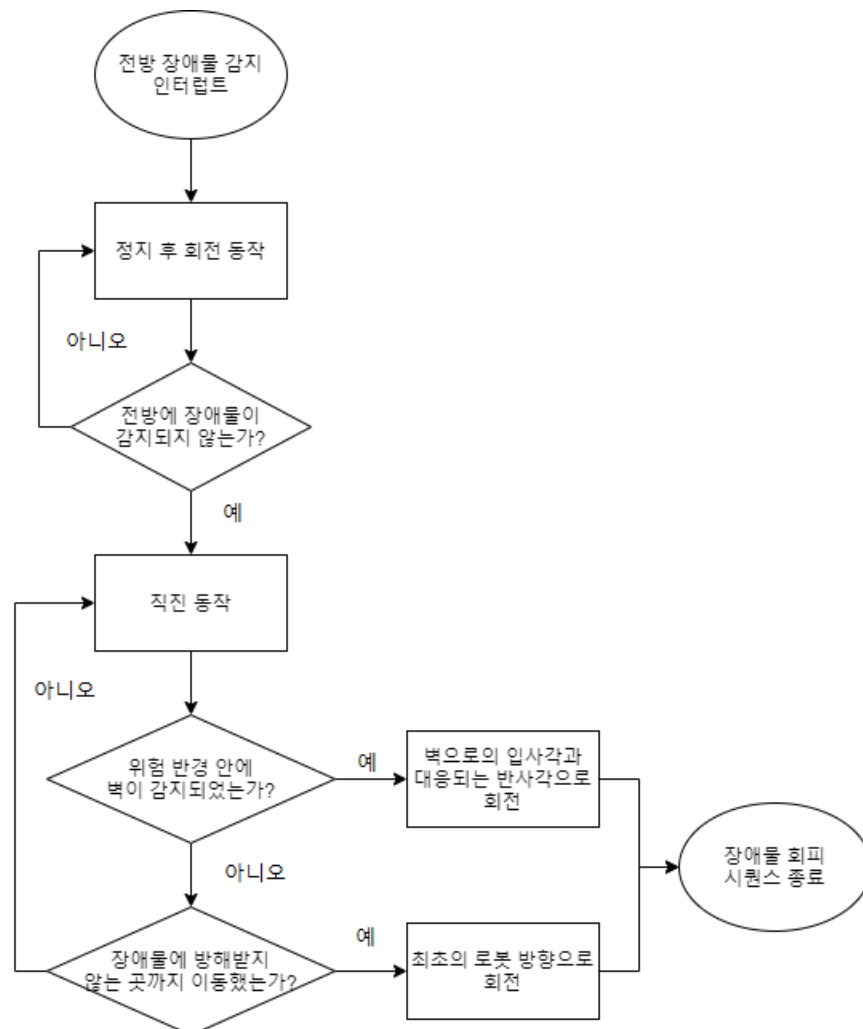


[그림 15] 탐지된 객체의 예시

Depth Camera 를 통해 객체가 탐지된 좌표를 노드 통신을 통해 수신하게 되면 객체가 탐지된 좌표의 x 값에 따라 이동 각도를 설정하였다. 그리고 탐지된 객체가 없다면, -1 값을 반환하여 객체 탐색 동작을 수행하도록 하였다.

카메라에 다수의 객체가 탐지된다면 객체들 중 depth camera 로 검출한 해당 위치의 거리 값을 받아서 최단거리에 있는 하나의 객체를 선택하여 추적할 수 있도록 하였습니다.

b) 장애물 회피



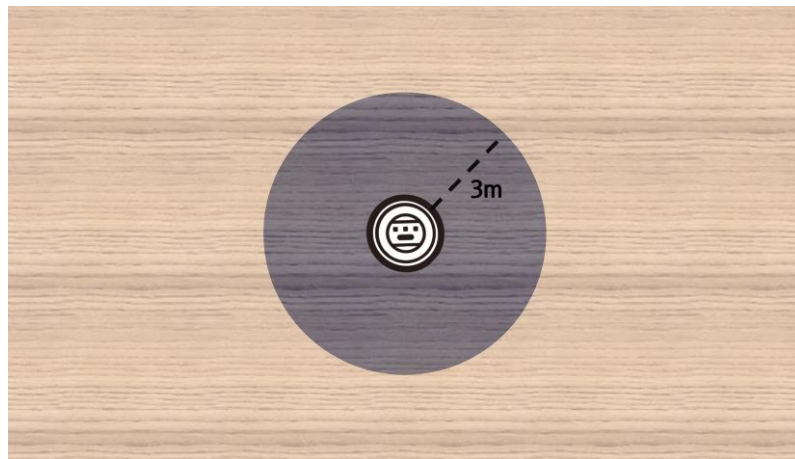
[그림 16] 장애물 회피 Flow Chart

장애물 회피는 LiDAR 를 통해 받아들인 거리 값 중 진행방향의 거리 값을

이용하여 실행하게 된다. 동작은 크게 두가지의 분기로 나뉘어 동작하도록 하였다. 벽이 아닌 장애물이 감지되었을 때는 장애물의 측면을 향해 회전하여 직진하는 동작을 수행하도록 하였다. 벽이 감지되었을 때는 입사한 방향과 반대 방향으로 뺏어나가는 반사각을 30도로 하여 튕기는 방향으로 회전하도록 하였다.

c) 객체 탐색 주행

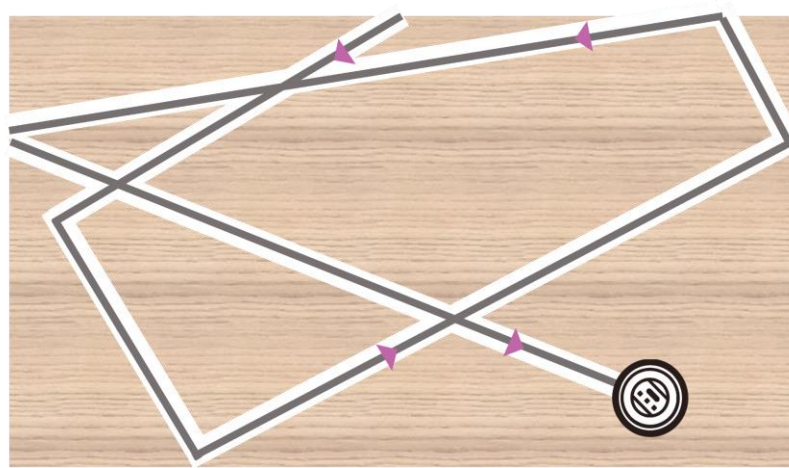
자율주행 중 카메라 프레임 내에서 객체가 탐지되지 않을 때, [그림 17] 과 같이 해당 시점부터 360도의 회전을 통해 반경 3m 내의 객체를 탐색한다.



[그림 17] 제자리 360 도 회전

해당 동작 이후에도 객체를 탐지하지 못할 경우, 기존 로봇 청소기의 주행 알고리즘을 두가지를 도입하여 객체를 탐색하도록 구현하였고 실험을 통해 직진 및 반사각 방향 회전 주행 방식을 채택하였다.

직진 및 반사각 방향 회전 주행 방식은 [그림 18] 과 같이 벽에 부딪힐 때까지 전진 후 30도만큼 반사되는 각도로 회전하며 객체를 탐색한다.



[그림 18] 직진 및 반사각 방향 회전 주행

나선형 주행 방식은 [그림 19] 과 같이 나선형으로 원을 조금씩 크게 그리며 객체를 탐색한다. 로봇의 각속도를 고정하고 선속도를 10초마다 10% 씩 증가시키며 나선형 주행을 구현하였다.



[그림 19] 나선형 주행

두가지 방식의 객체 탐색 주행 알고리즘에 대해 무작위로 위치한 테니스공 10개의 수거 완료 시간을 측정하였다. 이 중 평균 수거 시간이 적은 직진 및 반사각 방향 회전 주행 알고리즘을 최종적으로 선정하였다. 총 3세트의 실험을 진행하였으며 세트별 수거 시간은 [표 3] 과 같다.



[그림 20] 실제 실험 환경

[표 3] 객체 탐색 주행 알고리즘 별 수거 완료 시간

	1 세트	2 세트	3 세트	평균 수거 완료 시간
직진 및 시계방향 회전 주행	2 분 32 초	2 분 12 초	2 분 20 초	2 분 21 초
나선형 주행	2 분 45 초	3 분 15 초	3 분 32 초	3 분 30 초

2.4 하드웨어





자율주행의 로봇의 베이스는 Turtlebot3 burger 모델을 구입하여 이를 기반으로 하드웨어를 제작했다. Turtlebot3 는 라즈베리파이와 아두이노 기반 OpenCR 보드를 이용하여 로봇을 조작한다. 하지만 컴퓨터 비전과 자율주행을 함께 구현하는데 라즈베리파이의 연산 능력이 부족하다고 판단하여 Nvidia Jetson TX2 로 대체하였다.

로봇은 4개의 층으로 이루어져 있다. 각각의 층에 모터, 컨트롤러, 센서 등이 배치되며

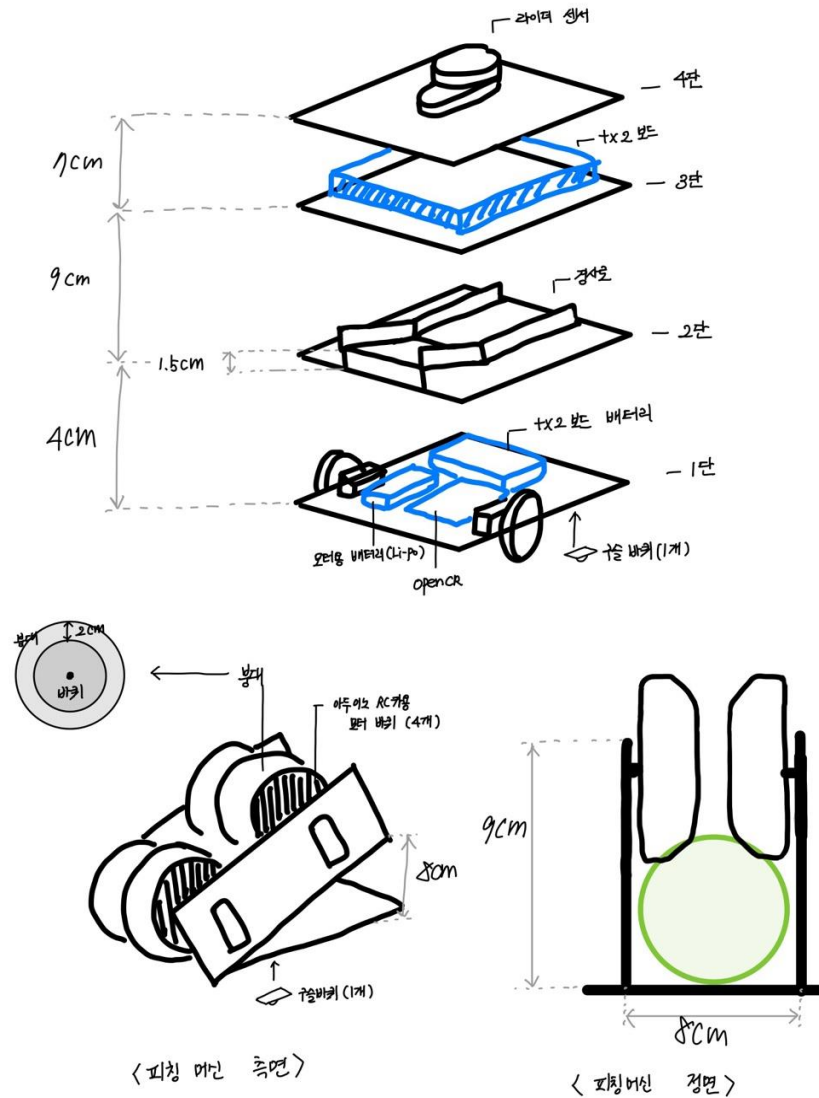
전방에는 공을 수거하기 위한 PVC 팔과 모터가 탑재되었고 후방에는 수거된 공을 담기 위한 바구니가 배치되었다.

1) 주요 부품 리스트

[표 4] 로봇에 사용된 보드 및 센서

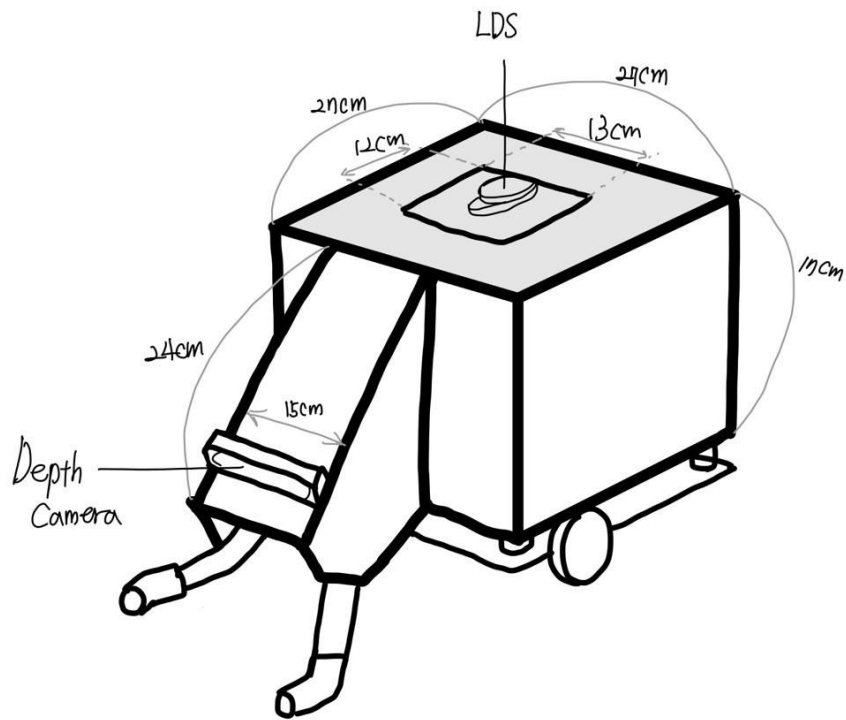
제품 위치	제품 이미지	제품명
1 단		OpenCR1.0
3 단		NVIDIA Jetson TX2
4 단		360 LDS-01 (LiDAR)
외부		Intel Depth Camera D435

2) 하드웨어 구상도 및 규격



[그림 21] 내부 구상도

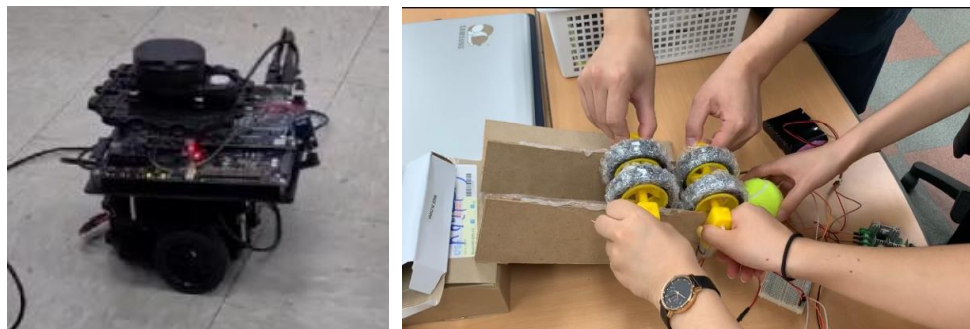
전체적인 로봇의 구성은 필요한 부품들을 쌓아 올리는 것이 초점을 맞추었고 목표로 하는 공을 수거하는 동작을 위하여 두번째 층은 공이 통과할 수 있는 통로를 만들었다. 공은 피칭머신을 모방한 네 개의 모터를 통하여 정면으로 들어간 뒤 통로를 통하여 후면의 바구니로 수거된다.



[그림 22] 외부 구상도

내부 회로와 배터리를 보호하기 위하여 로봇을 감싸는 외장을 제작하였다. 또 공의 수거를 원활하게 하기 위하여 로봇의 폭과 비슷한 폭을 가지는 팔을 로봇의 정면에 배치하였다.

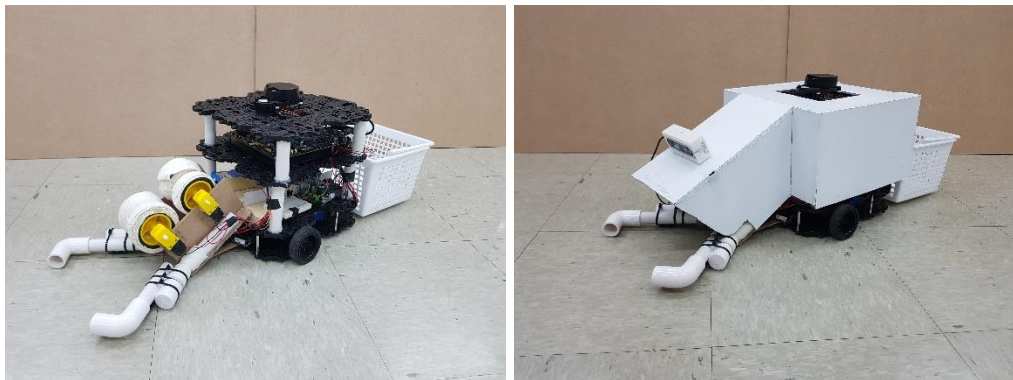
3) 제작 과정



[그림 23] 자율주행 테스트(좌) 및 피칭머신 테스트(우)

하드웨어는 테스트를 위해서 작은 크기의 로봇 형태로 먼저 제작하였다. 자율주행과 객체 추적을 테스트하면서 공을 수거하기 위한 피칭머신을 동시에 제작하였고 추후에 분해 후 현재의 형태로 재조립하였다.

2.5 결과



[그림 24] 구현 결과 - 내부(좌), 외부(우)

실제로 구현한 로봇은 turtlebot3 의 추가 부품과 PVC 파이프를 이용하여 골격을 만들었고 부품을 층별로 탑재한 형태가 되었다. 외부는 합판을 재단하여 내부를 보호할 수 있게 하였고 전방에는 공을 모터로 모으기 위한 갈고리형 팔과 후방에는 공을 담는 바구니가 장착되었다.

로봇은 동작을 시작하면 공을 탐색하기 위한 동작, 공을 추적하는 동작, 장애물을 회피하는 동작을 유기적으로 전환하면서 수행하였다. 무작위로 흩어져 있는 테니스공을 추적하여 수거하고 탐색을 하며 도중에 벽과 장애물을 만날 때는 부딪히지 않게 스스로 회피하는 동작을 보여주었다.

3. 기타

3.1 기대 효과

1) 볼퍼슨의 부상 예방

기존에는 경기장에서 공을 줍는 볼퍼슨의 역할을 Smart BallBot 이 대신함으로써 볼퍼슨이 경기장에서 공에 맞는 사고를 예방할 수 있기를 기대한다. 이는 단순히 사람이 다치지 않는데서 그치지 않고 사람의 부상을 방지한 사용자들의 컨디션에도 도움이 될 것이다.

2) 개인 스포츠 활동의 질 향상

개인 스포츠 활동 중 공을 직접 주워야 하는 불편함을 해소하고 충분한 연습 시간을 확보함으로써 스포츠 활동의 질적 향상을 기대할 수 있다. 운동 후 정리에 대한 부담을 줄여 사용자에게 운동에만 집중할 수 있는 환경을 제공하여 높은 만족감을 제공할 수 있을 것이다.

3) 다양한 시장에서의 성장 가능성

테니스 뿐만 아니라 최근 큰 성장세를 보이고 있는 골프, 야구 등 기타 구기 스포츠 종목에서 활용할 수 있기 때문에 솔루션 사업모델로서 성장 가능성을 기대할 수 있다. 비교적 넓은 환경에서 활용할 수 있게 로봇을 개조한다면 객체 인식 모델과 자율주행을 결합하여 다양한 환경에서 적용할 수 있는 로봇을 만들 수 있을 것이다.

또한 이미지 센서를 통한 객체 인식 모델과 자율주행에 대한 솔루션은 스포츠 뿐만 아니라 인명 구조나 물류 운송 등의 분야에서도 활용될 수 있을 것으로 기대된다.

3.2 한계 및 발전 방향

1) 단일 객체의 경로 계획

전체 맵에 대한 정보가 존재하지 않아 객체 인식에 대한 한계 범위가 존재하여 효율적인 경로 계획을 생성하지 못하였다. 전체 맵을 인식하거나 수색 가능 범위를 확대하기 위한 방안을 모색한다면 더 많은 객체 정보를 획득할 수 있을 것이라고 추정된다.

이러한 점은 이미지 데이터를 수집할 수 있는 추가적인 센서를 배치하거나 로봇에 탑재된 카메라의 위치를 조정하여 수색 가능 범위를 늘리거나, 전방위 물체를 인식할 수 있는 고성능의 LiDAR 등을 배치하여 전체 맵과 수거해야 할 객체에 대한 정보를 수집할 수 있게 하는 방법으로 개선할 수 있을 것이다.

2) 강화학습의 실적용

최초에 구상한 강화학습을 통한 동적 장애물의 회피를 적용하기 위하여 노력하였으나 시뮬레이션 환경과 실제 환경의 간극이 커서 적용할 수 없었다. Input 데이터가 통제되는 시뮬레이션 환경과 노이즈가 섞여서 오차가 생기는 실제 환경에서 강화학습된 모델이 의도한 동작을 제대로 실현하지 못하였다.

이는 강화학습 모델을 설계할 때 Input 데이터를 충분한 전처리 과정을 거치도록 하고 노이즈가 섞인 데이터의 오차 분산을 고려하여 유동적으로 대처할 수 있도록 해야 할 것이다. 또한 모델이 도출한 결과를 정확하게 반영하여 모터가 동작할 수 있도록 조정하는 작업이 추가적으로 필요할 것이다.

3) 센서 노이즈 및 노면 특성

로봇의 제작에 사용된 LiDAR 와 depth camera 에서 들어오는 데이터에 무시할 수

없는 크기의 오차가 발생하였다. 또한 로봇이 주행하는 노면의 특성에 따라 로봇의 동작이 영향을 받았다.

정확도가 높은 센서를 사용하거나 센서의 데이터를 노이즈를 감안한 전처리를 하고 로봇의 바퀴가 큰 것을 사용하여 노면의 특성에 상대적으로 덜 영향을 받는 설계를 한다면 이러한 한계를 극복할 수 있을 것이다.

3.3 프로젝트 진행 일정

[표 5] 프로젝트 진행 일정

	7 월				8 월			
	1	2	3	4	1	2	3	4
계획 및 일정 수립								
기술/ 하드웨어 조사								
객체 탐지								
자율 주행								
하드웨어 제작								
기술/ 하드웨어 연동								
점검 및 보고서 작성								

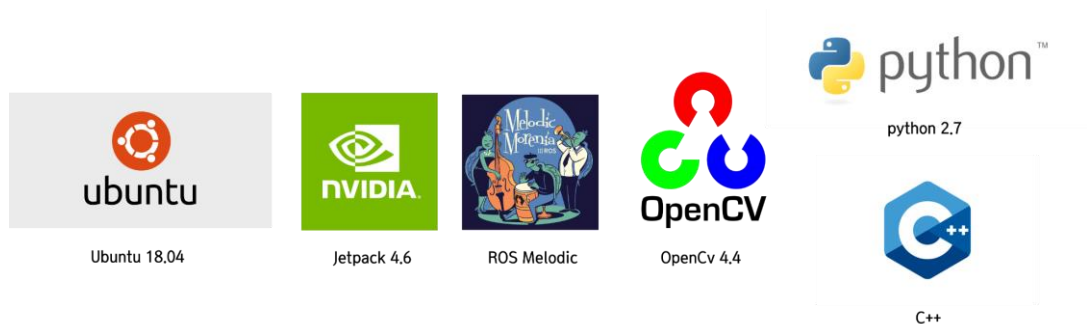
본격적으로 프로젝트 일정에 돌입했던 7월 초부터 8월 말까지 두 달 간의 바쁜 시간들이 있었다. 팀원들 간의 의견 교환으로 주제가 확정된 때부터 계획을 세워서 매주 두 번의

모임을 가지며 의견을 교환하는 시간을 가졌다.

비교적 늦게 배송되는 로봇을 고려하여 먼저 객체 인식을 위한 데이터 수집 및 학습을 위한 팀과 가상환경에서의 자율주행 강화학습을 담당한 팀을 나누어 각자 맡은 역할을 수행하였다. 이후 강화학습의 실제 적용에 대한 회의를 통하여 하드웨어를 제작하고 알고리즘을 통하여 자율주행의 제작에 들어갔다.

최종적으로 로봇의 기능은 마지막 주에 완성이 되었고 이후 발표 자료와 보고서를 작성하고 검토하는 시간을 가졌다.

3.4 개발 환경 소개



[그림 25] 사용 환경 목록

개발 환경은 연산 능력을 위하여 Jetson TX2 를 사용하기로 결정한 뒤 이에 맞추어 구성하게 되었다. Jetpack 4.6 을 통하여 설치된 TX2 의 운영 체제는 Ubuntu 18.04 버전이었고 해당 운영 체제는 ROS 1 Melodic 버전을 지원하였다.

해당 ROS 버전은 Python 2 버전을 지원하였고 이미지 프로세싱을 위한 OpenCV 4.4 버전을 이용하기 위하여 C++ 로 코드를 작성하여 두 프로그래밍 언어를 별도의 노드와 환경 설정으로 구동할 수 있게 하였다.

4. 팀원 소개 및 소감



[그림 26] 팀원 소개

박효정 ; 컴퓨터 비전에 관심이 있어 객체탐지조에 지원했었다. 데이터 수집부터 모델 학습까지 전반적으로 경험해보고, 모델의 성능 향상도 시켜볼 수 있어서 좋았다. 객체탐지조의 일이 대부분 마무리되고 새로운 조를 편성할 때 자율주행 쪽을 해보고 싶었으나 하드웨어를 맡게 되었다. 하드웨어를 직접 만드는 작업은 처음이라 자신이 없었는데, 계속 도전하고 고치고 하다 보니 어느새 만들어져 있었다. 생소한 프로젝트라 고생을 많이 해서 그런지 결과물이 나올 때쯤에는 많이 기뻐던 것 같다. 더 구현하고 싶은 부분이 많았지만 시간이 부족해서 아쉬웠다. 팀원분들 열심히 해줘서 고마워요. 모두 고생 많으셨어요!

김수영 ; 수업을 들을 때는 어떻게 적용할 수 있을지 감이 잡히지 않아서 막막했는데 프로젝트를 하면서 전기수 프로젝트나 수업 때 했던 내용을 더 잘 이해할 수 있었습니다. 처음 하는 프로젝트라 아쉬움이 많이 남고 다음 번에 프로젝트를 하게 된다면 좀 더 사회적으로 의미 있는 프로젝트를 해보고 싶습니다

이진주 ; 처음으로 하드웨어를 사용한 프로젝트를 진행할 수 있어 기뻐고, 생각 이상으로 쉽지 않아 정말 힘들었지만 잘 이겨내어 뿌듯합니다. 객체인식과 자율주행 모든 파트를 다뤄볼 수 있어 많은 성장을 할 수 있었습니다. 아직 아쉬운 점이 많고 개선의 여지가 많은 프로젝트이기에 다음기수에서 스마트볼봇을 업그레이드해주면 좋을 것 같습니다

배수영 ; 이론으로만 접했던 YOLO 와 강화학습을 활용한 프로젝트여서 의미 있었습니다. 또한 처음으로 ROS 프로그래밍과 터틀봇 하드웨어를 다뤄볼 수 있어서 흥미로웠지만, 생각보다 어려워서 아쉬움이 남습니다. 다음에 또 새로운 것에 대해 도전하게 된다면, 어떻게 해야 할지 알아가는 기회였습니다.

여동훈 ; 이전에 학교에서 배웠던 센서나 하드웨어에 관한 지식과 휴스타 과정 중 학습했던 이미지 프로세싱, 머신 러닝, 딥러닝 등의 인공지능을 결합시키는 경험을 할 수 있어서 역량강화를 하는데 큰 도움이 되었습니다. 프로젝트 팀원들과 함께 했던 휴스타 과정은 제 삶에서 잊을 수 없는 소중한 경험이었습니다. 모두 수고하셨습니다. 감사합니다!

5. 참고 문헌

[YOLO]

[1] You Only Look Once: Unified, Real-Time Object Detection

<https://arxiv.org/pdf/1506.02640.pdf>

[2] YOLOv4: Optimal Speed and Accuracy of Object Detection

<https://arxiv.org/pdf/2004.10934.pdf>

[3] YOLOv5 <https://github.com/ultralytics/yolov5>

[4] YOLOv5 아키텍처 분석 <https://ropiens.tistory.com/44>

[YOLO 사용 이유]

[1] [오토저널] 자율주행자동차를 위한 영상인식기술 동향

http://global-autonews.com/bbs/board.php?bo_table=bd_035&wr_id=392

[2] Object detection: speed and accuracy comparison

<https://jonathan-hui.medium.com/object-detection-speed-and-accuracy-comparison-faster-r-cnn-r-fcn-ssd-and-yolo-5425656ae359>

[3] Intelligent Tennis Robot Based on a Deep Neural Network

<https://www.mdpi.com/2076-3417/9/18/3746/htm>

[4] A Deep Learning Tennis Ball Collection Robot and the Implementation on NVIDIA

Jetson TX1 Board

<https://ieeexplore.ieee.org/document/8452263>

[DQN]

<http://wiki.hash.kr/index.php/DQN>

[자율주행 트렌드]

[https://news.kotra.or.kr/user/globalBbs/kotranews/782/globalBbsDataView.do?setI
dx=243&dataIdx=186364](https://news.kotra.or.kr/user/globalBbs/kotranews/782/globalBbsDataView.do?setI
dx=243&dataIdx=186364)