졸업 작품 보고서

SINCE 1947
서경대학교
SEOKYEONG UNIVERSITY

농구경기 행동분석모델

2024년 11월

서경대학교 컴퓨터공학과

고형석

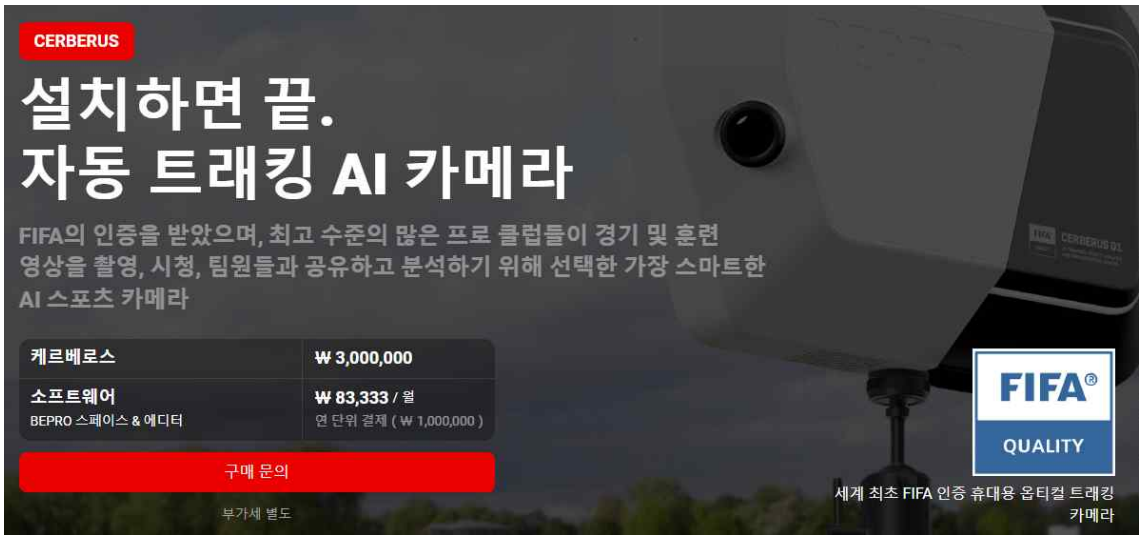
목 차

Ⅰ. 서론

(1). 제작 동기 및 목적

현재 다양한 스포츠 산업이 발달하고 있다. 많은 팀은 더 좋은 선수와 계약하면서 최고의 팀을 꾸리고 있다. 여기서 선수의 가치를 판단하기 가장 좋은 자료는 그 선수의 기록지를 보면 알 수 있다. 축구로 예를 들면 한 시즌에 몇 개의 골을 넣었는가, 패스의 정확도, 활동량은 어느 정도 인지 그 선수의 기록지를 통해 유추할 수 있다. 그렇다면 그 기록지는 현재 어떻게 만들어지고 있는가?

대부분의 스포츠 경우에는 여러 운동 경기 기록원이 직접 경기를 보면서 실시간으로, 혹은 녹화된 영상을 통해 수기로 작성한다고 한다. 한 선수씩 몇분 몇초에 무슨 행동을 하였고, 성공률이 어떻고, 언제 교체하였는지 일일이 적는다. 이러한 작업을 사람이 하기에 실수가 있을 수 있고, 직접 보면서 하기에 시간도 많이 소요된다. 물론 따로 사업 프로그램으로 도움을 받는 경우도 확인하였다. 하지만 이는 경기 전체의 카메라 데이터와 높은 비용으로 인해 프로 경기에서는 주로 사용하지만, 프로가 아닌 사람들은 이러한 프로그램을 사용하기에는 부담이 될 수밖에 없다.



(사진 1) B사의 AI 카메라와 소프트웨어 가격

이러한 상황에서 프로 경기만 사용할 수 있는 기록지를 프로가 아닌 사람도 기록지를 작성할 수 있게 프로그램을 만들고자 하였다. 자신이 직접 뛴 아마추어 경기, 혹은 친구들과 한 경기의 영상을 넣기만 해줘도 그 경기에서 자신이 한 기록에 대해 자동으로 만들어주는 그런 프로그램이다. 이런 프로그램을 만들게 되면 인재 양성과 실력 향상에 큰 도움을 줄 수 있을 것이다.

또한 프로 경기에서는 AI를 사용하지 않고 사람이 직접 기록지를 작성하는 경기에 경우 많은 자원이 들어가는 것을 볼 수 있다. (사진 2)를 보게 되면 프로 배구에서 한 경기에 나오는 기록지가 무려 1,000장에 달하고 여러 사람을 거치게 되면서 지금의 경기 결과 기록지가 나오게 되는 것이다. 이를 한 시즌으로 환산하게 되면 대략 A4용지 약 23만 장이 소모되는 셈인 것이다. 이러한 이면지를 선수 노트로 제작하여 팬들에게 선물하는 캠페인을 통해 이러한 문제를 극복하지만, 다른 각도로 문제를 극복하고자 이러한 프로젝트를 진행하였다.

(사진 2) V리그가 매 경기 버려지는 기록지를 활용해 업사이클 캠페인을 진행하는 사진

(2). 제작 내용

이러한 점들을 개선하고자 기획한 프로그램이 바로 '스포츠 자동 기록지' 프로그램을 만드는 것이다. 경기가 전체적으로 보이는 영상을 넣고서 자신이 원하는 사람의 경기 기록지를 기록하여 자신의 부족한 부분이 어느 점인지, 개선해야 할 지표는 어느 점인지 인지하면서 능력을 향상할 수 있게 하는 프로그램이 목표이다.

본 프로젝트의 방향성은 프로그램을 완성하는 것은 혼자 하기엔 비용적이나 역량적으로 힘들 것으로 보이기 때문에 필요한 프로그램의 모델 방향성을 말해준다. 선수의 각 행동에 대해 딥러닝을 통해 학습을 시키고 그것을 어떻게 프로그램에 적용을 시키는지 예시를 보여주면서 이 분야에 대한 방향성을 제시하며 발전하는 것을 기대하고 있다.

프로그램의 제작 순서는 다음과 같다. 먼저 화면에 보이는 사람을 Detection 하는 모델을 비교하여 찾고, 관절을 추출할 수 있는 Pose Estimation 기술을 활용하는 모델을 연구하고, 이 추출된 관절을 이용해서 현재 하는 동작을 유추하는 모델을 만든다. 이렇게 해서 만든 모델을 이용하여 경기 영상을 넣어서 어떤 식으로 진행

되는지 간단하게 프로그램을 작성하여 보여주는 것을 목표로 기획하였다.

이러한 프로그램 기능으로는 영상을 분석하면서 실시간으로 지정한 사람의 행동을 예측하는 것을 감시할 수 있게 보여주는 기능, 슛이나 패스 같은 동작하였을 때 행동 카운트가 누적되면서 영상 분석이 마무리되면 누적 기록이 나오는 기능, 분석한 영상의 기록이 잘 작동이 되었는지, 시간이 지나서도 확인 할 수 있도록 기록 영상을 저장하는 기능을 목표로 하고 진행하였다. 추가로 공을 가지고 있는 시간을 누적시키면서 공의 점유율을 표시하는 것도 기획하였다.

프로그램 기능	
실시간으로 동작 표시	공 소유에 따른 점유율 표시
분석 영상 저장	분석 종료 시, 총 기록 추출

(표 1) 기획한 프로그램의 기능

본 프로젝트에서 분석하고자 하는 스포츠는 농구 동작을 분석하는 것을 목표로 하였다. 그러한 이유는 여러 스포츠 중에서 카메라에 전체적인 선수를 보여주는 스포츠는 농구가 탁월하다고 생각했기 때문이다. 카메라에 안보인 선수의 행동을 분석하기에는 어렵다고 판단이 되었고, 이러한 영향을 덜 받을 수 있다고 생각이 되어 농구로 진행하였다. 이 프로젝트가 성공적으로 기록지를 추출할 수 있다면 다음에 다른 스포츠도 비슷한 방식으로 모델을 만들고 프로그램을 진행할 수 있다고 판단이 되었다.

(3). 관련 연구 및 동향

본 연구를 진행하기에 앞서 현재 논문은 어떤 방식으로 진행하는지 분석하였다. 비디오 영상에서 2차원 자세 추정과 LSTM 기반의 행동 패턴 예측 알고리즘(최지호, 황규태, 이상준 2022)에서는 영상 기반 재활 운동에서 환자의 동작 의도를 추정하는 방법을 제안하고 있다. 관절 추출 방법으로 OpenPose를 활용하여 2D 관절 좌표를 추정하였고, LSTM 기반의 Intention Network에 입력되어 동작을 "앉기", "서기", "걷기"로 분류했습니다. LSTM은 시계열 데이터를 효과적으로 처리하여 짧은 시간 동안의 동작 변화를 학습하였다. 정확도는 훈련 데이터에서 100% 정확도를 달성했으며, 테스트 데이터에서도 높은 성능을 보였다.

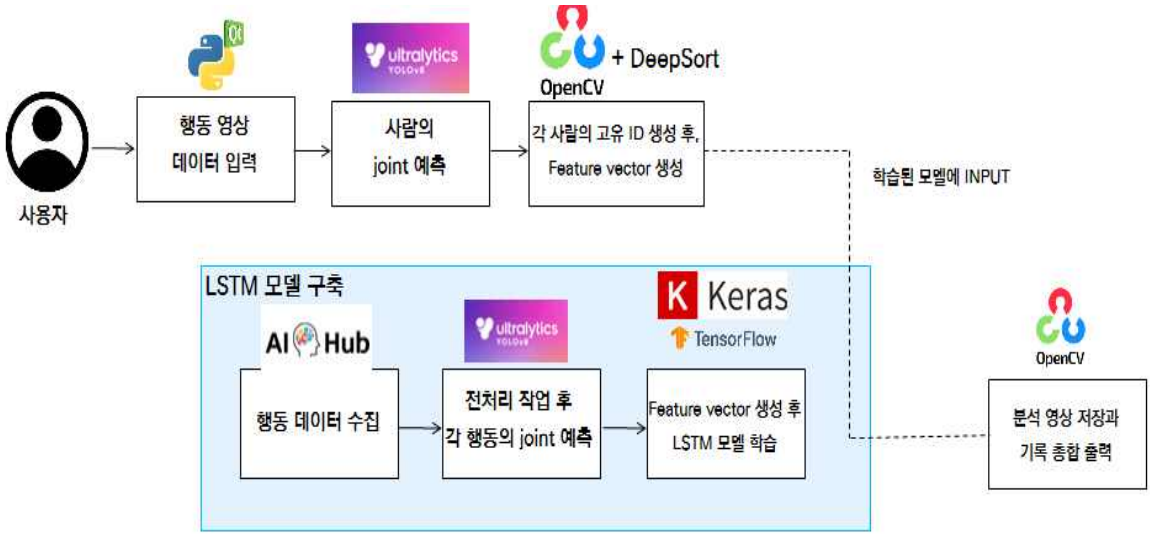
이 논문뿐만 아니라 여러 프로젝트와 논문을 참고한 결과 행동 분석을 위해 사용한 딥러닝 모델을 주로 LSTM을 사용하고 있다는 것을 알았다. 그렇기에 이 내용을 토대로 LSTM과 다른 모델을 사용해서 어느 모델을 사용하는 것이 좋은 지도 연구를 진행하였다.

Ⅱ. 소요기술

(1). 개발환경 및 시스템 구성도

개발 환경은 NVIDIA GeForce GTX 1660 SUPER GPU와 AMD Ryzen 5 3600 CPU로 높은 스펙에서 모델 훈련과 평가를 진행하였다. 운영 체제는 Window 환경이고, 프로그래밍 언어는 Python 3.9, 프레임 워크는 Pytorch, keras를 사용하여 모델 구축을 완료하였다. 프로그래밍 프로그램은 Pycharm, colab을 사용하여 전체적인 프로그램을 제작하였다.

시스템 구성도의 전체적인 구조는 다음과 같다.



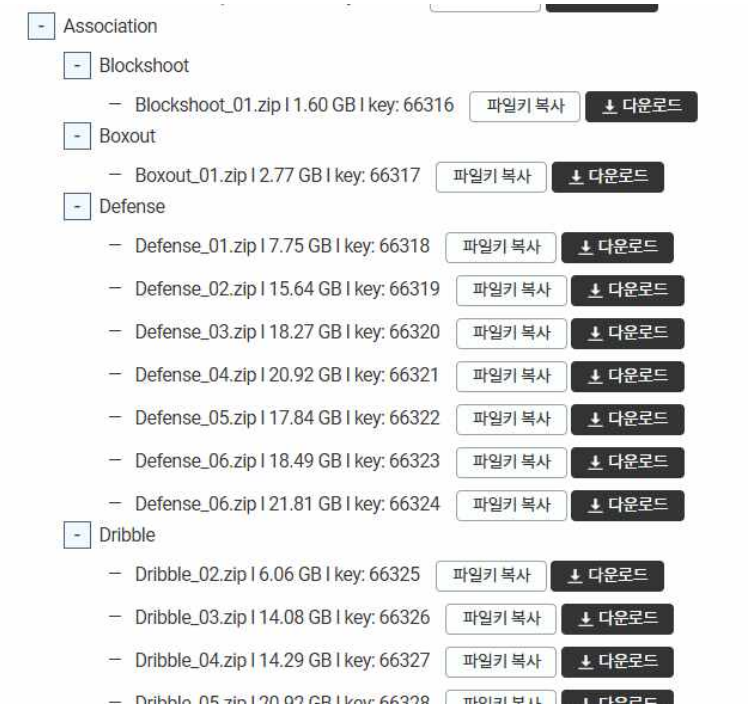
(사진 3) 시스템 구성도

사진 3을 보면서 시스템의 흐름을 설명하자면 먼저 LSTM을 통해 모델을 구축하게 된다. AIHUB에 있는 스포츠 사람 동작 (농구) 데이터의 여러 농구 행동들의 데이터를 사용한다. 이렇게 받은 행동의 관절을 YOLOv8을 통해 관절을 예측하여 관절 데이터를 추출하여 Feature vector를 생성한 뒤에 Keras, TensorFlow를 사용하여 LSTM 모델을 학습하여 분류 모델을 만들게 된다.

이렇게 만든 모델을 사용하는 과정은 사용자가 농구 경기 영상 데이터를 프로그램에 입력하고, 행동을 예측하고 싶은 사람을 ROI를 통해 정하게 된다. 그렇게 지정한 사람을 매 프레임마다 추적하여 YOLOv8을 통해 예측 관절 데이터를 추출하게 되고 Feature vector를 만들게 되면 그 데이터를 만들어 놓은 LSTM 모델에 input하게 된다. 매 프레임 마다 예측한 행동을 출력하게 되면서 경기 영상을 분석하게 된다. 이러한 방법을 통해 농구 경기에서 특정 사람의 기록지를 만들게 되어 원하는 목표를 이룰 수 있게 된다.

(2). 데이터 처리

전체적인 학습과 테스트 데이터는 AIHUB에 스포츠 사람 동작 (농구) 데이터를 사용하려 기획하였다. AIHUB는 AI 학습데이터를 쉽게 구하지 못하는 데이터를 올려주는 데이터 사이트이고, 여기에 있는 데이터 중에 농구 동작 데이터를 사용하여 진행하려 하였다. 농구 행동 동작은 크게 3가지로 구분하려 하기에 슈팅, 패스, 드리블의 데이터를 활용하였다.



(사진 4) AIHUB 스포츠 사람 동작(농구) 데이터 중 일부

하지만 실제로 사용하려 보니 데이터의 질이 아쉬웠다. 여기 데이터는 사람이 직접 한 프레임씩 슈팅하는 사람의 바운딩 박스를 체크하여 annotation 파일이 따로 있었기에 그 정보를 이용하여 행동하는 사람의 데이터만을 학습하려 했다. 실제로 사용해보니 바운딩 박스 크기가 일정하지 않고, 한 프레임씩 행동하는 사람이 아닌 바운딩 박스가 있거나 행동하는 사람이 다른 사람에 가려져있거나 행동하는 모든 모습이 안나와서 관절을 추출하기 어렵거나 등등 문제가 많았다. 그래서 이 데이터만으로는 모델의 정확도가 많이 안좋았기에 본인이 각 행동을 직접 촬영하여 정확도를 보완하였다.

총 사용한 데이터의 수는 한 동작의 영상 데이터 30FPS, 3가지 동작 분류, 각 행동의 영상 300개, 30x3x300 = 27000장의 프레임을 사용하였다.

전처리 과정은 먼저 학습 영상 데이터의 길이가 일정하지 않았기 때문에 이것을 먼저 통일해주었다. 각 행동의 영상을 하나씩 30프레임으로 학습을 진행하기 위해 모든 영상을 30프레임으로 리샘플링을 진행하였다. 예를 들어 90프레임을 중간중간 프레임을 건너뛰어 30프레임으로 요약을 하는 기술이다.

```
cap = cv2.VideoCapture(video_path)
total_frames = int(cap.get(cv2.CAP_PROP_FRAME_COUNT))
frame_indices = np.linspace(0, total_frames - 1, sequence_length).astype(int)
    # 리샘플링 인덱스
```

(코드 1) 리샘플링 코드

sequence_length에 30을 넣어서 0부터 total_frames 사이의 숫자를 30개로 압축시켜서 프레임 숫자를 추출하고 추출된 프레임만을 선택하여 영상을 압축시켰다.

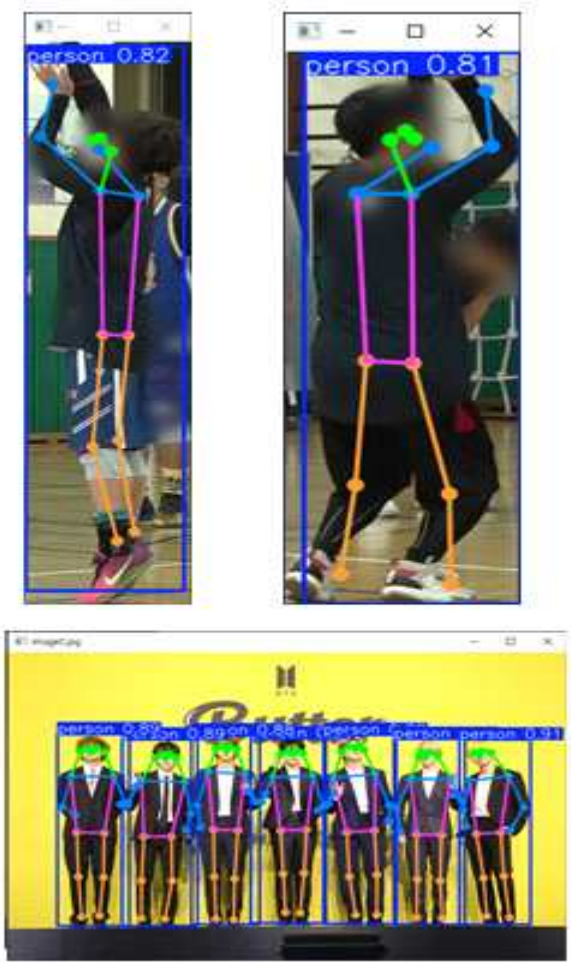
(3). Pose Estimation

sequence 데이터를 만들기 전에 관절을 추정하는 모델을 선택하려 한다. 보통 논문이나 프로젝트에서는 OpenPose, Mediapipe 같은 모델 라이브러리를 자주 사용한다. 굉장히 옛날부터 나온 모델이지만 성능도 준수하고 사용도 용이하기 때문이다. 하지만 본 프로젝트에서는 이러한 모델보다 최근에 나온 YOLOv8 pose estimation을 사용하려 한다. 처음에는 단순히 다른 프로젝트와 차별점을 두고 싶어서 YOLOv8 pose를 사용하려 하였지만 실제로 찾아보니 다른 포즈 모델과 다르게 장점이 여럿 있었다.

이런 관절 추출을 하기 위한 과정은 크게 Top-down 방식과 Bottom-up 방식이 있다. Top-down 방식은 사진에 보이는 사람을 heavy person detector를 수행하여 검출하고, 각각의 사람의 pose estimation을 진행한다. 이러한 방식의 복잡도는 사람의 수에 비례하여 증가하게 되고 이는 실시간 검출에 적절하지 않은 high complextity와 variable runtime을 야기한다. 실시간으로 행동을 분석해야 하는 이번 프로젝트에는 적합하지 않은 방법이라 생각하였다. Bottom-up 방식은 위 방식과는 다르게 일관적인 run-time을 가진다. 한 번의 연산으로 이미지의 heatmap을 계산하고 예상되는 모든 keypoint를 검출해내기 때문에 일정한 시간이 나오게 된다. 하지만 인접한 keypoint간의 heatmap을 분별할 수 없기 때문에 부정확한 결과를 도출할 수 있다. 또한 이런 과정을 가속화하는 방법도 굉장히 힘들다고 한다. 이러한 과정을 토대로 YOLO-Pose 에서는 두 방식의 장점만을 가져와 Pose estimation을 수행한다.

YOLO-Pose는 사람을 잘 검출할 수 있는 YOLOv5 framework를 base로 삼아서 짧은 시간안에 이미지에 나온 사람을 검출할 수 있다. 또한 공간적으로 가까운 위치의 사람이더라도 다른 anchor box를 지니고 있기 때문에 분별이 쉽고, 이는 다른 사람과의 keypoints와 분별을 가능하게 한다는 것이다. 이는 bottom-up 방식의 단점을 해결해준다. 정리하자면 간단하게 이미지에 보이는 사람을 검출한 다음에 이미지에 보이는 모든 keypoint를 검출한 다음에 사람 anchor box를 통해 keypoint들의 각각 연관을 시켜준다. 이러한 방식이면 사람이 많아져도 비교적 Top-down 방식보단 빠르며 bottom-up 방식보단 정확도가 높게 나온다는 것을 말한다. 해당 YOLO-Pose 논문에서도 이를 입증하여 좋은 정확도를 나타내었다.

이러한 이유로 인해 YOLO-Pose를 사용하여 본 프로젝트에 사용하려 한다. 이를 사용할 때 주의점은 다른 관절 추출 모델과는 다른 Keypoint의 변수를 가지게 된다. 이를 유의하여 양팔, 양다리, 몸의 관절을 찾아서 진행하면 된다.



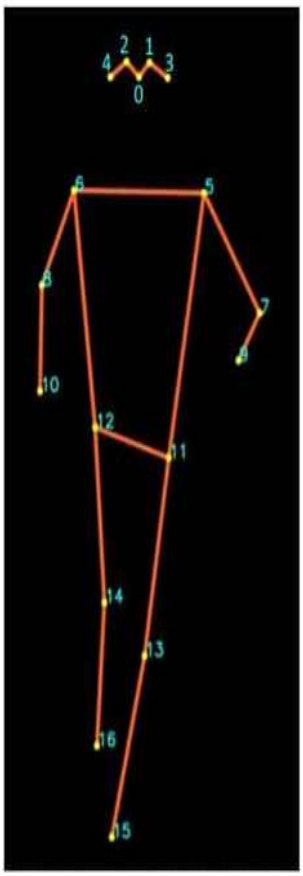
(사진 5) YOLO-pose 예시 사용

(4). Action Classifier

YOLO-Pose를 이용하여 올바른 joint를 추정하고서 이를 이용해 행동 분류 모델을 개발한다. 초기 단계에서는 프레임별 joints 위치를 단순히 학습시켜 행동을 예측하려 했으나, 선행 연구 및 관련 프로젝트 분석을 통해 몇 가지 중요하게 고려해야 할 점을 도출하였고, 이를 토대로 모델 input 설계를 변경하였다.

첫 번째로 고려할 특징은 행동 예측에 필요 없는 joint이다. 행동 예측에 사용되는 joints 데이터는 일반적으로 눈, 귀, 어깨, 팔꿈치, 손목, 힙, 무릎, 발목, 추가적으로 손가락과 발가락 관절까지 구할 수 있다. 하지만 특정 관절 데이터가 특정 행동을 할 때 큰 영향을 미치지 않는다. 예를 들어 앉기, 서기 등을 구분할 때 눈, 귀, 손가락 같은 관절은 필요로 하지 않는다. 이러한 input 관절 데이터를 줄이게 되면 조금 더 정확도 있고 속도도 비교적 빠르게 될 것이다. 따라서 몸통, 팔, 다리를 제외한 관절 데이터는 사용하지 않고 전처리를 진행하였다. (사진 6)을 보면 0~4번을 제외하고 나머지 keypoint로 분석을 진행하였다.

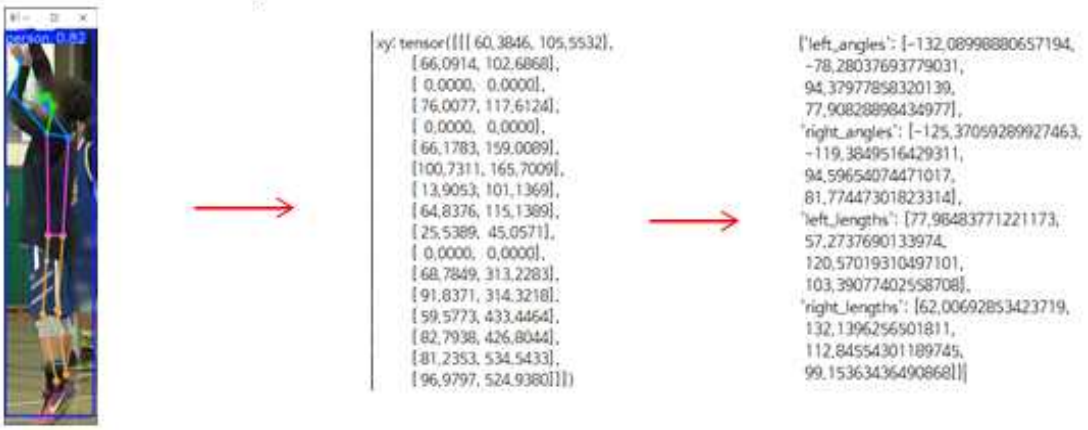
Index	Key point
0	Nose
1	Left-eye
2	Right-eye
3	Left-ear
4	Right-ear
5	Left-shoulder
6	Right-shoulder
7	Left-elbow
8	Right-elbow
9	Left-wrist
10	Right-wrist
11	Left-hip
12	Right-hip
13	Left-knee
14	Right-knee
15	Left-ankle
16	Right-ankle



(사진 6) YOLO-pose keypoint 변수

두 번째는 위치 데이터만을 학습했을 때 발생하는 문제점이다. 입력하는 영상의 크기, 사람들의 bounding box 크기, 관절 현재 위치 등 이러한 요소는 항상 바뀔 수 있고 특정 상태만 학습하면 여러 상황의 정확도가 많이 안 나올 것으로 예상이 되었다. 예를 들어 특정 프레임만 보면 정면에서 걷는 모습과 서있는 모습이 비슷하게 보일 수 있고, 사람의 bounding box좌표를 임의로 200x400으로 고정시킨다면 작게 보이는 사람의 좌표를 억지로 늘리게 되면 관절을 예측하기에도, 불완전한 데이터를 학습하게 된다. 또한 카메라와 사람 각도에 따라 학습하는 행동 데이터가 혼동이 올 수 있다고 판단하였다. 다른 논문과 프로젝트에서도 해당 부분을 고려하여 좌표 자체만을 input으로 하지 않고 다른 식으로 변환하여 input을 진행하였다.

좌표값 만을 의존하지 않고 데이터를 활용하기 위해서 각도와 거리를 활용했다. 이렇게 입력 벡터를 변경한다면 이미지의 크기, 카메라의 각도에 크게 영향을 받지 않고 일정한 데이터를 얻을 수 있을 것이다. 따라서 (사진 7)처럼 입력 벡터 형태는 각도와 joint간의 길이를 넣어준다. 추가로 본 프로젝트는 단순히 이미지로만 판단하는 것이 아닌 영상 데이터를 입력하고서 전체적인 행동을 분류하는 모델을 만드는 것이 목표기 때문에 시간적 변화 정보 또한 입력 벡터에 추가하였다. 전 프레임과 비교해서 각도가 어느정도 변화하였고, 관절과 관절 사이의 거리가 어느정도 변화하였는 지에 대해 입력 벡터를 설정하였다.



(사진 7) 관절 좌표 데이터를 통해 각도 변환 과정

이러한 내용을 토대로 입력 벡터를 수정하여 모델 정확도와 분류 속도를 높이려는 과정을 보였다. 각 입력 벡터와 계산 방법의 실제 코드는 다음과 같다.

```
joint_pairs = [
    (5, 7),    # 왼쪽: 어깨 - 팔꿈치        (7, 9),    # 왼쪽: 팔꿈치 - 손목
    (6, 8),    # 오른쪽: 어깨 - 팔꿈치      (8, 10),   # 오른쪽: 팔꿈치 - 손목
    (11, 13), # 왼쪽: 힙 - 무릎             (13, 15), # 왼쪽: 무릎 - 발목
    (12, 14), # 오른쪽: 힙 - 무릎           (14, 16), # 오른쪽: 무릎 - 발목
    (5, 11),   # 상체 기울기: 왼쪽 어깨 - 왼쪽 힙
    (6, 12)    # 상체 기울기: 오른쪽 어깨 - 오른쪽 힙
]

... 중략 ...

# 각도 계산
angles = []
            for (j1, j2) in joint_pairs:
                if (
                    j1 < keypoints.shape[0] and
                    j2 < keypoints.shape[0]
                ):
                    v1 = keypoints[j1]
                    v2 = keypoints[j2]
                    angle = np.arctan2(v2[1] - v1[1], v2[0] - v1[0])
                    angles.append(np.degrees(angle))

# 기준 길이 계산
            if (
                keypoints.shape[0] > 11 and
                not np.any(np.isnan(keypoints[5])) and
                not np.any(np.isnan(keypoints[11]))
            ):
                reference_length = np.linalg.norm(keypoints[5] - keypoints[11])
            else:
                reference_length = 1
```

(코드 2) 관절 데이터와 각도, 길이 계산

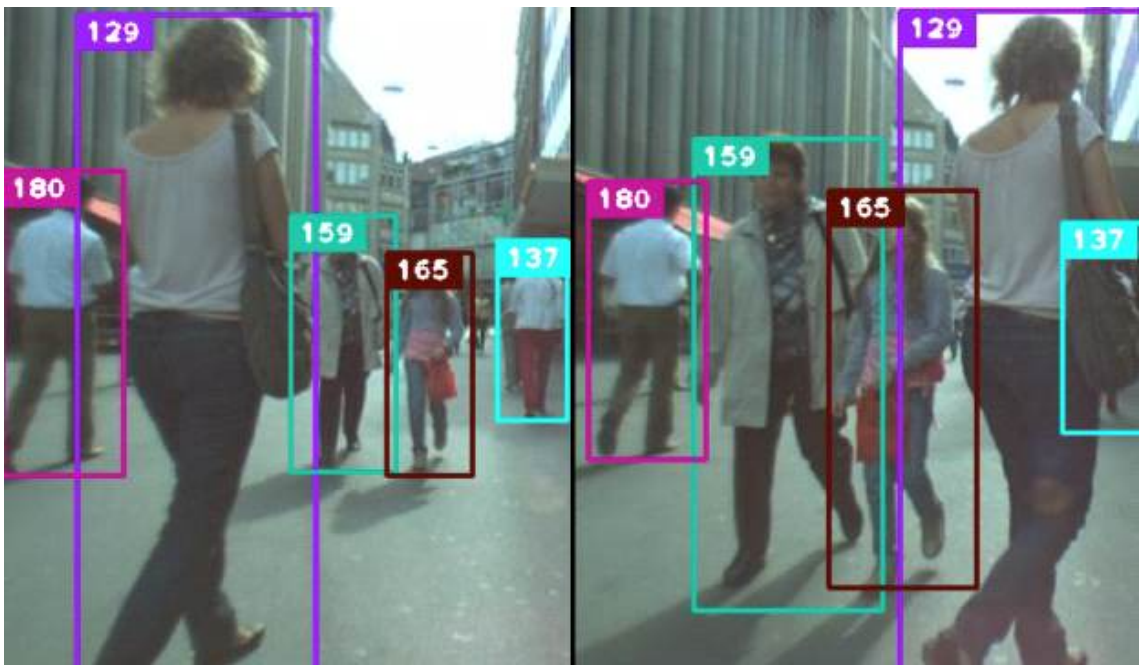
YOLO-Pose에서 해당하는 관절끼리 이어서 선을 만들고 그 선에 대해 기울기를 각도로 계산하였고, 거리를 구한다. 다음 프레임에서 이 데이터를 사용하여 각도와 거리의 변화량을 구한 다음에 모델 입력 데이터에 넣어 주었다. 총 입력 벡터는 관절 10개 x (각 관절에 따른 현재 각도, 각도 변화량, 거리 변화량) 3가지 = 30가지 이고 추가로 30 시퀀스 까지 하여 900개가 입력 벡터가 된다.

(5). 그 외에 사용한 기술

기본적으로 이미지, 동영상을 처리하는 기술이 필요하기 때문에 영상 처리를 위해 Python OpenCV를 통해 영상의 크기 조정, 노이즈 제거, ROI 설정 등 전처리 과정을 진행하여 모델의 정확도를 높일 수 있다.

딥러닝 모델을 선정할 때 여러 논문과 프로젝트를 분석한 결과 LSTM 모델을 사용하는 것을 깨달았다. 처음에는 RNN을 사용하여 진행을 하려 했지만 동영상의 데이터가 계속 한 프레임씩 모델을 적용하려면 많은 데이터의 양을 축적시켜야 하고, 이렇게 되면 장기 기억 보존 문제가 생길 수 있다. 이를 해결하기 위해서 앞서 알아본 논문과 프로젝트가 LSTM을 사용하고 있었다. 이 점 때문에 이 프로젝트에서도 LSTM을 채택하여 사용하고자 한다.

사람을 ROI로 선택하여 그 사람만을 따라가 관절을 추출하는 과정을 진행해야 하는데 이 추적 방법에 대해 사용한 기술이 DeepSort이다. 우선 YOLO를 통해 이미지 안에 사람을 인식하게 되면 사람마다 ID가 부여된다. 하지만 이 ID는 다음 프레임으로 넘어가서 다시 사람을 Detecting하면 ID가 똑같이 부여되는 것이 아닌 랜덤으로 바뀌게 된다. 그렇기 때문에 이것을 방지하고자 DeepSort를 사용하게 되면 전 프레임과 비교해서 IOU 값을 통해 오브젝트의 ID를 이어서 설정할 수 있다. 이렇게 추적 형태를 만들고서 학습과 프로그램 제작에 DeepSort를 사용하게 되었다.



(사진 8) DeepSort를 이용한 ID 고정

Ⅲ. 개발 내용

(1). LSTM 모델

LSTM(Long Short-Term Memory)은 Recurrent Neural Network(RNN)의 한 종류로, 시계열 데이터나 순차적인 데이터 처리에 적합한 딥러닝 기술이다. RNN의 주요 단점인 기울기 소실(Gradient Vanishing) 문제를 해결하기 위해 설계된 LSTM은 데이터를 기억하고 잊는 과정을 조절하는 게이트 구조(Forget Gate, Input Gate, Output Gate)를 포함하고 있다. 이를 통해 LSTM은 긴 시간 간격에서도 데이터 간의 상관관계를 학습할 수 있기에 이번 프로젝트에서 사용하는 것이 맞다고 판단하였다.

모델을 구성하는 층은 기존에 행동 분류에서 사용한 모델 구조를 참고하여 구성하였다. 3개의 LSTM 레이어를 사용해서 계층적인 시간적 특징을 학습하고자 한다. 첫 번째 LSTM 레이어는 128개의 유닛을 가지며, return_sequences=True로 설정되어 다음 레이어에도 전체 시퀀스 정보를 전달한다. 이 레이어는 과적합을 방지하기 위해 L2 정규화와 드롭아웃(30%)을 적용하였으며, Batch Normalization을 통해 학습 안정성을 높였다. 두 번째 LSTM 레이어는 64개의 유닛으로 구성되며, 첫 번째 레이어의 출력 시퀀스를 처리한다. 동일하게 L2 정규화, 드롭아웃(30%), Batch Normalization이 적용되어 모델의 일반화 성능을 향상시켰다. 세 번째 LSTM 레이어 역시 64개의 유닛을 가지지만, return_sequences=False로 설정하여 마지막 타임스텝의 출력만 반환한다. 이는 시퀀스 데이터를 최종적으로 고정된 크기의 벡터 표현으로 변환하는 역할을 한다. LSTM 레이어 이후에는 64개의 유닛을 가진 Dense 레이어가 추가되어 비선형 활성화를 적용하며, 과적합 방지를 위해 L2 정규화와 Batch Normalization을 사용한다. 마지막으로, 출력층은 모델의 클래스 수에 해당하는 유닛을 가지며, 소프트맥스 활성화 함수를 통해 각 클래스에 대한 확률 분포를 제공한다.

이 모델은 Adam 옵티마이저를 사용해 학습되며, 손실 함수로 다중 클래스 분류 문제에 적합한 categorical_crossentropy를 사용한다. 학습 중 성능 향상을 위해 검증 정확도를 모니터링하고, 가장 우수한 성능을 보인 모델만 저장하는 체크포인트를 설정하여 최적의 가중치를 확보하도록 설계하였다.

```
# LSTM 모델 정의
model = Sequential([
    # 첫 번째 LSTM 레이어
    LSTM(128,input_shape=(sequence_length,num_features), return_sequences=True,
kernel_regularizer=l2(0.001)),
    BatchNormalization(),
    Dropout(0.3),

    # 두 번째 LSTM 레이어
    LSTM(64, return_sequences=True, kernel_regularizer=l2(0.001)),
    BatchNormalization(),
    Dropout(0.3),

    # 세 번째 LSTM 레이어
    LSTM(64, return_sequences=False, kernel_regularizer=l2(0.001)),
    BatchNormalization(),
    Dropout(0.3),

    # 출력 레이어
    Dense(64, activation='relu', kernel_regularizer=l2(0.001)),
    BatchNormalization(),
    Dense(num_classes, activation='softmax')
])

# 모델 컴파일
model.compile(optimizer='adam',loss='categorical_crossentropy', metrics=['accuracy'])

# 모델 저장 설정 (최고 성능 모델만 저장)
checkpoint = ModelCheckpoint('best_lstm_model.keras', monitor='val_accuracy',
save_best_only=True, mode='max', verbose=1)
```

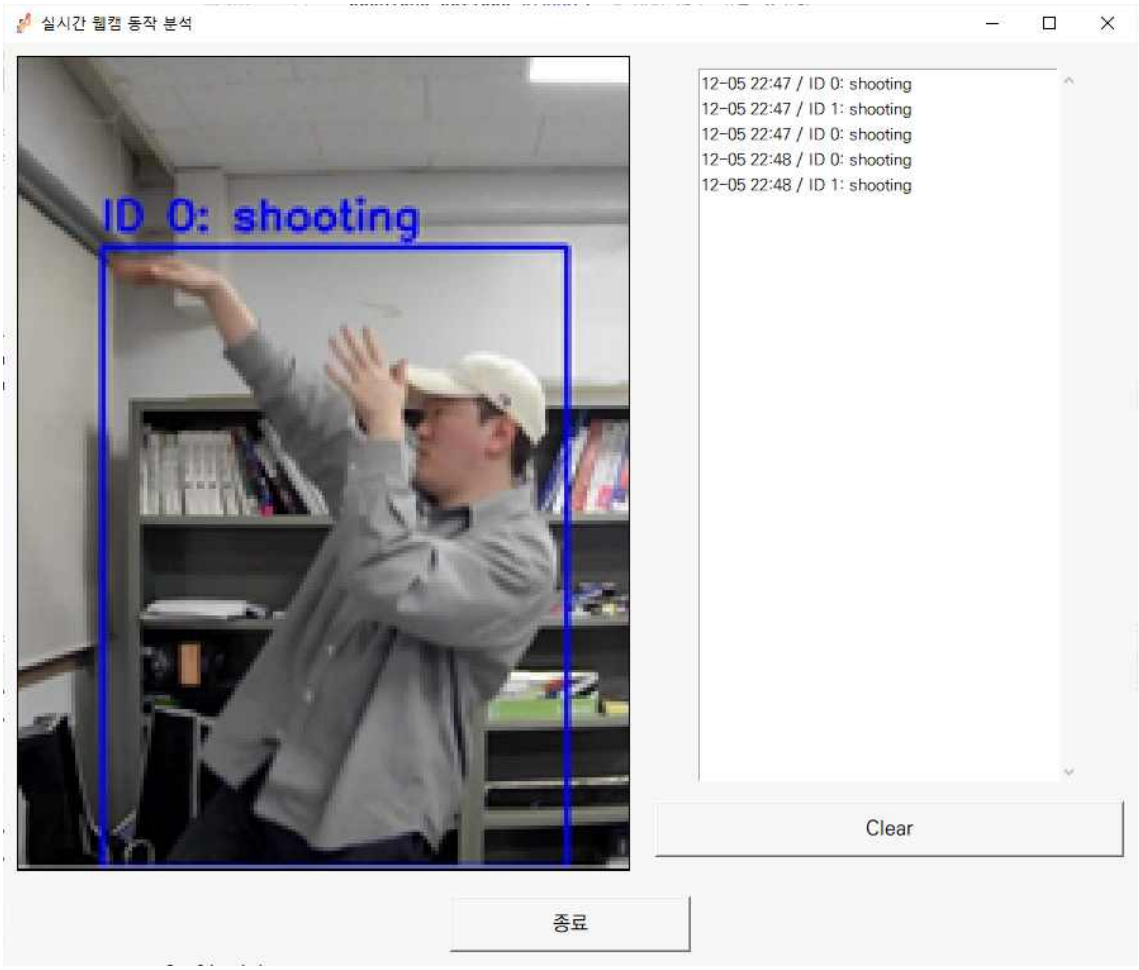
(코드 3) LSTM 모델 구축 코드

(코드 3)을 보게 되면 앞서 설명한 내용으로 코딩을 진행했다. train accuracy는 0.98, train loss 0.16까지 내려갔다. test 정확도는 0.756 정도로 꽤나 준수한 정확도를 보였다.

(2). 실시간 웹캠 분석 프로그램

이렇게 만든 모델을 테스트하면서 어떤 방향성을 가지고 응용 프로그램을 만들 수 있는지 간단하게 만들었다. 프로그램 첫번째는 실시간으로 웹캠 영상을 불러와서 분석하는 프로그램이다. 단순하게 화면에 보이는 사람을 추적하면서 시간, ID, 어떤 행동을 했는지 옆에 로그 창에 출력하게 된다.

GUI는 PyQT를 사용해서 웹캠 영상, 로그창, 로그 초기화 버튼을 배치하였다.



(사진 9) 실시간 웹캠 동작 분석 UI

왼쪽 화면이 현재 캠 화면이다. 사람이 인식되면 사람의 바운딩 박스가 파란색으로 나오게 되고, ID 번호를 출력한다. 해당 사람이 하는 행동을 예측하여 박스 위에 나타내고 로그를 출력한다. 3가지 행동 분류를 진행하는데 슈팅, 패스, 드리블을 분류하고 3가지의 행동이 임계값 0.7을 넘기지 못하면 none으로 출력하게 되고 none이 아니면 로그에 출력하게 된다. 또한 매 프레임마다 슈팅, 패스를 로그에 출력하는 것이 아닌, 전 프레임에서 슈팅을 분류했는데 슈팅이 또 나오게 되면 출력을 안하도록 중복출력을 방지하였다.

```
            confidence_threshold = 0.6
            max_prob = max(action_probs[:3])
            action_label = action_labels[np.argmax(action_probs)] if max_prob >=
                        confidence_threshold else 'None'

...중략...

            if track_id not in previous_actions:
                previous_actions[track_id] = None

            if action_label in ['shooting', 'pass']:
                if previous_actions[track_id] != action_label:  # 새로운 행동만 기록
                    current_time = datetime.now().strftime("%m-%d %H:%M")
                    log_text.insert(tk.END, f"{current_time} / ID {track_id}:
                                                                {action_label}\n")
                    log_text.see(tk.END)
                    previous_actions[track_id] = action_label
            else:
                previous_actions[track_id] = None  # 행동이 None이면 초기화
```

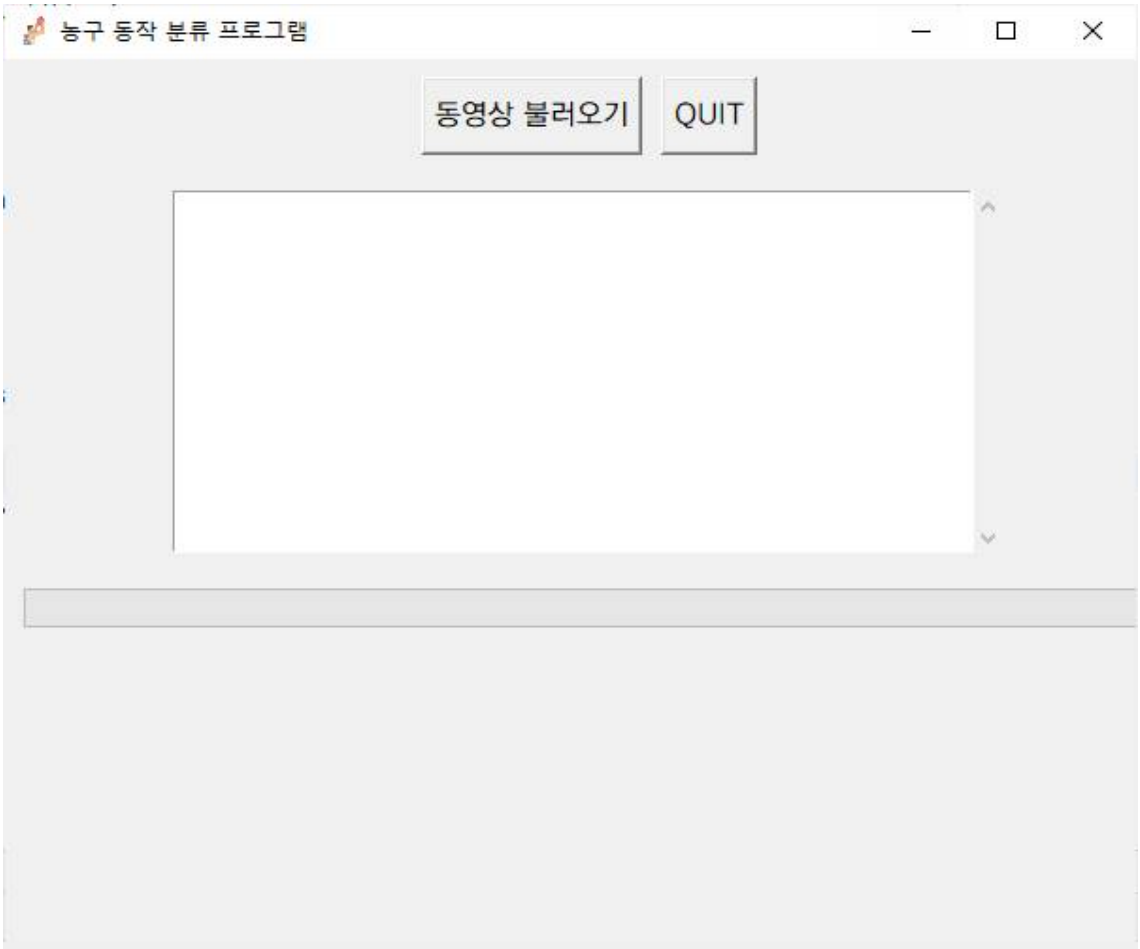
(코드 4) 임계값 설정과 중복 출력 방지

이 프로그램은 단순히 모델이 잘 작동하는지, 정확도는 어떻게 나오는 지 알아보기 위해 만든 것이지만 발전 가능성은 충분하다고 생각한다. 이번 프로젝트는 스포츠중 하나인 농구를 선택해서 모델을 구성하고 했지만 더 나아가 지능형 CCTV로 이상 행동이 생겼을 때 인지할 수 있을 것으로 보인다. 실제로 졸업작품 전시회에서 직접 행동을 체험해보았을 때 슈팅과 드리블은 행동이 명확해서 행동할 때마다 결과가 좋았지만, 패스는 정석 포즈로 한 데이터가 많이 없었기 때문에 행동을 예측하기 힘들고 정확도도 낮았다.

(3). 동영상 데이터 분석 프로그램

두 번째 프로그램은 동영상 데이터 분석 프로그램이다. 앞서 보여준 시스템 구성도의 내용이 본 프로그램이다. 개요에서 말한 주제이기 때문에 이 프로그램 제작에 집중하였다. 이 또한 GUI를 PyQT를 이용하여 프로그램을 제작했다.

프로그램의 흐름은 준비한 농구 동영상을 불러오게 되면 첫 프레임을 보여주고 사진에 나와있는 사람 중에 동작 분류 대상을 ROI를 통해 고르게 되면 그 사람의 ID를 저장하게 된다. 여러 사람이 있어도 그 ID만을 분석하여 한 프레임씩 관절 데이터를 추출하고 시퀀스 데이터를 만들게 된다. 또 하나의 창이 나오게 되면서 실시간으로 분석하고 있는 모습을 보여준다. 현재 행동이 어떠한지를 나타내고 메인 화면 막대 바는 분석 진행 상황을 보여준다. 전체 프레임을 100으로 놓고 프레임을 처리 할때마다 갱신하는 방식이다. 모든 분석이 완료되면 어떠한 행동을 몇번 했는지 간이 기록지를 출력하게 된다.

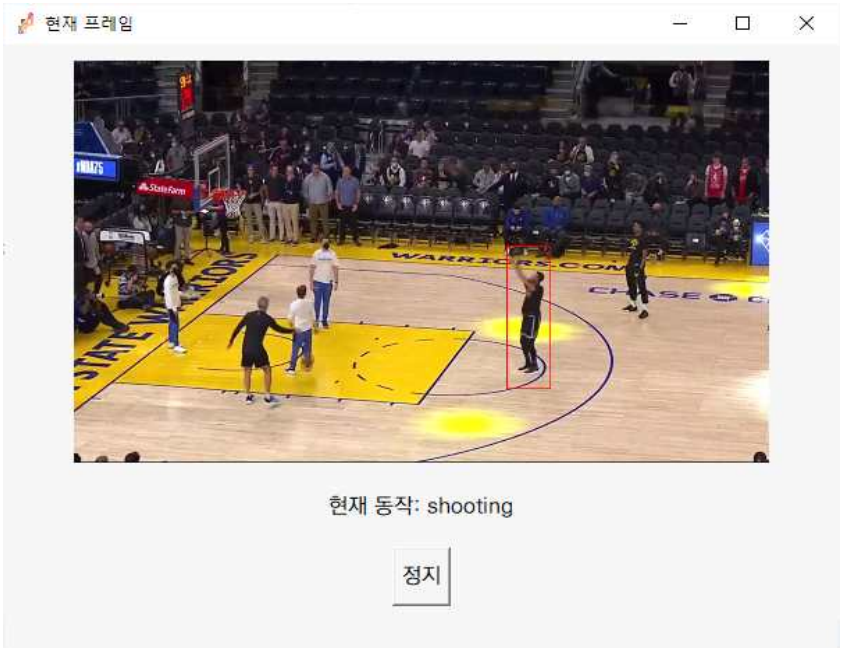


(사진 10) 동영상 데이터 분석 프로그램 UI

(사진 10)에서 동영상 불러오기를 누르게 되면 컴퓨터에 저장되어 있는 동영상 데이터를 불러올 수 있다. 불러오게 되면 선수를 ROI로 선택하는 창이 나오게 된다.

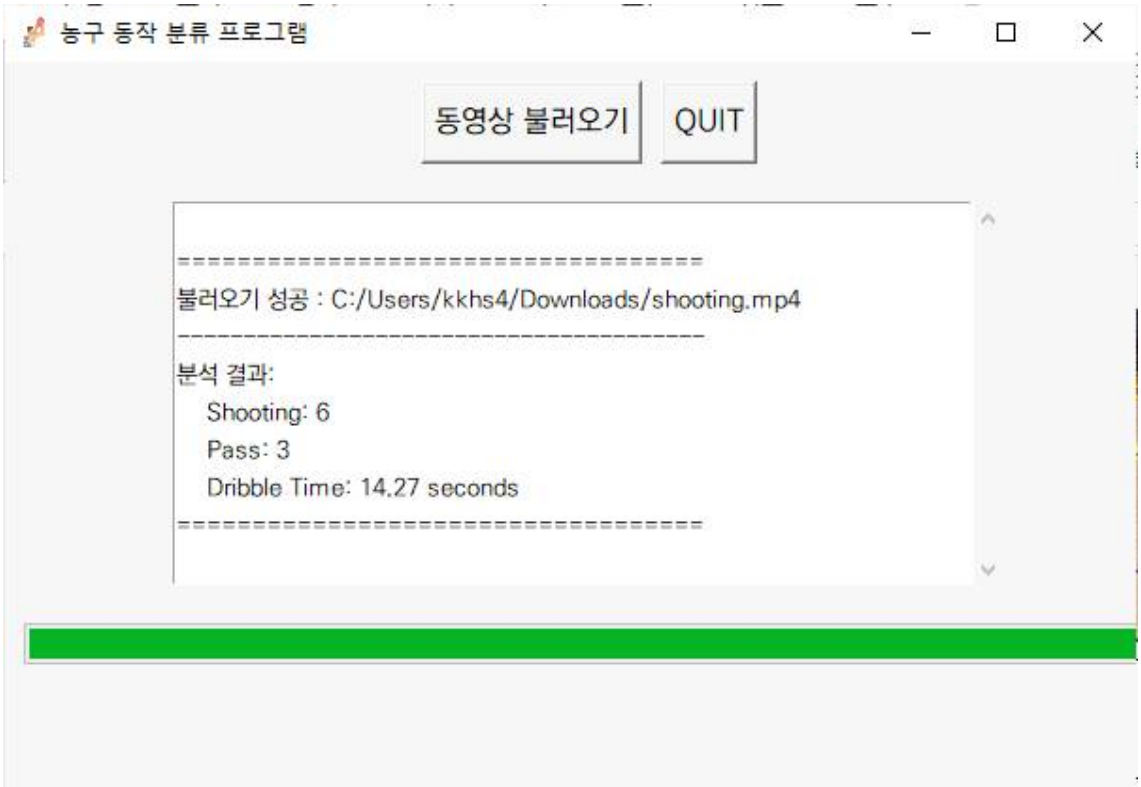
(사진 11) ROI 창에서 선수 한명을 지정한 상태

(사진 11)처럼 한 사람을 지정하고 엔터를 누르면 해당 선수를 계속해서 추적하며 매 프레임마다 현재 동작이 어떤 동작인지 출력하게 된다. 정지 버튼으로 중간에 정지도 가능하게 하였다.



(사진 12) 현재 분석 상황을 보여주는 창

현재 추적하는 사람은 빨간색 bounding box로 보여주고 분석 결과를 매 프레임마다 보여주게 된다.



(사진 13) 분석 결과 출력 화면

분석하는 동작을 카운팅하는 방법은 앞서 보여준 실시간 웹캠 프로그램에서 사용한 카운팅 방식을 사용했다. 전 프레임과 같은 동작을 하면 카운팅을 안하는 방식으로 진행하였고, Dribble Time은 매 프레임 Dribble로 분류되면 카운팅을 하여 그 값을 30FPS 영상이면 30으로 나누고, 60FPS 영상이면 60으로 나누어서 출력하였다. 분석 속도는 그렇게 높지는 않지만 실제로 경기중인 영상을 넣어보았을 때 정확도는 좋았다.

Ⅳ. 결론

이번에 진행한 프로젝트는 기존 행동 분석 프로그램에 사용하지 않았던 YOLOv8 Pose를 사용하여 관절 정보 sequence를 입력 벡터로 사용하고 이를 LSTM 모델을 통해 학습시키고 구축하였다. 이 모델을 통해 농구 행동 분석 프로그램을 간단하게 제작하였고 나아가고자 하는 방향성에 대해 제시한다. 행동 분석 정확도를 높이기 위해 여러가지 입력 벡터를 넣어서 연구를 한 결과 train accuracy는 0.98, train loss 0.16 정도로 좋은 지표를 만들게 되었다. 더욱 정확도를 높이려면 완전한 행동의 많은 데이터와 조금 더 체계적인 입력 벡터, 모델 구성에 대한 연구를 통해 발

전 할 가능성이 있다. 또한 실시간으로 사용하기에는 분석 속도가 오래 걸리므로 모델 경량화 및 추론 속도 최적화도 중요한 과제가 될 것이다. 본 프로젝트에서는 농구 스포츠를 한정적으로 보여주었지만 추후 이상 행동 분석 영상 혹은 다양한 스포츠 행동 분석으로 진행되는 것을 기대한다.

참고자료

[1] Debapriya Maji, Soyeb Nagori, Manu Mathew, Deepak Poddar, "YOLO-Pose: Enhancing YOLO for Multi Person Pose Estimation Using Object Keypoint Similarity Loss"

[2] imsoo, "Real time Fight Detection Based on 2D Pose Estimation and RNN Action Recognition"

[3] Nicolai Wojke, Alex Bewley, Dietrich Paulus, "Simple Online and Realtime Tracking with a Deep Association Metric"

[4] ultralytics Pose Estimation [https://docs.ultralytics.com/ko/tasks/pose/]

[5] Jiho Choi, Gyutae Hwang, Sang Jun Lee, "Behavior Pattern Prediction Algorithm Based on 2D Pose Estimation and LSTM from Videos"

[6] Jo, Joong Yeon Kim, Jeong Hyun Moon, Hyeun Jun, "Occupant Activity Classification Process Using YOLOv5 based Object Detection "