

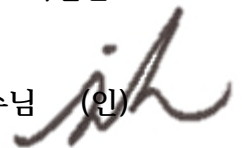
종합설계 프로젝트 수행 보고서

프로젝트명	스마트 축구 훈련 어플리케이션
팀번호	S5-2
문서제목	수행계획서() 2차발표 중간보고서() 3차발표 중간보고서() 4차발표 중간보고서() 최종결과보고서(O)

2020.11.22

팀원 : 이찬솔 (팀장)
이교범
이기열
최철환

지도교수 : 정의훈 교수님 (인)



문서 수정 내역

작성일	대표작성자	버전(Revision)	수정내용	
2020-01-17	이찬솔(팀장)	1.0	수행보고서	최초 작성
2020-02-15	최철환(팀원)	2.0	2차발표자료	설계서 추가
2020-05-01	이기열(팀원)	3.0	3차발표자료	시험결과 추가
2020-06-26	이교범(팀원)	4.0	4차발표자료	시험코드 추가
2020-11-22	최철환(팀원)	5.0	최종결과보고서	시험결과 수정

문서 구성

진행단계	프로젝트 계획서 발표	중간발표1 (3월)	중간발표2 (5월)	학기말발표 (7월)	최종발표 (10월)
기본양식	계획서 양식	계획서 양식	계획서 양식	계획서 양식	계획서 양식
포함되는 내용	I. 서론 (1~6) II. 본론 (1~3) 참고자료	I. 서론 (1~6) II. 본론 (1~4) 참고자료	I. 서론 (1~6) II. 본론 (1~5) 참고자료	I. 서론 (1~6) II. 본론 (1~7) 참고자료	I. 서론 (1~6) II. 본론 (1~7) III. 결론 참고자료

이 문서는 한국산업기술대학교 컴퓨터공학부의 “종합설계”교과목에서 프로젝트“스마트 축구 연습 어플리케이션”을 수행하는 팀원(팀번호: S5-2)들이 작성한 것으로 사용하기 위해서는 팀원들의 허락이 필요합니다.

목 차

I. 서론

1. 작품선정 배경 및 필요성	4p
2. 기존 연구/기술 동향 분석	4p
3. 개발 목표	4p
4. 팀 역할 분담	4p
5. 개발 일정	5p
6. 개발 환경	6p

II. 본론

1. 개발 내용	7p
2. 문제 및 해결방안	13p
3. 시험 시나리오	13p
4. 상세 설계	14p
5. Prototype 구현	20p
6. 시험 / 테스트 결과	25p
7. Coding & DEMO	26p

III. 결론

1. 연구 결과	
2. 작품제작 소요자료 목록	

참고 문헌	22p
-------------	-----

I. 서론

1.1 작품선정 배경 및 필요성

20세기부터 축구는 이미 세계 인기 스포츠, 시장규모, 선수 연봉에서 1위라는 순위를 유지하고 있다. 이러한 인기로 인해 축구를 배우고자 하는 사람은 수많이 존재하고 있다. 하지만 축구를 배우기 위한 레슨에는 시간당 36,000원, 유학에는 2개월에 약 360만 원이라는 고비용이 필요하다. 보다 적은 비용으로 혼자 다양한 연습을 할 수 있는 대안이 필요하고, 코치 없이 실시간으로 피드백을 받을 수 있는 환경이 필요하다고 생각하여 졸업작품으로 선정하였다.

1.2 기존 연구/기술 동향 분석

관련 제품으로는 스마트폰을 이용한 농구 연습 어플리케이션인 홈코트가 존재한다. 이 제품은 증강현실을 이용하여 혼자서 슈트, 드리블의 훈련이 가능한 농구 연습 어플리케이션으로 인공지능을 이용한 자세 교정을 통해 비싼 코치 없이 혼자 연습이 가능하다는 장점을 갖고 있다. 하지만 연습 가능한 스포츠가 농구로 한정되어 있다.

1.3 개발 목표

최종 목표로는 스마트 축구 연습 어플리케이션을 개발하여, 보다 적은 비용으로 다양한 연습을 도와줌과 동시에 개인 단위로 혼자 연습할 수 있는 환경을 만들어 주는 것이다. 단계별로 들어가자면 일단 opencv를 이용하여 파이썬 환경에서 구현해 본 뒤, 안드로이드 스튜디오 환경에서 공을 인식하여 공의 궤적과 속도, 도착한 지점 등을 영상 처리를 통한 시각화 구현과 간단한 UI 작성과 서버 구축이다. 다음 단계로는 opencv와 tensorflow_lite를 이용한 자세 인식구현 및 자세 인식의 신경망 및 딥러닝 구축으로 안드로이드 스튜디오 환경에서 신경망의 기본 형식을 구현한 후 CNN을 이용하여 세부적인 부분을 추가하여 신경망을 구축한다. 그 후 사용자의 자세를 신경망에 학습된 자세와 비교하여 사용자의 자세를 교정하는 것을 구축한다. 마지막 단계로는 1 단계에 간단히 작성되었던 UI를 좀 더 사용자 편의에 맞게 구축하고 어플리케이션으로 쉽게 연습 선택, 영상 시청, 연습 기록을 수치화 등의 기능을 제공하고 서버에 저장된 영상을 사용자에게 제공할 수 있게 인터페이스를 구축한다.

1.4 팀 역할 분담

역할 분담으로는 인공지능, 영상 처리, 앱 UI 작성, DB 및 서버 구축으로 4개로 나누어 각자 2개씩 맡음으로써 혼자서 해결하기 어려운 문제나 막히는 부분을 상호보완하여 해결할 수 있도록 하였다. 각자 역할로는 이찬솔 인공지능-영상 처리, 이기열 영상 처리-DB 및 서버 구축, 이교범 인공지능, 영상 처리, 최철환 앱 UI 작성, DB 및 서버 구축을 맡도록 하였다. 산출물 관리는 개인 PC와 구글드라이브, 개인 메일로 3개에 공간에 백업, 보관 하며 코드작성은 Github를 활용하여 수정 부분을 알 수 있도록 하였다

1.5 개발 일정

개발 일정은 다음의 표대로 진행할 수 있도록 일정을 잡았고 매주 금요일마다 만나 서로 작성한 것을 공유하고 진행 상황을 알 수 있도록 하였다.

<표 1> 개발 일정

활동 내용	9월				10월				11월				12월			
	1주	2주	3주	4주	1주	2주	3주	4주	1주	2주	3주	4주	1주	2주	3주	4주
[과제기획]																
팀 구성	●	●														
업무분담		●	●													
아이디어 회의 및 정리			●	●	●	●	●	●								
개발 아이디어 선정						●	●	●								
[요구사항 도출]																
특징, 장점, 단점 분석							●	●	●	●						
문제점 정리 및 해결방안 도출								●	●	●	●					
개발 내용(기능) 정리									●	●	●	●				
[기본설계]																
관련 API 조사 및 분석							●	●	●	●	●	●	●			
시나리오 설계									●	●	●	●	●	●		
시스템 구조 설계										●	●	●	●	●	●	●
시스템 평가방법 설계													●	●	●	●
최종기획 보고서 완료														●	●	●

1.6 개발 환경

개발 환경으로는 앱을 작성할 수 있도록 안드로이드 스튜디오를 사용하고 언어로는 자바를 사용하여 작성한다. 사용하는 스튜디오 버전은 3.4.2이다. 안드로이드 타킷은 최소 8.1(오레오)_API Level 27으로, 최대 타킷은 10.0(안드로이드 10)_API Level 29이다. 다음으로 DB는 아마존에서 제공하는 AWS-RDS를 사용하여 MY-sql DB를 구축하고 서버와 연동할 수 있도록 하였다. 사용 언어로는 sql문을, 버전은 8.0.15를 사용한다. 서버 역시 마찬가지로 아마존에서 제공하는 AWS-EC2를 사용해 unbuntu 환경을 구축하고 사용 언어로 c++을 사용하여 버전 18.0.4에 구축한다.

<표 2> 개발 환경 및 라이브러리

환경	버전	개발사	사용 가능 언어
안드로이드 스튜디오	3.4.2	구글	• 자바,코틀린, C++
Aws RDS - MY SQL	8.0.15	MySQL LAB	• sql문
Ubuntu Server	18.0.4	UBUNTU	• C++, 자바

라이브러리	버전	개발사	사용 가능 언어
안드로이드 openCV	4.1.0	인텔	• 자바, C++
tensorflow	tensorflow_lite	구글	• 자바, C++

II. 본론

2.1 개발 내용

(1) UI



<그림 1> 어플 실행 시 화면

어플 실행 시 로그인, 회원이 아니라면 회원가입을 해야 한다. <그림 1>과 같은 실행 페이지와 소셜 로그인 및 회원가입은 구글 또는 페이스북 계정으로 가능하도록 계획하였다.



<그림 2> 메인 화면 다른_사용자 게시물



<그림 3> 서브 메뉴화면_업로드 & 영상추천

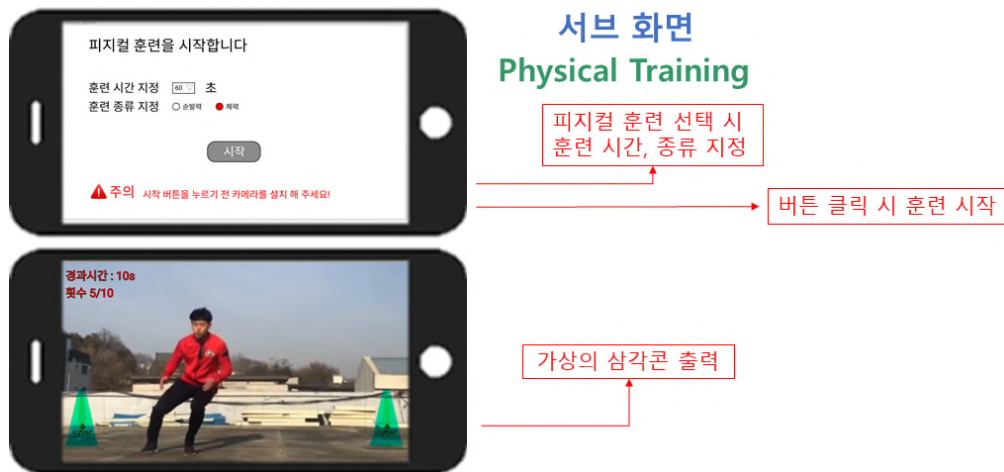
<그림 2>는 로그인 후 메인 화면으로 다른 사용자의 영상을 시간 순으로 추출하여 보여준다. 북마크 버튼으로 SNS로 공유할 수 있고 메뉴로는 영상, POST, 훈련, 프로필이 있다.

다음으로 <그림 3>은 서브화면인 POST 화면이다. 사용자의 연습 영상이 기록되고 사용자의 훈련 수치를 토대로 부족한 능력에 맞는 영상을 추천 해준다.



<그림 4> 서브 메뉴화면-훈련

<그림 4>는 훈련 화면으로 기능으로는 체력 훈련, 슈팅훈련, 트래핑 훈련이 있으며 사용자가 선택하여 원하는 훈련을 진행할 수 있다. 훈련이 끝난 후에는 자신의 게시물로 올릴 수 있고, 게시물로 올리면 다른 사용자가 볼 수 있도록 메인 화면에 추출된다.



<그림 5> 체력훈련 화면

<그림 5>는 체력훈련 화면으로 훈련 선택 시 훈련 시간과 종류를 입력받는다. 시작 버튼을 누를 시 가상의 장애물이 화면에 출력된다. 훈련은 제한시간 동안 진행되며 시간 내에 수행한 횟수를 기준으로 사용자의 체력과 순발력을 측정한다.



<그림 6> 슈팅훈련 화면

<그림 6>은 슈팅훈련 화면으로 사용자가 훈련할 항목을 선택하여 훈련이 진행되며 목표 공간마다 점수를 차별화하여 고득점의 공간을 목표로 할 수 있도록 한다. 표시되는 결과로는 다음과 같다. 목표 공간 성공 여부와 궤적을 카메라 트래킹으로, 슛 종류 일치 여부와 슈팅 동작을 딥러닝으로 전 처리된 데이터로 보여주며, 슈팅 속도와 성공률을 알려준다.



<그림 7> 트래핑 훈련 화면

<그림 7>은 트래핑 훈련 화면으로 훈련 시간을 지정하고 시작하면 무작위로 위치를 선정해준다. 훈련은 각 지정한 시간 동안 진행되며 진행될 때마다 다른 위치를 선정해주며 성공 여부와 전체 연습 횟수를 알려준다.



<그림 8> 서브 메뉴화면_profile (1)

<그림 8>은 profile 메뉴화면으로 간략한 훈련 정보와 나와 비슷한 자세를 가진 선수 영상을 제공한다. 또한, 자신의 훈련 상황을 자세히 볼 수 있도록 하며, 자신의 게시물을 관리할 수 있다. 설정에서는 자신의 개인정보를 관리하거나, 공지사항 등을 볼 수 있다.



<그림 9> 서브 메뉴화면_profile (2)

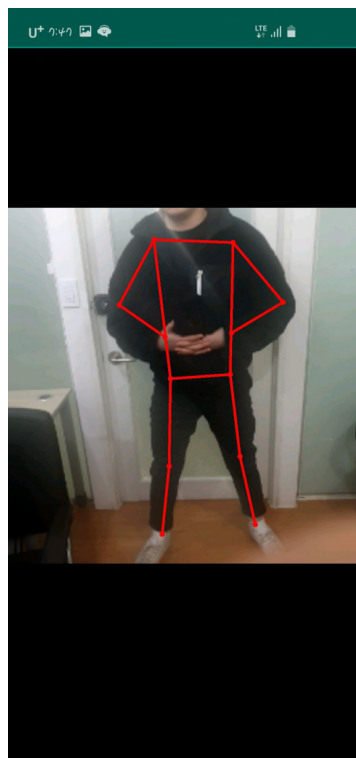
<그림 9>는 훈련 상황을 자세히 볼 수 있는 화면으로 여러 차트로 주차 별, 월별, 훈련 상황의 변화와 지난 4주간의 최고·저, 평균을 알 수 있으며 자신이 진행한 훈련 별 빈도를 제공하여 부족한 훈련을 진행할 수 있도록 한다.

(2) 영상 처리



<그림 10> 트래킹 구동 화면

openCV의 `Imgproc.HoughCircles` 함수를 이용한 원 검출로 코드를 작성하였고 <그림 10>은 CircleTrackig를 구현하여 실제 어플로 공의 원형을 인식하여 트래킹하는 모습이다

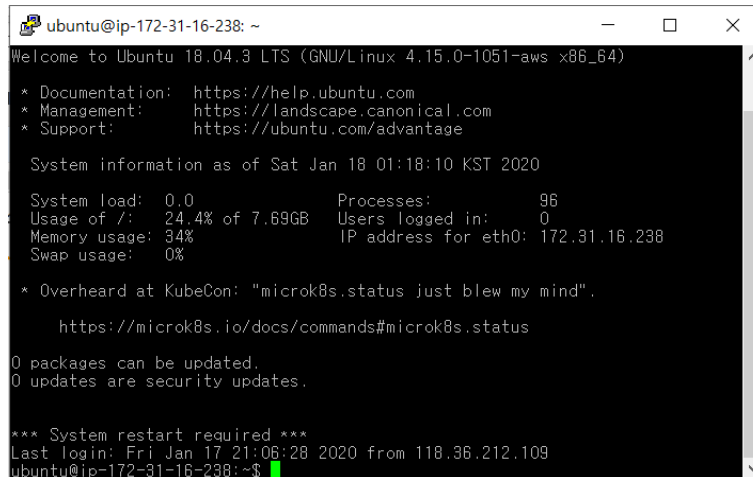


<그림 11> 자세측정 구동 화면

tensorflow_lite의 posenet 라이브러리를 이용하여 사람의 관절 인식을 구현 딥러닝으로 형성한 데이터 셋을 통해 <그림 11>과 같이 어깨, 팔, 손목, 골반, 무릎, 발목의 관절을 랜드마크로 인식하고, 사람의 자세 측정한다.

(3) 서버 구축

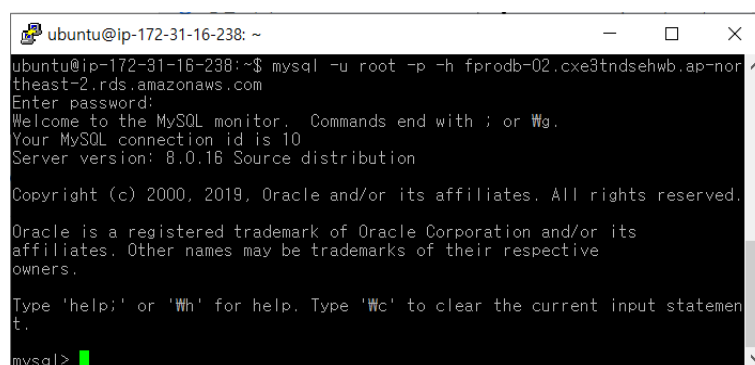
AWS EC2 인스턴스를 생성하여 ubuntu 서버를 개설 및 운용한다.



```
ubuntu@ip-172-31-16-238: ~  
Welcome to Ubuntu 18.04.3 LTS (GNU/Linux 4.15.0-1051-aws x86_64)  
  
 * Documentation:  https://help.ubuntu.com  
 * Management:    https://landscape.canonical.com  
 * Support:        https://ubuntu.com/advantage  
  
System information as of Sat Jan 18 01:18:10 KST 2020  
  
System load:  0.0          Processes:      96  
Usage of /:   24.4% of 7.69GB   Users logged in:  0  
Memory usage: 34%          IP address for eth0: 172.31.16.238  
Swap usage:   0%  
  
 * Overheard at KubeCon: "microk8s.status just blew my mind".  
  
      https://microk8s.io/docs/commands#microk8s.status  
  
0 packages can be updated.  
0 updates are security updates.  
  
*** System restart required ***  
Last login: Fri Jan 17 21:06:28 2020 from 118.36.212.109  
ubuntu@ip-172-31-16-238:~$
```

<그림 12> Putty를 통한 AWS_Ubutu 서버 접속 화면

<그림 12>는 서버에 접속한 화면으로 putty를 통해 접속하며, 인스턴스 생성 시에 등록한 Key(ppk)를 pem 파일로 변환하여 접속 시에 사용하고 암호를 입력하여 서버에 접속한다.



```
ubuntu@ip-172-31-16-238:~$ mysql -u root -p -h fprodb-02.cxe3tndsehw.ap-nor  
theast-2.rds.amazonaws.com  
Enter password:  
Welcome to the MySQL monitor.  Commands end with ; or \g.  
Your MySQL connection id is 10  
Server version: 8.0.16 Source distribution  
  
Copyright (c) 2000, 2019, Oracle and/or its affiliates. All rights reserved.  
  
Oracle is a registered trademark of Oracle Corporation and/or its  
affiliates. Other names may be trademarks of their respective  
owners.  
  
Type 'help;' or '\h' for help. Type '\w' to clear the current input statemen  
t.  
mysql>
```

<그림 13> AWS EC2 Ubuntu 통한 RDS_Mysql 접속 화면

<그림 13>과 같이 RDS로 MySQL DB를 생성하여 EC2의 ubuntu 서버를 통해 접속하며 정상적인 접속을 위해 서버의 보안규칙 중 인바운드 규칙에 DB를 포함하고 ubuntu와 MySQL에 적용한다.

2.2 문제점 및 해결방안

1. Ubuntu 서버와 Mysql DB간에 보안으로 인한 접속 문제가 발생하였다. 이를 해결한 방법으로는 Mysql DB에 접속할 특정 IP를 인바운드 규칙에서 허용하여 해당 IP주소에서 DB에 접속할 수 있도록 함으로써 해결해야 했다.

2. 공 트래킹을 할 때 색을 따라 트래킹을 하도록 하였는데 이 방식이 아닌 원이나 아니면 축구공을 인식할 다른 방법을 찾아서 트래킹을 하는 방법이 필요하였다. 원 검출을 이용하기 위해서 안드로이드 openCV의 `Imgproc.HoughCircles` 함수를 이용하여 원을 검출하였다

3. 궤적을 그려줄 때 공의 반지름이 작아질 경우 슈팅을 하였다고 인식하여 궤적이 표시되게 하려고 했으나 빠른 속도로 처리를 하는 만큼 반지름의 크기가 순서대로 저장되지 않아 올바른 궤적을 그려줄 수 없었다. 궤적을 그려주는 것을 좌표를 이용하여 슈팅한 것을 인식하도록 수정하였다.

4. 자바와 Python 연동 문제로는 Kivy를 이용하여 안드로이드 위에 파이썬 코드를 올리는 방법을 이용하여 해결하려 했으나 카메라 제어 문제로 파이썬 openCV 코드를 안드로이드 openCV로 변환하여 해결하였다

2.3 시험 시나리오

총 5가지의 시나리오를 진행하였다.

첫 번째로, openCV 라이브러리를 이용하여 공 감지로 카메라에서 초록색 물체를 감지하게 되면 초록색 물체를 공으로 인식한 후 그 테두리에 원을 그려준다.

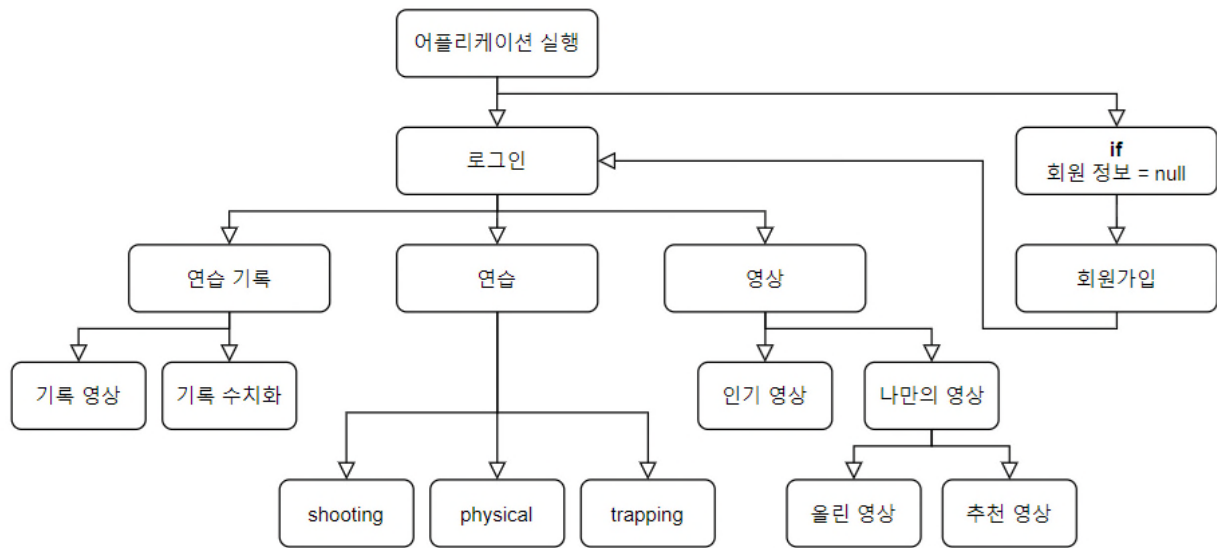
두 번째로, 인식한 공이 움직일 때 움직임을 감지하여 이동 경로를 따라 궤도를 표시, 저장, 출력한다.

세 번째로, tensorflow_lite의 posenet 라이브러리를 이용하여 사람 관절을 인식하게 되면 관절을 랜드마크로 인식한 후 표시한다.

네 번째로, 속력을 구하기 위해 거리를 구하는데 카메라로 보이는 골대 높이를 측정하여 각각 거리마다 저장한다. 충분히 쌓인 데이터를 통해 최소 자승법을 이용하여 회귀분석을 통한 골대 높이-거리의 함수를 구하고 실제 이용 시에는 골대높이를 측정하고 함수를 이용하여 거리를 구한 후 속력을 구한다.

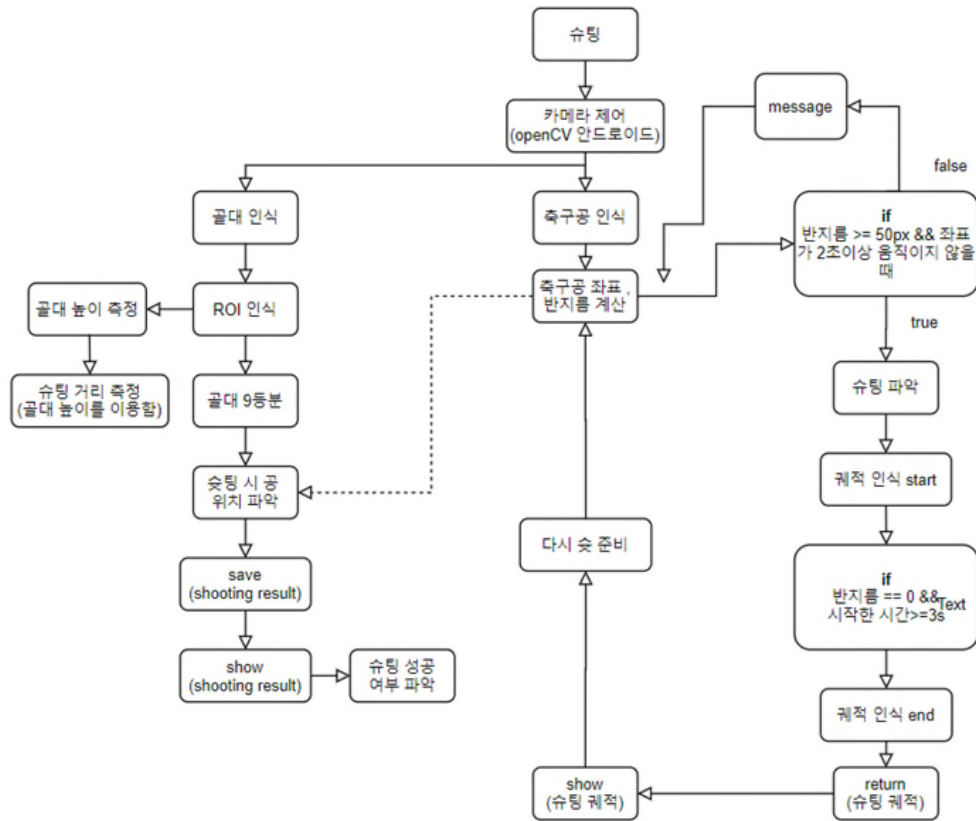
다섯 번째로, 서버 통신으로 aws-ec2를 이용하여 ubuntu 서비스를 구축한 후 서버에서 my-sql을 이용하여 db를 수집한다. 그 후 서버에서 aws-rds를 연동하여 db에 저장한다. 마지막으로 애플리케이션 통신으로 사용자가 애플리케이션을 통해 요구사항을 입력하게 되면 입력된 데이터를 db에 저장한다.

2.4 상세 설계

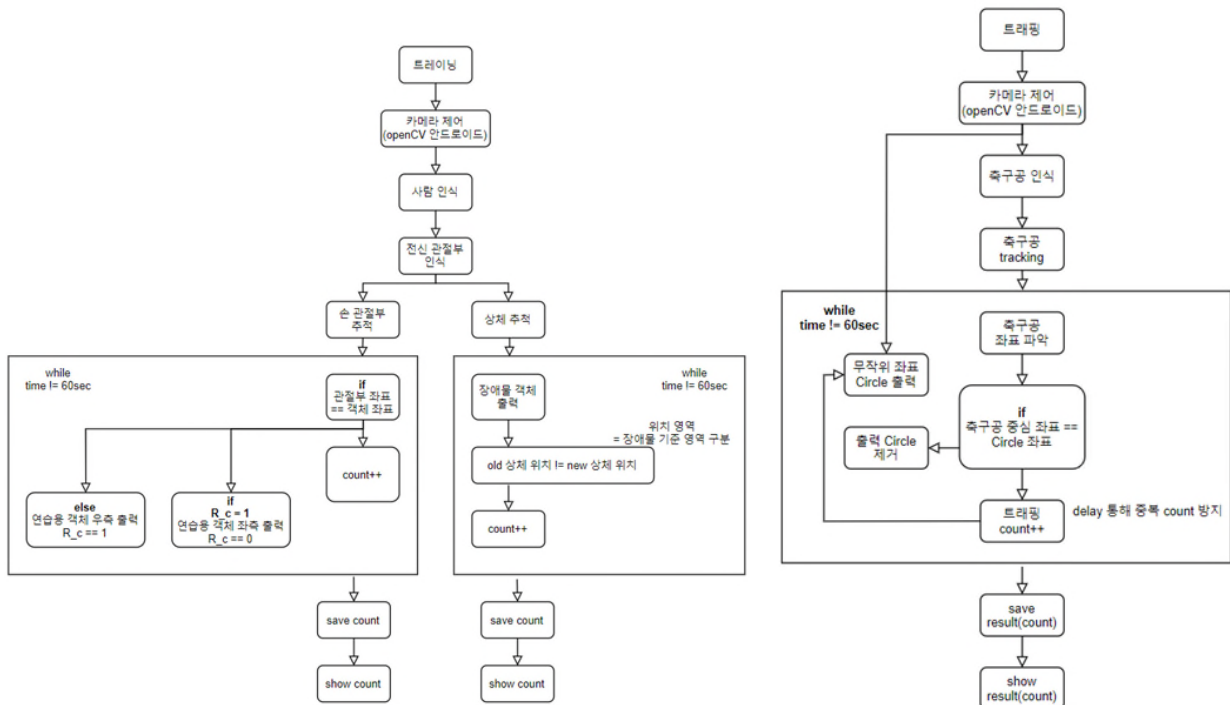


<그림 14> 전체 시퀀스

<그림 14>는 어플리케이션의 전체 시퀀스이다. 어플리케이션을 실행하게 되면 로그인을 통해 접속하게 된다. 그 후 연습을 선택하게 되면 슈팅, 피지컬, 트래핑 중 하나를 선택하여 훈련할 수 있고 자신의 연습 내용이 기록된다. 기록된 연습 영상이나 수치는 언제든지 연습 기록으로 쉽게 확인 가능하도록 하였고 영상 파트를 또 따로 두어 자신의 영상을 올릴 수 있고 사용자에게 맞는 영상을 추천해주고 인기 영상과 자신이 업로드한 영상도 쉽게 시청과 열람이 편하도록 하였다. 각 훈련의 세부 사항은 아래의 그림과 같다.



<그림 15> 슈팅 시퀀스



<그림 16> 피지컬 시퀀스

<그림 17> 트래핑 시퀀스

(1) 인식

동작의 순서는 위의 시퀀스 대로 진행이 되도록 설계를 하였고 내부 코드로 들어가면 다음과 같다. 일단 공과 골대 인식 같은 경우에는 opencv를 이용하여 인식한다. 인식할 때에는 색상을 통해 인식하며 이를 위해 HSV 색상변경을 하고 특정 색상 값을 마스크로 지정하여 트래킹을 진행하게 된다. 그 후 공의 경우 슈팅 인식은 지면에서 떨어졌을 경우라고 생각을 하고 y좌표가 감소할 때에 궤적을 그려주게 된다. 모션 인식의 경우에는 tensorflow_lite의 posenet 라이브러리를 이용하여 인식하게 된다. 그래서 사용자의 관절을 인식할 수 있게 되고 인식한 관절을 통해 지면과의 각도나 관절들 간의 각도를 이용하여 올바른 자세의 피드백이 가능하도록 해 준다.

(2) 수치화

인식한 내용으로 사용자의 기록을 수치화해준다. 수치화는 모든 연습에서 진행이 되며 각각의 연습에서 얻는 내용을 다르게 하였다. 일단 자신의 기록을 그대로 저장해 주는 것은 물론이고 슈팅 연습의 경우에는 공의 속력을 이용하여 사용자의 슈팅 파워를 측정한다. 속력을 구하기 위해 거리를 구해야 하는데 카메라로 보이는 골대 높이를 측정하여 각각 거리마다 저장한다. 충분히 쌓인 데이터를 통해 최소 자승법을 이용하여 회귀분석을 통한 골대 높이-거리의 함수를 구하고 실제 이용 시에는 골대 높이를 측정하고 함수를 이용하여 거리를 구한 후 속력을 구한다. 우리가 구한 함수는 $y=12.925x^{-1}$ 으로 결정계수 R^2 , 즉 신뢰도가 97%라는 결과를 내었다. 공이 골대에 들어간 횟수와 골대를 9등분 하였을 때 외곽에 가깝게 위치할수록 높은 점수를 주어 사용자의 정확성을 기록한다. 피지컬 연습의 경우에는 체력, 순발력 연습이 존재하는 데 시간을 기준으로 각각의 연습을 통해 점수를 얻고 그 점수를 통해 사용자의 순발력이나 체력을 측정한다. 트래핑 연습은 화면에 공을 보내야 하는 위치를 동그라미로 표시를 해주어서 자신이 원하는 곳에 한 번의 터치로 보낼 수 있는 능력을 기르는 데에 의의를 둔다. 트래핑 연습에서의 점수는 시간을 기준으로 이용하여 사용자의 트래핑 능력을 측정한다. 수치화는 각각 훈련 별로, 주차별로 저장, 출력하여 사용자가 어느 훈련을 얼마큼 했는지, 어느 훈련이 부족한지 알 수 있게 하고, 주차별로 훈련계획을 세울 수 있도록 출력한다.

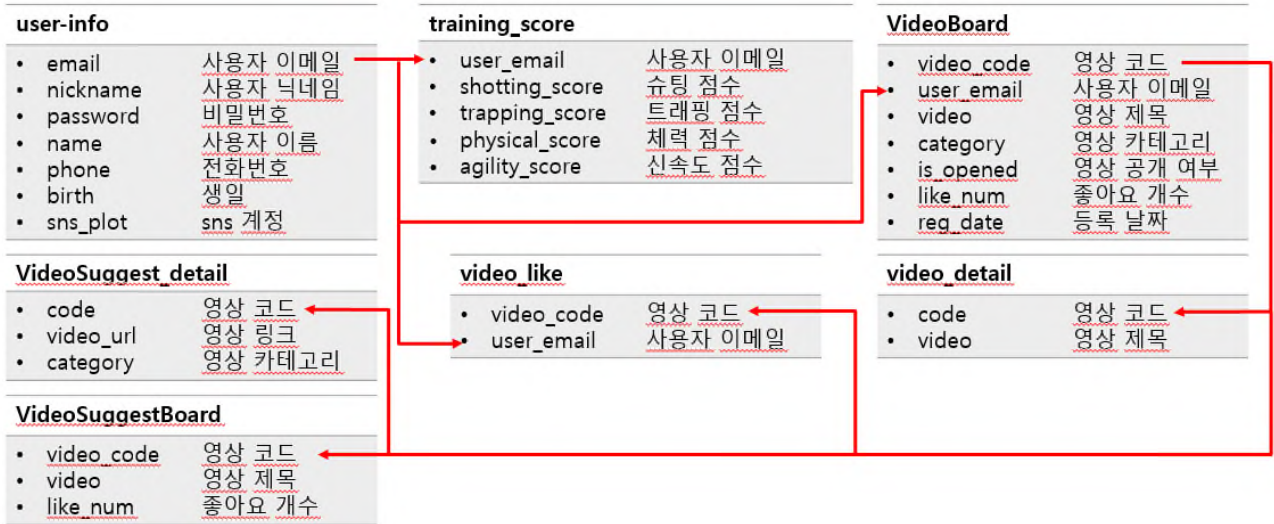
(3) 딥러닝

딥러닝은 다음의 방식으로 설계하였다. 먼저 일반적인 FCNN과 CNN의 가장 큰 차이점을 알아야 한다. CNN은 이미지 인식에 있어 이미지의 모든 픽셀을 분석하지 않고 지역적, 공간적 상관관계를 고려한 학습을 한다. 따라서 효과적으로 Data-representation을 수행하고 이를 통해 분류 작업에 있어 높은 성능을 보인다. 매트랩을 활용한 CNN으로 좋은 자세의 데이터를 획득한다. 데이터 전처리는 유튜브를 통해 얻은 슈팅 방법별 영상을 다운로드, 편집하여 데이터를 확보하였다. 모델 설계는 분석을 위해 최소한의 CNN 구조를 2conv 계층, 1fcnn 계층으로 구성하였다. 학습 과정으로는 over fitting을 방지하기 위해 early stopping을 하였다.

(4) 데이터베이스

데이터베이스의 설계는 다음과 같다. 회원 정보 테이블은 닉네임을 기본, 외래키로 지정하고 이름 핸드폰 생년월일 이메일 sns 회원 번호를 스키마로 지정한다. 사용자 영상 테이블은 영상번호를 기본 키로 회원 번호+닉네

임을 외래키로 가지며 사용자 영상 날짜 사용자 능력(수치화)-숫, 피지컬, 트래핑을 스키마로 지정한다. 슈팅 테이블은 영상 이름을 기본 키로 숫 이름 영상(URL), 추천 능력치를 스키마로 지정하고 코멘트 테이블은 코멘트 번호를 기본 키, 영상번호를 외래키, 닉네임을 스키마로 좋아요 테이블은 좋아요 번호를 기본 키, 영상번호를 외래키, 닉네임을 스키마로 지정한다.



<그림 18> 데이터 베이스 설계

```

CREATE TABLE `user_info` (
  `nickname` varchar(20) NOT NULL,
  `name` char(30) NOT NULL,
  `phone` int(11) NOT NULL,
  `birth` date NOT NULL,
  `email` varchar(80) NOT NULL,
  `sns_plot` varchar(20) DEFAULT NULL,
  PRIMARY KEY (`nickname`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

CREATE TABLE `VideoBoard` (
  `video_id` int(11) NOT NULL AUTO_INCREMENT,
  `user_nickname` varchar(10) NOT NULL,
  `video` char(60) DEFAULT NULL,
  `like_num` int(5) DEFAULT '0',
  `reg_date` datetime DEFAULT CURRENT_TIMESTAMP,
  PRIMARY KEY (`video_id`),
  KEY `user_nickname` (`user_nickname`),
  CONSTRAINT `VideoBoard_ibfk_1` FOREIGN KEY (`user_nickname`) REFERENCES `user_info` (`nickname`)
) ENGINE=InnoDB AUTO_INCREMENT=70 DEFAULT CHARSET=utf8;

CREATE TABLE `video_detail` (
  `id` int(11) NOT NULL,
  `video` varchar(255) NOT NULL,
  KEY `id` (`id`),
  CONSTRAINT `video_detail_ibfk_1` FOREIGN KEY (`id`) REFERENCES `VideoBoard` (`video_id`) ON DELETE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

CREATE TABLE `video_like` (
  `video_id` int(11) NOT NULL,
  `user_nickname` varchar(20) NOT NULL,
  KEY `video_id` (`video_id`),
  KEY `user_nickname` (`user_nickname`),
  CONSTRAINT `video_like_ibfk_1` FOREIGN KEY (`video_id`) REFERENCES `VideoBoard` (`video_id`),
  CONSTRAINT `video_like_ibfk_2` FOREIGN KEY (`user_nickname`) REFERENCES `user_info` (`nickname`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

CREATE TABLE `VideoSuggestBoard` (
  `video_id` int(11) NOT NULL AUTO_INCREMENT,
  `video` char(60) DEFAULT NULL,
  `like_num` int(5) DEFAULT '0',
  PRIMARY KEY (`video_id`)
) ENGINE=InnoDB AUTO_INCREMENT=70 DEFAULT CHARSET=utf8;

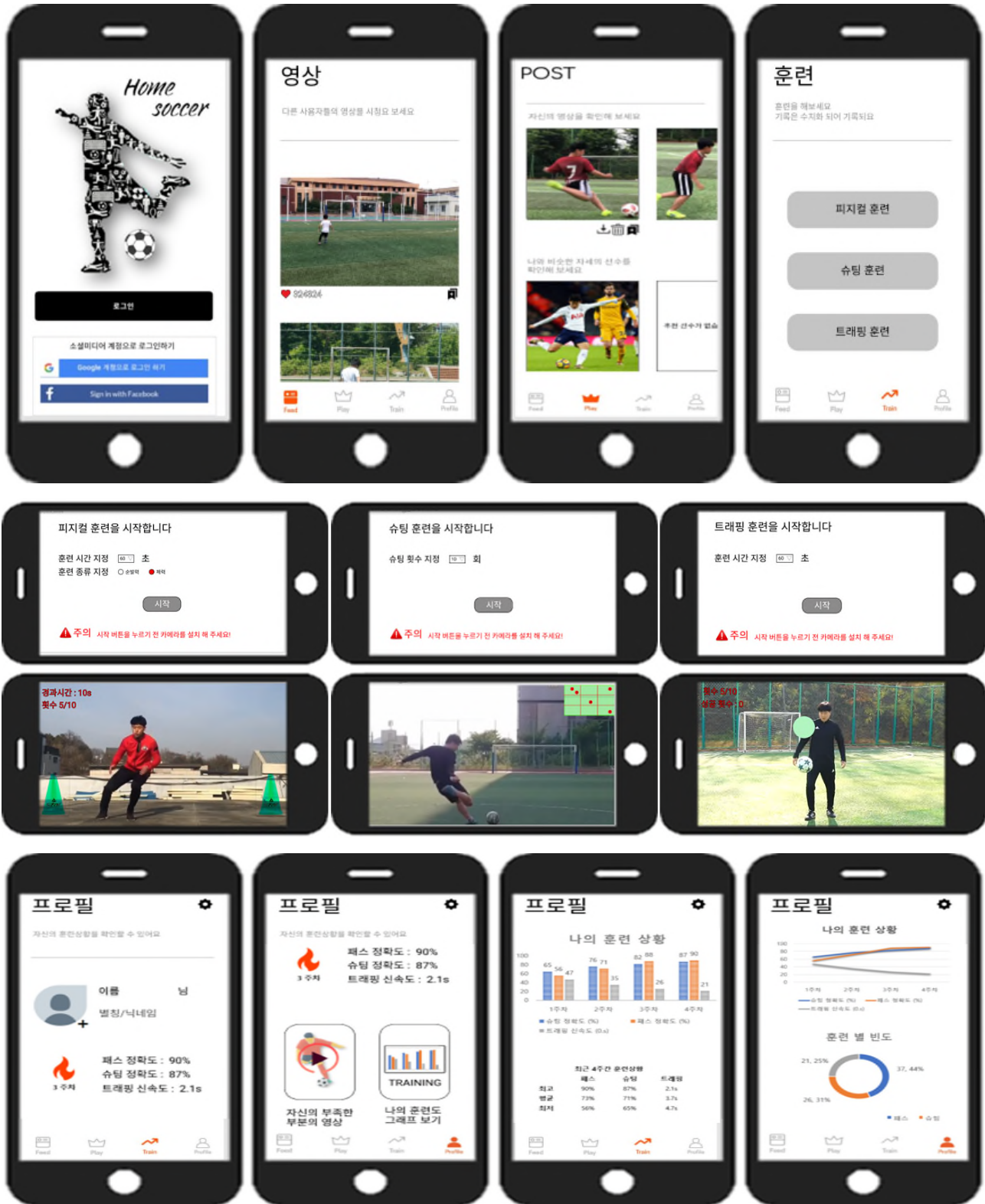
CREATE TABLE `VideoSuggest_detail` (
  `id` int(11) NOT NULL,
  `videourl` varchar(255) NOT NULL,
  `category` varchar(30) not null,
  KEY `id` (`id`),
  CONSTRAINT `VideoSuggest_detail_ibfk_1` FOREIGN KEY (`id`) REFERENCES `VideoSuggestBoard` (`video_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

```

<그림 19> 데이터 베이스 DDL

(5) UI

다음으로 <그림 20>은 UI의 상세 설계이다.



<그림 20> UI 설계

(6) 증강현실

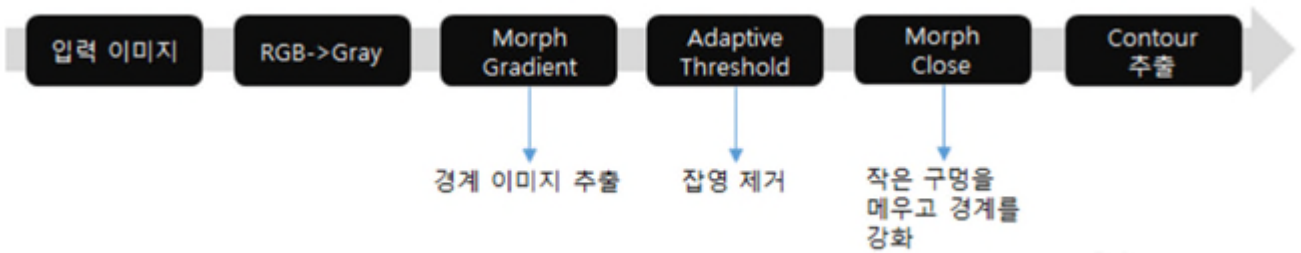
증강현실은 피지컬 훈련과 트래핑 훈련에 사용이 된다. 피지컬 훈련에서는 고깔과 땀틀을 나타내주고 고깔을 건들 경우에는 점수를 얻게 하고 땀틀에 걸렸을 경우 점수를 감점하는 방향으로 설계하였다. 또 트래핑 훈련에는 원을 무작위로 화면에 출력하여 공을 그 위치로 보내도록 설계하였다. 환경은 Unity에서 제공하는 pose tracking driver와 ARCore를 이용하여 설계를 하였으나 모션 인식의 경우 ios의 최신 기기 몇 종만 지원하여 안드로이드 스튜디오에서 ARCore를 사용하는 것으로 교체하였다.

2.5 Prototype 구현

(1) 인식

인식의 경우 공 궤적인식과 골격 인식, 골대 인식, 거리 인식이 있다.

골격 인식의 경우 연산속도의 저하와 영상의 프레임 저하를 위해 골격 다중인식에서 싱글 인식으로 연산 처리량을 줄였으며 한 프레임마다 인식하게 하여 프레임을 증대시켰다. 더 작은 모델도 인식 가능하도록 데이터 셋을 추가하였으며 그에 대한 과부하 방지로 멀티 쓰레딩을 이용했다. 4개의 쓰레드를 연결 4개씩, 각 쓰레드의 서브 쓰레드가 스케줄이 가능하도록 알고리즘을 추가하고, 멀티 쓰레딩으로 동시 수행 가능하도록 하였다.



<그림 21> 공 이미지 라벨링 - 잡영 제거 과정

공인식의 경우 하프변환을 이용하여 Opencv 라이브러리를 이용하려고 하였지만, 공이 작아지면 인식이 불명확해지는 부분이 있어 공의 이미지를 덤퍼닝으로 훈련 시켜 개선 시키고자 하였다. 공의 이미지는 구글의 이미지와 공인구 데이터베이스 사이트로부터 BeautifulSurp를 이용해, 크롤링하였다. 궤적에 있어 공이 찌그러지는 것도 인식하기 위해 이미지를 변형하여 총 57000장을 모아 텐서플로우의 keras를 이용하여 합성곱 신경망을 구축하여 Epoch를 20, batch를 600으로 두어 학습률을 높였고 85%가 넘는 훈련된 모델을 얻었다. 이 학습된 모델을 기반으로 가중치 파일을 객체 인식을 위한 Yolo 알고리즘의 훈련모델로 전환하고 Yolo망을 구축했다. 객체 인식을 위해 각 이미지마다 라벨링을 하기 위한 형태학적 변환을 이용하였다. 공의 이미지에 가우시안 블러를 진행한다. 회색조로 변환한 Gray Scale 이미지에 임계(threshold)를 적용한다. 즉, <그림 21>의 과정대로 특정 값인 임계값 이하의 값을 갖는 픽셀을 검정색으로 변환하고, 임계값 이상의 값을 갖는 픽셀을 흰색으로 변환해 흑백 이미지를 얻는다. 불필요한 영역 등 잡영(noise)을 제거하는 효과가 있기 때문에 사물을 탐지할 때 효과적으로 Contour를 추출할 수 있다. 하지만 실제 이미지에는 잡영이 생각보다 많기 때문에 다음과 같은 과정을 거쳐 잡영을 최대한 제거해야 한다. Contour 추출 과정에 몇 가지 과정이 더 들어갔다. Morph Gradient는 사물의 테두리를 더 정확하게 추출할 수 있게 이미지를 처리하는 과정이고, Morph Close는 Contour의 영역을 찾는 과정이다. Morph Gradient는 사물의 테두리에 팽창(dilation)을 적용한 결과와 침식(erosion)을 적용한 결과의 차이로 테두리(외곽선)를 더 정확하게 추출하는 과정이다. 커널의 크기를 2 x 2픽셀로 하였고, 그 다음 findContours를 이용하여 테두리를 검출하였다. 이 결과로 얻은 공의 테두리에 라벨링, 바운딩하여 이미 훈련된 모델을 통한 전이학습을 통한 훈련을 진행하였다. 이를 통해 인신을 진행하려 하였으나, 학습 데이터를 사용하고 자하는 포맷으로 변환하는 과정에서 프로젝트와 호환되지 않는다는 문제를 발견하여 이전의 인식 방법으로 다시금 전환하였다.

피드백을 주는 것으로 수정하였다. 자세한 자세 인식은 공을 인식해 공의 중심좌표를 찾아내고 공의 중심좌표와 양쪽 발목의 좌표의 거리가 일정 거리 이하로 줄어들면 그때부터 리스트에 공과 몸의 위치의 좌표를 기억한다. 그 후 슈팅이 끝난 것을 공의 위치로 특정해 낸 다음 아까 저장한 위치 좌표를 불러와 슈팅하기 전 디딤발 딛는 순간과 공에 한쪽 발목이 만난 순간, 슈팅하고 난 직후의 순간을 뽑아내고 이를 이용해 디딤발과 공의 거리, 발목과 무릎의 각도, 골반과 무릎의 각도, 손목과 팔꿈치 어깨를 이용한 상체의 기울기를 계산하고 올바른 자세의 각도를 입력한 값과 비교를 한다. 그 후 각도가 얼마나 차이가 나는지 디딤발의 위치에 대해 피드백을 해준다.

표 1. 운동학적 변인

변 인	단 위	정지완공	굴러가는공	굴러오는공	F-value	유의도
공의 투사 절대속도	[m/sec]	29.78±0.81	27.85±1.05	28.83±1.87		
공의 투사각도	[°]	4.98±0.48	5.35±0.69	4.90±0.71		
보 폭	[m]	1.75±0.05(0.97)	1.79±0.88(0.99)	1.76±0.06(0.98)	21.47	0.042 *
공중심에서 무릎 까지 수평거리	[m]	0.12±0.05	0.13±0.08	0.10±0.07	0.24	0.843
공중심에서 무릎 중심까지 수평거리	[m]	0.05±0.07	0.12±0.00	0.06±0.06	18.24	0.038 *
임팩트 순간 발끝 절대속도	[m/sec]	20.92±0.60	20.47±0.56	20.68±0.52		
임팩트순간 차는 다리의 발목각도	[°]	154.01±8.4	136.61±6.2	144.86±6.8	2.47	0.82
임팩트순간 차는 다리의 무릎각도	[°]	131.68±13.3	126.31±19.9	122.77±10.1	1.01	0.40
임팩트순간차지 다리의 발목각도	[°]	82.30±4.0	92.80±13.2	90.14±7.8	3.41	0.93
임팩트순간차지 다리의 무릎각도	[°]	131.30±8.9	134.30±7.8	132.28±10.4	1.59	0.52
임팩트순간 고관절의 상대각(θ)	[°]	22.07±7.03	19.39±8.12	14.25±4.37	2.14	0.47
임팩트순간 고관절의 상대각(θ)	[°]	-25.46±7.03	-11.21±4.08	-14.31±13.19	4.73	0.82
임팩트순간 고관절의 상대각(θ)	[°]	25.86±17.68	23.06±10.91	27.28±14.56	5.24	0.26

<그림 25> 자세 피드백 - 운동학적 변인

(3) 데이터베이스

데이터베이스의 경우 상세 설계에서 계획했던 스키마 설계를 서버 데이터베이스에 등록하여 구현을 완료하였고, 최종적으로는 타 SNS 계정을 등록하여 회원가입이 가능하도록 하려 하며, 현재는 사용자 정보의 nickname을 아이디처럼 키로 활용하여 로그인 및 회원가입 기능을 대체하고 있다. 서버 데이터베이스와의 데이터는 php 파일을 경유하여 어플리케이션과 주고받으며 동영상의 업로드 역시 같은 방식으로 진행하였다. 서버는 유동 퍼블릭 IP를 사용하였으나, 최종 사용 시 IP의 잦은 변동은 불편이 많을 것이라 생각되어 고정 IP를 할당하였다. 다중 사용자의 S3 스토리지 접근 및 권한 관리를 위해 Cognito를 활용하여 불특정 다수의 사용자들에게 필수적인 자료 업로드 및 다운로드 권한만을 주도록 하였다. 스토리지에 보관된 영상에 사용자들이 접근하는 방식에 대해선 현재 방식을 논의 중이며 데이터 이동 소요가 많은 다운로드보다 URL을 통한 일시적 영상 스트리밍을 고려 중이다.

(4) 영상 기록

camera2 라이브러리와 MediaRecorder를 이용하여 슈팅 연습을 시작하고 공을 차는 순간부터 동영상 녹화가 시작되고, 공이 골대에 들어간 순간 녹화가 종료된다.

세부적인 순서는 슈팅 연습을 시작하면 openCamera 메소드에 의해 카메라가 켜지고 camera manager를 통해 여러 카메라 정보를 얻는다. 그 뒤, camera device를 열고 cameracapturesession 메소드를 이용하여

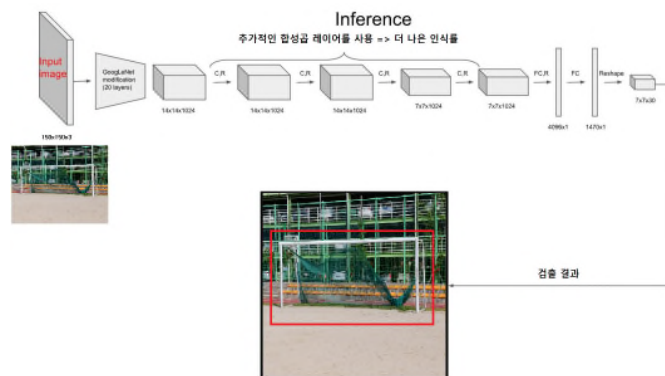
surfaceview로부터 영상의 미리 보기를 할 수 있도록 설정해준다. 다음으로 직접 정의한 startrecording과 stoprecording을 이용하여 영상을 녹화하고 정지해줄 수 있다. startrecording은 녹화를 시작해주는 메소드로서, 먼저 MediaRecord 객체를 이용하여 오디오와 영상의 저장경로나 인코더, 소스 등을 지정해주고 prepare 함수를 이용하여 녹화 시작의 준비를 해준다. 다음으로 MediaRecorder 객체의 start 함수를 이용하여 녹화를 시작할 수 있다. stoprecording은 MediaRecorder 객체의 stop 함수를 통해 녹화를 멈추고 지정한 저장경로에 저장해준다. 마지막으로 release 함수를 이용하여 영상을 다시 녹화할 수 있도록 메모리 해제한다.

(5) 증강현실

한 달간 Unity의 pose track driver와 3d human body, vuforia, ARFoundation, XRcore를 이용하여 작성했었지만, 해당 여러 기능 중 사용하려는 핵심기능들이 ios의 최신 기기 몇 종에서만 제한적으로 사용할 수 있다는 것을 알고 안드로이드 스튜디오의 ARCore를 이용하여 화면에 AR을 출력해주고 모션 인식에 사용했던 Posenet을 이용하여 관절을 인식하고 해당 객체를 터치하거나 뛰어넘는 방식으로 연습이 가능할 수 있도록 진행 중이다.

(6) 딥러닝

슈팅 연습 등에 사용할 공과 골대의 인식을 위해 각각의 객체를 DeepLearning으로 검출하고자 하였다. 데이터 라벨링 초기에는 Faster-Rcnn을 사용하여 딥러닝을 진행하려 했지만, 축구의 경우 공의 움직임이 매우 빠른 스포츠이기 때문에 인식의 정확도를 약간 희생하더라도 빠르고 지속적인 인식이 필요했다. 이로 인해 초당 처리 속도가 상대적으로 뛰어난 Yolo를 사용하기로 하였다. 추가로 Yolo는 F-Rcnn에 비해 훨씬 적은 False-Positive(목표 객체가 아닌 대상을 목표 객체로 판단하는 오류)를 보여주어 적은 background error가 발생한다는 장점이 있다.



<그림 26> Yolo 추론 과정

Yolo 는 빠른 객체 검출을 위한 딥러닝 기법이다. 이미지를 분할 하지 않고 한 번에 인식하기 때문에, 각 객체에 대한 검출 오류가 상대적으로 적다.

전체적인 추론 과정은 다음과 같다.

1. 입력 이미지를 S x S 그리드로 나눈다.
2. 각각 grid cell은 B개의 bounding box와 그에 대한 confidence score를 갖는다. (객체가 없다면 score=0)
3. 각 grid cell은 C개의 conditional class probability를 갖는다.

4. 각 bounding box와 x, y, w, h, confidence로 구성된다.
5. grid cell의 여러 bounding box 중 ground-truth box와 IOU가 가장 높은 box를 predictor로 설정한다.
6. 객체가 존재하는 grid cell을 분류한다.
7. ground-truth box의 중심점이 일부 grid cell 내부에 위치하면 해당 cell에 객체가 있다고 인식한다.

```

[11] 0.0000 0.0000
avg_output = 518123
Loading weights from darknet53.conv.74
mean is trained 8 images (0.418-batches_84)
Images loaded 75 images from weights-file
Learning Rate: 0.001, Momentum: 0.9, Decay: 0.0005
Resizing, random_crop = 1.48
Creating 6 permanent cpu-threads
*** x 655

Loaded 0.00000 seconds
[1] (see loss, Normalizer: (iou: 0.75, cls: 1.00) Region 82 Avg (IOU: 0.278381,
IOU: 0.278381), Class: 0.473385, Obj: 0.507450, No Obj: 0.484461, SR: 0.250000,
FSR: 0.000000, count: 4, class_loss = 267.882525, iou_loss = 1.568355, total_
loss = 269.471824
[2] (see loss, Normalizer: (iou: 0.75, cls: 1.00) Region 94 Avg (IOU: 0.000000,
IOU: 0.000000), Class: 0.000000, Obj: 0.000000, No Obj: 0.531933, SR: 0.000000,
FSR: 0.000000, count: 1, class_loss = 1380.438252, iou_loss = 0.000000, total_
loss = 1380.438252
[3] (see loss, Normalizer: (iou: 0.75, cls: 1.00) Region 106 Avg (IOU: 0.000000,
IOU: 0.000000), Class: 0.000000, Obj: 0.000000, No Obj: 0.503127, SR: 0.000000,
FSR: 0.000000, count: 1, class_loss = 4681.618652, iou_loss = 0.000000, total_
loss = 4681.618652
[4] (see loss, Normalizer: (iou: 0.75, cls: 1.00) Region 82 Avg (IOU: 0.325507,
IOU: 0.195607), Class: 0.607425, Obj: 0.421353, No Obj: 0.484568, SR: 0.250000,
FSR: 0.000000, count: 4, class_loss = 267.147461, iou_loss = 1.756327, total_
loss = 268.904388
[5] (see loss, Normalizer: (iou: 0.75, cls: 1.00) Region 94 Avg (IOU: 0.000000,
IOU: 0.000000), Class: 0.000000, Obj: 0.000000, No Obj: 0.552840, SR: 0.000000,
FSR: 0.000000, count: 1, class_loss = 1380.080533, iou_loss = 0.000000, total_
loss = 1380.080533
[6] (see loss, Normalizer: (iou: 0.75, cls: 1.00) Region 106 Avg (IOU: 0.000000,
IOU: 0.000000), Class: 0.000000, Obj: 0.000000, No Obj: 0.502885, SR: 0.000000,
FSR: 0.000000, count: 1, class_loss = 4671.710338, iou_loss = 0.000000, total_
loss = 4671.710338
[7] (see loss, Normalizer: (iou: 0.75, cls: 1.00) Region 82 Avg (IOU: 0.230391,
IOU: 0.171108), Class: 0.614827, Obj: 0.402561, No Obj: 0.484345, SR: 0.000000,
FSR: 0.000000, count: 3, class_loss = 268.131805, iou_loss = 1.294403, total_
loss = 269.426208
[8] (see loss, Normalizer: (iou: 0.75, cls: 1.00) Region 94 Avg (IOU: 0.269779,
IOU: 0.208773), Class: 0.431487, Obj: 0.457452, No Obj: 0.552123, SR: 0.000000,
FSR: 0.000000, count: 1, class_loss = 1381.751709, iou_loss = 1.403546, total_
loss = 1383.161255
[9] (see loss, Normalizer: (iou: 0.75, cls: 1.00) Region 106 Avg (IOU: 0.000000,
IOU: 0.000000), Class: 0.000000, Obj: 0.000000, No Obj: 0.502936, SR: 0.000000,
FSR: 0.000000, count: 1, class_loss = 4684.998047, iou_loss = 0.000000, total_
loss = 4684.998047

```

<그림 27> YOLO를 이용한 모델 훈련

(7) UI

피그마를 이용하여 모든 디자인 설계를 완성하고 와이어 프레임 구축하여 메인페이지, 로그인 페이지, 가입하기 페이지, 아이디/비밀번호 찾기 페이지, 영상 페이지, 포스트 페이지, 프로필 페이지, 훈련 페이지, 각 훈련 시작 전 페이지, 설정 페이지, 게시물 관리 페이지, 나의 정보 관리 페이지, 갤러리에서 사진 불러오기, 네비게이션 하단 바, 평가하기 페이지, 프로그램 정보 페이지, 공지사항 페이지 등을 완료하였고 각 페이지 별 여러 버튼들의 이벤트와 intent 설정을 완료하였으면 각각의 모듈들을 합병할 때 각 버튼의 이벤트에 맞게 구축만 하면 완료되는 상황이다.

2.5 시험/테스트 결과

YOLO를 이용한 훈련으로 윈도우의 정지영상에서는 객체 위치추정의 인식율이 꽤 높았으나 훈련된 모델 파일을 android 환경에 적합한 파일로 변환하는 과정에서 정확도가 손실되는 상황이 발생하여 여러 방법을 시도한 결과 Opencv 라이브러리를 이용하여 HSV 색상과 허프 변환을 이용한 공 위치 추적방법을 개선하기로 하였다. HSV 색상의 범위를 증대시켜 공의 회전과 공이 멀어질 때 프레임 상 색상 변화에서도 인식 가능케 하였다. 다음, ROI를 이용하였다. 공을 인식하게 되면 추적에 성공한 구역의 각각의 RGB 값을 받아 평균을 내어서 저장한 다음 추적 대상을 놓쳐버린 시점에서 마지막으로 추적 성공한 구역의 상단 3방향을 탐색, 각각 RGB 값을 평균값을 비교하여 허프 변환으로는 추적이 불가능한 고속 이동 물체에 대해서도 보다 정확한 추적이 가능하게 하였다. 객체 위치 추적을 위하여 불필요한 연산이 증가하게 되었고, 이를 줄이기 위해 추적에 이용되는 프레임의 수를 줄여 연산을 줄여 이전보다 효율을 높일 수 있었다.



<그림 28> 공 위치 추적과 궤도 추출



<그림 29> 공 위치 추적과 사람 자세 추정

2. Cord / Demo

(1) 공 검출 메소드

```
Imgproc.HoughCircles(matGaussian, circle, Imgproc.HOUGH_GRADIENT, 2.0, 20, 70, 30, 5, 1000);

if (circle.cols() > 0) { // 검출
    if (operate != 5) { // 동작 하지않았다면
        operate++; // 1회성
        for (int x = 0; x < Math.min(circle.cols(), 1); x++) {
            double circleVec[] = circle.get(0, x);
            if (circleVec == null) { break; }
            center = new Point((int) circleVec[0], (int) circleVec[1]);
            Log.i("center_r", center.toString());
            circleVec_save[0] = circleVec[0];
            circleVec_save[1] = circleVec[1];
            circleVec_save[2] = circleVec[2];
            int radius = (int) circleVec[2];
            x_newSaver = circleVec_save[0].intValue();
            y_newSaver = circleVec_save[1].intValue();
            if (frame < 11) { w_newSaver=20; }
            else{ w_newSaver=10; }
            Imgproc.circle(matInput, center, radius, new Scalar(0, 0, 0), 2);
            line_list.add(0, center);
        }
    }
    if(operate==5){
        if (x_newSaver != 0 && y_newSaver != 0 && w_newSaver != 0) { // 검출확인
            int[] avg = trackingArea(bitmap, x_newSaver, y_newSaver, w_newSaver); // 구역 평균 계산
            if (R_old_avg == 0 && G_old_avg == 0 && B_old_avg == 0) { // 처음
                R_old_avg = avg[0];
                G_old_avg = avg[1];
                B_old_avg = avg[2];
            } else { // 처음이 아니라면
                FindDifference(
                    bitmap,
                    avg[0],
                    avg[1],
                    avg[2],
                    C_confirmed);
            }
        }
    }
}
```

(2) 구역 내 RGB 정보 추출 메소드

```
int[] trackingArea(Bitmap bitmap, int x, int y, int w){
    int[][] rgb=new int[2*w][2*w];
    int R=0,G=0,B=0;
    Double cnt=0.0;
    int[] avg=new int[3];
    avg[0]=0;
    avg[1]=0;
    avg[2]=0;
    for(int i=0; i<(2*w);i++){
        for(int j=0;j<(2*w);j++){
            cnt+=1;
            rgb[i][j] = bitmap.getPixel(x-w+i,y-w+j); //원하는 좌표값 입력
            R += Color.red(rgb[i][j]); //red값 추출
            G += Color.green(rgb[i][j]); //green값 추출
            B += Color.blue(rgb[i][j]); //blue값 추출
        }
    }
    avg[0]=(int) (R/cnt);
    avg[1]=(int)(B/cnt);
    avg[2]=(int) (G/cnt);
    Log.i("iu", (avg[0])+" a "+(avg[1])+" "+(avg[2]));
    return avg;
}
```

(3) 상단 3방향 구역 정보 수집 메소드

```
void Check_bound(Bitmap bitmap, int x, int y, int w){
    int[] avg;
    x_LSaver=0; y_LSaver=0;
    x_MSaver=0; y_MSaver=0;
    x_RSaver=0; y_RSaver=0;
    if(x-3*w>80 && y-3*w>0 && x+3*w<560) { // 바운드 검사
        x_LSaver = x - 2 * w; y_LSaver = y - 2 * w;
        avg=trackingArea(bitmap,x_LSaver,y_LSaver,w);
        R_LAvg=avg[0]; G_LAvg=avg[1]; B_LAvg=avg[2];
        x_MSaver = x; y_MSaver = y - 2 * w;
        avg=trackingArea(bitmap,x_MSaver,y_MSaver,w);
        R_MAvg=avg[0]; G_MAvg=avg[1]; B_MAvg=avg[2];
        x_RSaver = x + 2 * w; y_RSaver = y - 2 * w;
        avg=trackingArea(bitmap,x_RSaver,y_RSaver,w);
        R_RAvg=avg[0]; G_RAvg=avg[1]; B_RAvg=avg[2];
    }
    else{
        R_LAvg=0; G_LAvg=0; B_LAvg=0;
        R_MAvg=0; G_MAvg=0; B_MAvg=0;
        R_RAvg=0; G_RAvg=0; B_RAvg=0;
        if(y-3*w>0){
            x_MSaver = x; y_MSaver = y - 2 * w;
            avg=trackingArea(bitmap,x_MSaver,y_MSaver,w);
            R_MAvg=avg[0]; G_MAvg=avg[1]; B_MAvg=avg[2];
            if(x_newSaver-3*w>80){
                x_LSaver = x - 2 * w; y_LSaver = y - 2 * w;
                avg=trackingArea(bitmap,x_LSaver,y_LSaver,w);
                R_LAvg=avg[0]; G_LAvg=avg[1]; B_LAvg=avg[2];
            }
            if(x_newSaver+3*w<560){
                x_RSaver = x + 2 * w; y_RSaver = y - 2 * w;
                avg=trackingArea(bitmap,x_RSaver,y_RSaver,w);
                R_RAvg=avg[0]; G_RAvg=avg[1]; B_RAvg=avg[2];
            }
        }
    }
}
```

(4) ROI 구역 구분 공 추적 메소드

```
void FindDifference(Bitmap bitmap, int R_avg, int G_avg, int B_avg, int confirmed){
    Point center;
    int R_compareValue=Math.abs(R_old_avg-R_avg); //R_이질감 확인
    int G_compareValue=Math.abs(B_old_avg-G_avg); //G_이질감 확인
    int B_compareValue=Math.abs(G_old_avg-B_avg); //B_이질감 확인
    if(frame<8) {
        if(R_compareValue>30||G_compareValue>15||B_compareValue>15){ //비교
            C_checked=true;
            Check_bound(bitmap, x_newSaver,y_newSaver,w_newSaver);
            if(confirmed==1){
                x_newSaver=x_LSaver;
                y_newSaver=y_LSaver;
                R_old_avg=R_LAvg;
                G_old_avg=G_LAvg;
                B_old_avg=B_LAvg;
                L_checked=true;
                center = new Point(x_newSaver, y_newSaver);
                line_list.add(0, center);
            }
            if(confirmed==2){
                x_newSaver=x_MSaver;
                y_newSaver=y_MSaver;
                R_old_avg=R_MAvg;
                G_old_avg=G_MAvg;
                B_old_avg=B_MAvg;
                M_checked=true;
                center = new Point(x_newSaver, y_newSaver);
                line_list.add(0, center);
            }
            if(confirmed==3){
                x_newSaver=x_RSaver;
                y_newSaver=y_RSaver;
                R_old_avg=R_RAvg;
                G_old_avg=G_RAvg;
                B_old_avg=B_RAvg;
                R_checked=true;
                center = new Point(x_newSaver, y_newSaver);
                line_list.add(0, center);
            }
        }
    }
}
```

```

}
else if(frame<15){
    if(R_compareValue>16||G_compareValue>9||B_compareValue>9){ //비교
        C_checked=true;
        Check_bound(bitmap, x_newSaver,y_newSaver,w_newSaver);
        if(confirmed==1){
            x_newSaver=x_LSaver; y_newSaver=y_LSaver;
            R_old_avg=R_LAvg; G_old_avg=G_LAvg; B_old_avg=B_LAvg;
            L_checked=true;
            center = new Point(x_newSaver, y_newSaver);
            line_list.add(0, center);
        }
        if(confirmed==2){
            x_newSaver=x_MSaver; y_newSaver=y_MSaver;
            R_old_avg=R_MAvg; G_old_avg=G_MAvg; B_old_avg=B_MAvg;
            M_checked=true;
            center = new Point(x_newSaver, y_newSaver);
            line_list.add(0, center);
        }
        if(confirmed==3){
            x_newSaver=x_RSaver; y_newSaver=y_RSaver;
            R_old_avg=R_RAvg; G_old_avg=G_RAvg; B_old_avg=B_RAvg;
            R_checked=true;
            center = new Point(x_newSaver, y_newSaver);
            line_list.add(0, center);
        }
    }
}
else{
    if(R_compareValue>9||G_compareValue>6||B_compareValue>6){ //비교
        C_checked=true;
        Check_bound(bitmap, x_newSaver,y_newSaver,w_newSaver);
        if(confirmed==1){
            x_newSaver=x_LSaver; y_newSaver=y_LSaver;
            R_old_avg=R_LAvg; G_old_avg=G_LAvg; B_old_avg=B_LAvg;
            L_checked=true;
            center = new Point(x_newSaver, y_newSaver);
            line_list.add(0, center);
        }
        if(confirmed==2){
            x_newSaver=x_MSaver; y_newSaver=y_MSaver;

```

```

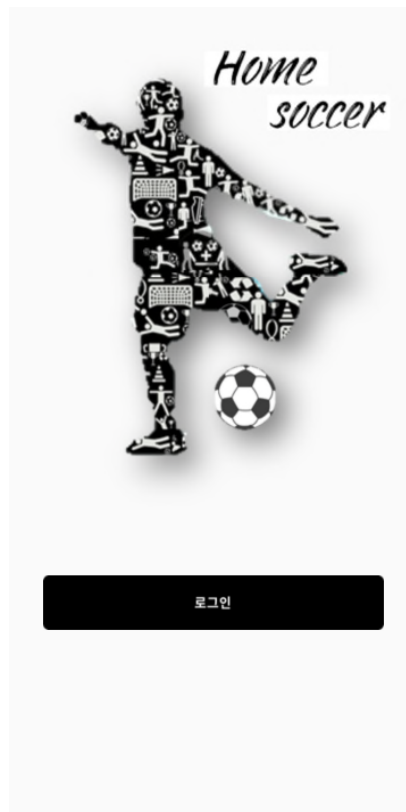
    R_old_avg=R_MAvg; G_old_avg=G_MAvg; B_old_avg=B_MAvg;
    M_checked=true;
    center = new Point(x_newSaver, y_newSaver);
    line_list.add(0, center);
}
if(confirmed==3){
    x_newSaver=x_RSaver; y_newSaver=y_RSaver;
    R_old_avg=R_RAvg; G_old_avg=G_RAvg; B_old_avg=B_RAvg;
    R_checked=true;
    center = new Point(x_newSaver, y_newSaver);
    line_list.add(0, center);
}
}
}
if(C_checked==true){
    if(R_LAvg!=0&&L_checked==false){
        L_checked=true;
        FindDifference(bitmap, R_LAvg, G_LAvg, B_LAvg,L_confirmed);
    }
    if(R_MAvg!=0&&M_checked==false){
        M_checked=true;
        FindDifference(bitmap, R_MAvg, G_MAvg, B_MAvg,M_confirmed);
    }
    if(R_RAvg!=0&&R_checked==false){
        R_checked=true;
        FindDifference(bitmap, R_RAvg, G_RAvg, B_RAvg,R_confirmed);
    }
}
frame++;
C_checked=false;
L_checked=false;
M_checked=false;
R_checked=false;
}

```

(6) 데모 영상

유튜브에 영상 등록

<https://youtu.be/ZDZcbXj722w?t=0>



<그림 37> 데모 영상 썸네일

III. 결론

(1) 연구 결과

본 졸업작품의 목적은 고비용, 시간 등의 제약을 받는 현 축구 연습 프로그램을 대체하여 개인이 다른 이의 도움 없이 슈팅 시의 자세 및 공의 궤적, 트래핑 능력, 기초적인 순발력 및 체력 등을 연습할 뿐만 아니라 해당 연습 영상을 추후 재확인하거나, 타인의 연습 영상을 확인할 수 있는 어플리케이션의 개발에 있다.

슈팅 자세를 교정해주기 위해서는 그에 맞는 자세의 각도와 슈팅 궤적을 통해 나타나는 킥의 종류를 파악해 올바른 각도와 알맞은 타점, 궤적을 알려줄 수 있어야 하고 이를 위해 공의 인식, 인체의 관절을 인식할 수 있어야 한다. 이에 본 졸작의 결과는 공 인식의 경우 opencv를 이용한 hsv색상 추적으로는 공이 작아지는 경우 인식을 할 수 없게 되었다. 그에 반해 CAMshift의 경우 공이 작아지는 경우에도 제대로 인식을 할 수 있었지만 느린 작동 속도 때문에 빠른 속도의 공을 인식하지 못하는 결과를 가져오게 되었다. 이러한 결과를 통해 공 인식의 경우 먼저 색상 인식을 한 후 색상 인식을 하지 못하는 경우에만 CAMshift로 인식할 수 있게 연동을 하였고 고속으로 이동하는 공을 추적하기 위하여 kalman-filter를 이용하였다. 자세 인식의 경우에는 tensorflow의 posenet을 이용해 관절을 인식한 후 관절의 각도를 축구 인사이드 페널티킥 동작 시 목표변화에 따른 하지 분절의 운동학적 분석의 논문 내용을 참고하여 자세 교정을 하였다. 이를 통해 올바른 자세에 맞춰 관절의 각도를 피드백해줄 수 있었고 공의 궤적을 나타내 줄 수 있게 되었다.

트래핑 훈련의 목적은 실전에서의 트래핑 능력 향상에 있다. 트래핑 능력의 향상은 반복적인 리프팅 훈련을 통해 이루어진다. 리프팅 훈련은 공을 지면에 떨어뜨리지 않고 연속해서 공을 차는 볼 리프팅(Ball lifting) 훈련이며 이러한 훈련을 진행하기 위해 opencv와 posenet을 이용하여 공과 머리 또는 다리의 위치를 파악하고 파악한 데이터를 사각형 충돌 알고리즘으로 공이 다리나 머리에 닿는지 확인해 주어서 이를 통해 연속적인 볼 리프팅을 가능하게 해준다. 이렇게 연속적인 볼 리프팅 훈련을 진행하면 실전에서 날아오는 공을 받을 때 쉽게 컨트롤 할 수 있는 트래핑 능력을 향상시킬 수 있도록 도와준다.

피지컬 훈련은 사용자가 혼자 피지컬 적 측면에서 순발력과 체력을 증진 시킬 수 있는 훈련을 제공한다. 순발력 훈련은 opencv와 posenet를 이용한 영상처리를 통하여 공의 위치에 따라 좌우로 움직이는 사람과 공의 위치를 파악하며 사각형 충돌 알고리즘에 의해 훈련을 진행한다. 이를 통해 사용자들이 순발력 훈련에 있어 공에 대한 집중력과 빠르게 수비를 할 수 있도록 증진 시켜준다. 체력 훈련은 opencv와 posenet를 장애물을 피해 점프하는 사람을 파악하여 훈련을 진행하게 된다. 이를 통해 사용자들의 체력 훈련에 있어 상대 팀의 테클을 피하고, 체력을 증진 시켜, 좀 더 지치지 않고 축구를 할 수 있도록 도와준다.

연습 영상의 경우 어플리케이션에서 서버를 거쳐 RDS의 MYSQL과 S3 스토리지에 각각 영상의 이름, 영상을 저장하며 어플리케이션에서 영상을 불러올 때는 서버를 거쳐 MYSQL의 영상 이름을 확인 후 S3에서 해당 영상을 다운로드하는 수순을 거친다. 이를 통해 영상 데이터의 저장 및 불러오는 기능을 수행할 수 있도록 하였다.

참고 문헌

[1] 앱과 DB연동 -

https://docs.aws.amazon.com/ko_kr/elasticbeanstalk/latest/dg/java-rds.html

[2] 외부 클라이언트와 DB서버 연결 -

https://docs.aws.amazon.com/ko_kr/AmazonRDS/latest/UserGuide/USER_VPC.html

[3] 딥러닝 라이브러리 언어별 종류 -

<https://aikorea.org/blog/dl-libraries>

[4] 딥러닝에 대해 -

<http://t-robotics.blogspot.com/2015/05/deep-learning.html#.XfOaHegzaUk>

[5] 트래킹 코드 -

<https://diy-project.tistory.com/96>

[6] 공 관련 메소드 -

<https://liveupdate.tistory.com/281>

[7] open cv ROI 추출 -

<https://yeolco.tistory.com/57>

[8] Haar-like feature 프로토타입 관련 -

2011_송자혜, 장연진_강원대학교 IT대학 전기전자공학부 전기전자공학 졸업논문

[9] Openpose라이브러리

<https://m.blog.naver.com/worb1605/221297566317>

[10] Imgproc.HoughCircles 함수 -

http://docs.opencv.org/2.4/doc/tutorials/imgproc/imgtrans/hough_circle/hough_circle.html

[11] 안드로이드 tensorflow_lite에 대해 -

<https://github.com/tensorflow/examples>

[12] 최소자승법에 대해 -

<https://m.blog.naver.com/PostView.nhn?blogId=moojigai07&logNo=120186757908&proxyReferer=https%3A%2F%2Fwww.google.com%2F>

[13] 어플리케이션 와이어프레임 figma -

<https://www.figma.com/>

[14] CNN에 대해 -

<https://untitledtblog.tistory.com/150>

[15] 세가지 축구 슈팅 동작의 운동학적 비교-

1999,한국체육학회지_진영완_동의대학교,최지영,신제민_연세대학교

[16] YOLO에 대해 -

<https://pjreddie.com/darknet/yolo/>