20210280 안현태

Develop the first project.  
프로젝트 개관: 반복 댄스 및 그와 관련된 형태의 댄스의 녹화 및 분석, 그리고 모듈의 섞기

Define your module of dancing

여기에서 모듈의 정의는 같은 위치, 동작에서 시작해서 끝나는 하나의 단위를 의미한다.  
이를 선택한 이유는, 비록 대부분의 ‘춤’은 같은 ‘모션’에서 시작해서 끝나는 것이 적지만, 비디오게임 등에 사용되는 ‘모션’은 특정한 행동을 반복하는 형태로 형성된다. 대부분의 춤에서 이러한 형태의 모션이 드러나는 경우는 거의 없지만, 특정 형태의 반복, 그리고 그 형태의 변주를 엮어놓은 형태의 춤들이 존재하기는 한다. 비록 아이디어 자체는 다양한 형태의 춤에 적용될 수 있지만, 개인의 기술적 한계/능력적 한계로 인해 “모듈”의 시작과 끝이 어느 정도 일정하거나 수수한 형태의 댄스들을 주로 테스트하고자 한다.(에: 라인댄스)

Explain your technology

아마도 중요하게 이용해야 할 기술은 특정 모션 데이터의 시작과 끝을 섞어줄 수 있는, 그러니까 ‘보간’(Interpolation)을 해줄 수 있는 알고리즘/프로그램일 것이다. 또한, 앞에서의 모듈의 정의에 의해 비슷한 시작점과 종착점을 가지고 있는 모션 데이터를 가진다 하더라도, 사람은 완벽하지 않기에 이러한 모션 데이터의 자세가 완전히 같을 것이라 예상할 수 없으며, 이렇게 비슷한 모션 데이터를 혼합, 취사선택 함으로써 춤의 모션을 안정되게 하는 것이 주요 목표이다.

현재 이에 관련된 기술에 대해서는 더 조사가 필요하다. 우선 비디오에서 모션 데이터를 추출할 수 있는 Mediapipe와 같은 Data Generation, 두 모션 데이터의 특정 부분(시작과 끝)을 비교하고 이의 중간점을 찾을 수 있는 알고리즘, 그리고 이러한 새 모션 데이터를 보여줄 수 있게 다시 한번 적용시킬 수 있는 엔진으로 구성될 것이다.

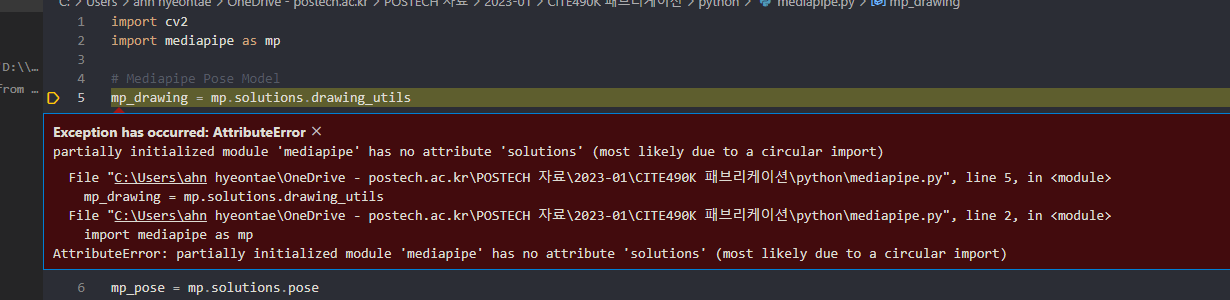
Implementation 순서:

1. Mediapipe을 python을 통해 implement하기

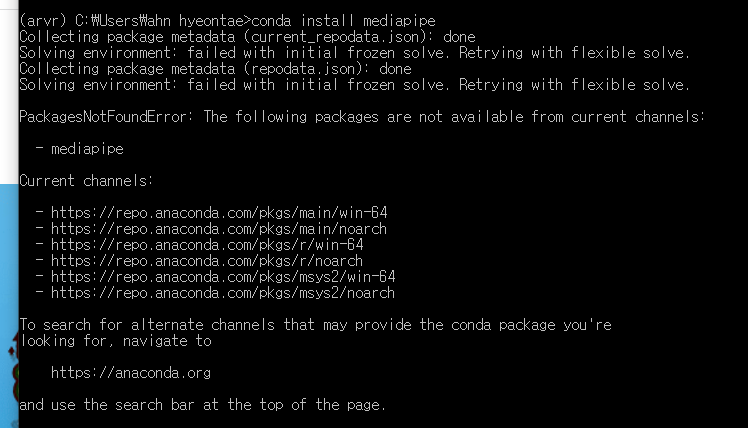
4/27 VSCode를 Python을 사용할 수 있게 수리. 이를 위해 아나콘다 재설치 및 미니콘다 삭제.

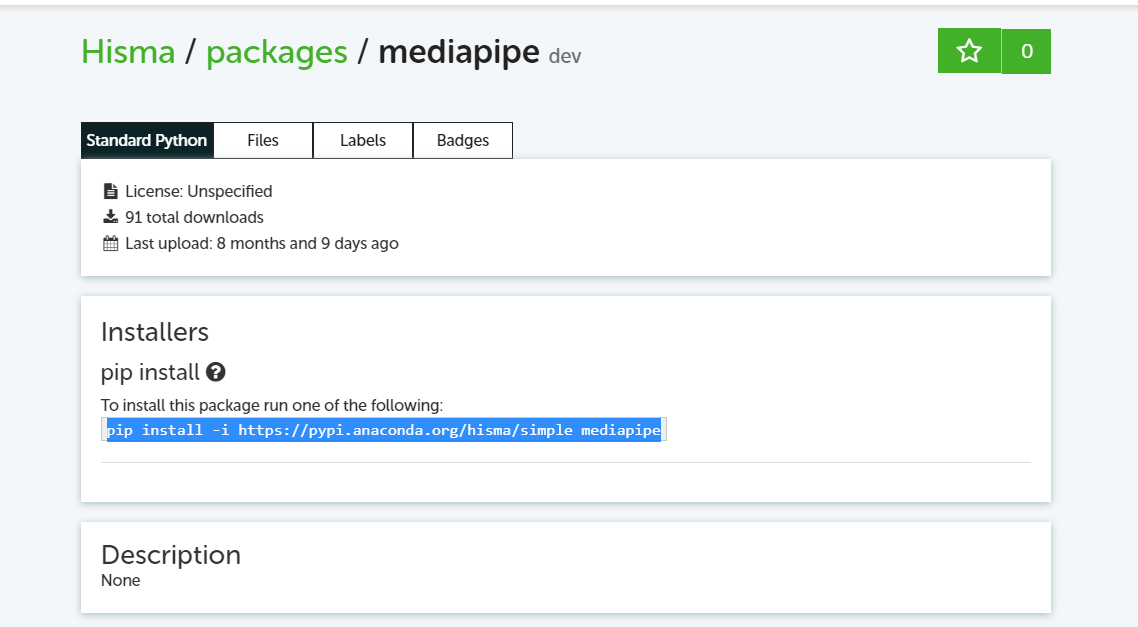
4/29 mediapipe 사용을 위한 기본 과정 재-수강

4/30 현재 mediapipe에 대해서 python 코드 실험중



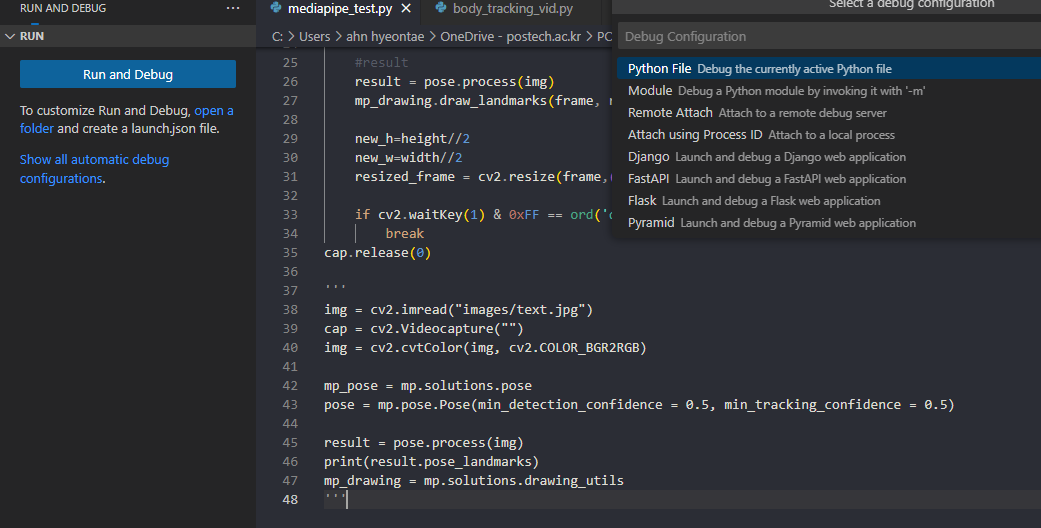
이름을 mediapipe.py로 정했기에 난 오류를 뒤로 하고 새롭게 conda env에 다운로드 하려 하였으나,

와 같이 오류가 발생함으로써 이를 고치고자 함.



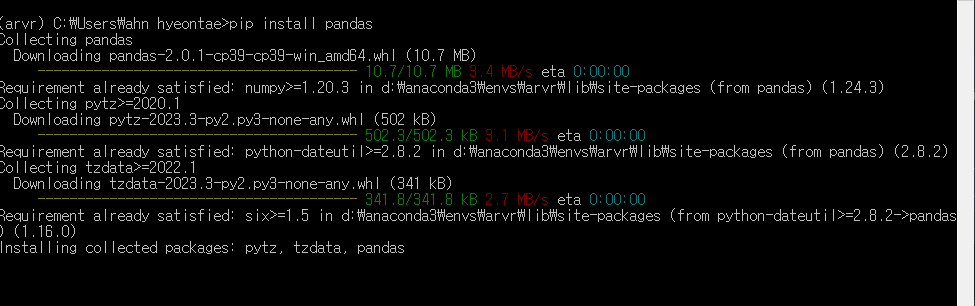
현재 이를 고치고자 하였으나 오류가 계속 나서 Rona 선생님께 질문하고자 월요일로 미룸.

5/1: 근로자의 날로 인해 Rona 선생님 만나지 못함. 또한, 문제를 해결하지 못함. Mediapipe가 아닌, 다른 Motion Detection 모델에 대해서 알아보는 중.

5/2: python -m pip install mediapipe 을 통해 mediapipe 문제 해결. 해당 코드의 동작 확인. 해당 코드의 기록을 위해, 우선적으로 이를 display할 수 있게 수정하는데 몰두. 

이를 통해 작동되는 것을 확인. 하지만, 예시 동영상에 따른 다양한 문제점들을 확인. 이를 해결하기 위해 다양한 방식들을 사용해 볼 예정

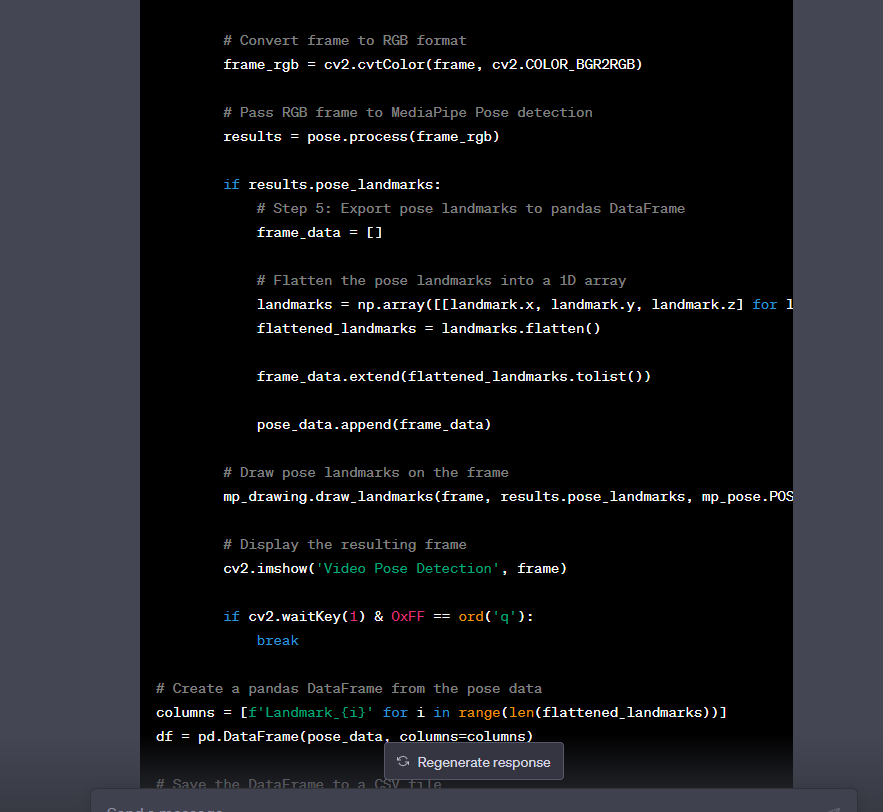
5/3: 우선적으로 관절들의 데이터를 csv 파일로 출력할 수 있도록 해야 한다. 이를 위해 예제로 주어진 ZED Project의 예시를 통해 모델을 csv 형태로 저장할 수 있도록 해본다.  
이를 위해서 pandas libarary를 다운로드한다.

  
5/8: 다운로드된 pandas library를 통해 어떻게 하면 Motion Data를 기록할 수 있는지 알아본다.

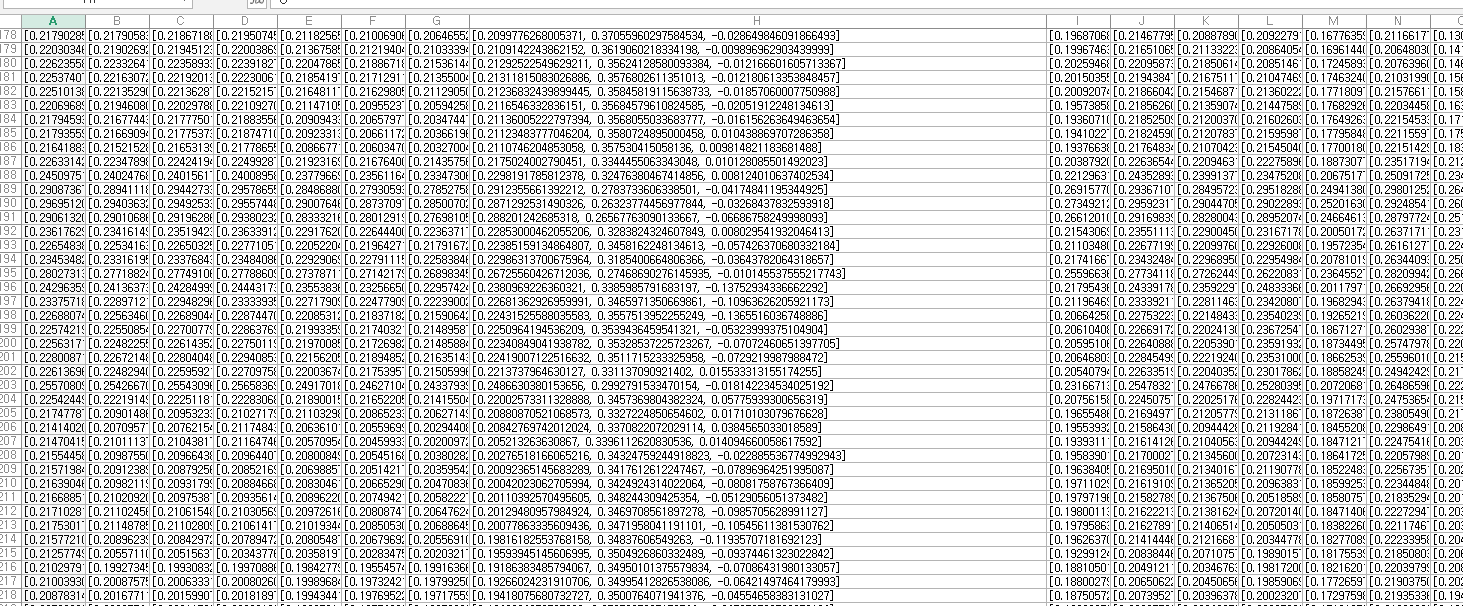
5/9: Mediapipe에서 주어지는 다른 형태의 Data Structure, Holistic에 대해 알 수 있다. 허나, 현재 필요한 것은 pose이기에, 이를 굳이 사용하지는 않는다.

5/13: 다른 작업에 일정이 밀렸지만, Data Structure csv export 구현하기

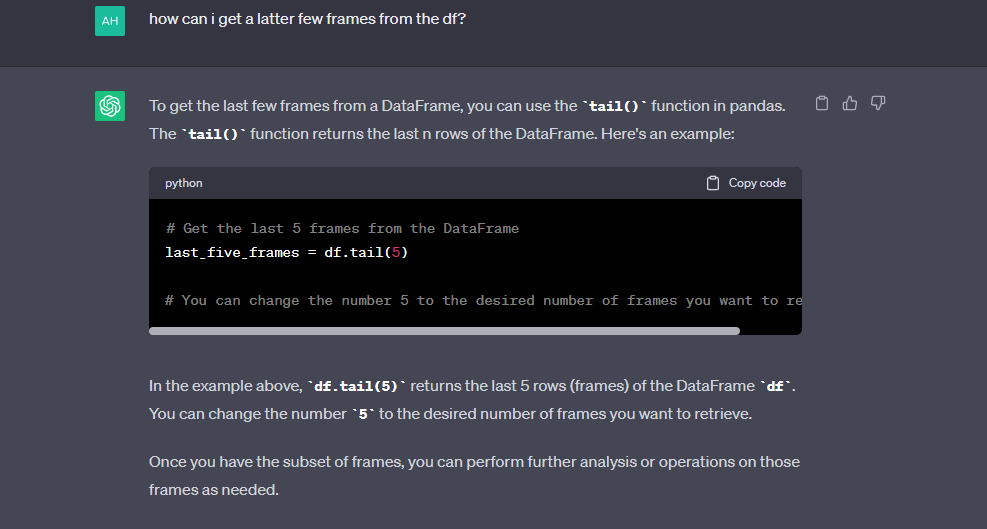
5/14: <https://www.kaggle.com/code/venkatkumar001/pose-prediction-generate-csv-keypoints-mediapipe/notebook> 참조를 통해 csv 구현, <https://github.com/google/mediapipe/issues/1803> <https://pastebin.com/BA4E1fAF> 이 더욱 도움이 됨



ChatGPT를 사용, 이를 통해 사용하는 방법 알아내는 중

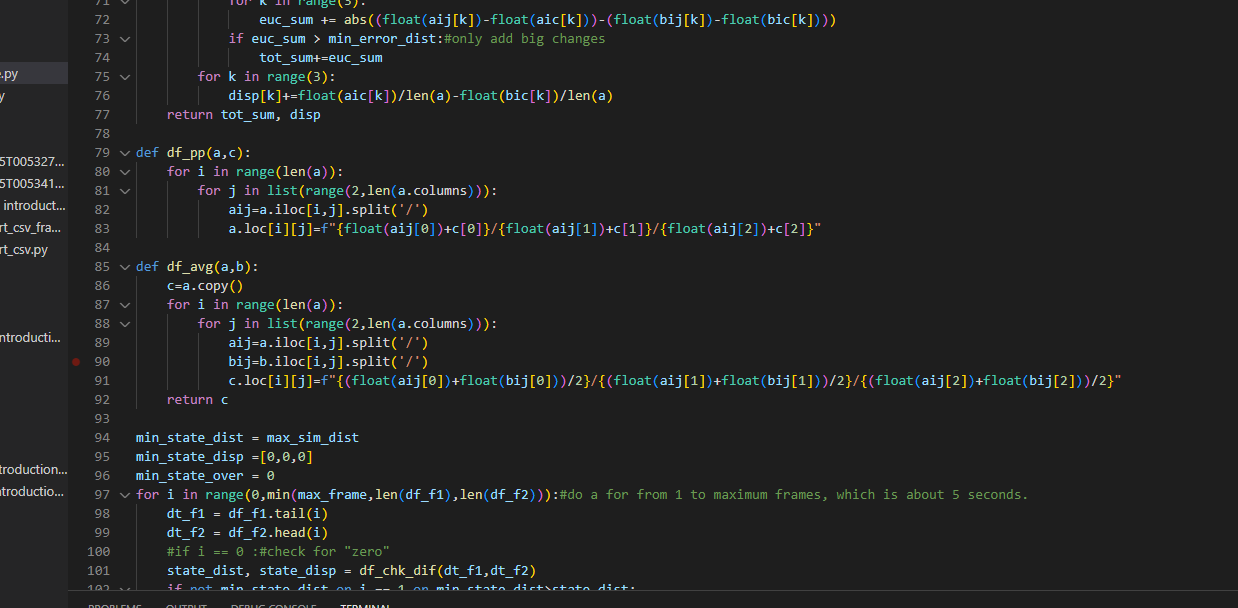


csv 형태로 추출 가능. 이때, `,`로 구분할 경우 이를 나중에 Unity에서 읽어들이거나 할 때 분류하기가 불편해 지기에, 이를 대신하기 위해 file string을 사용해서 “x/y/z” 형태로 각 점의 좌표를 저장함.  
5/15: 해당 코드를 특정 대상에 대해서 뿐만 아니라 특정 파일 디렉토리에 있는 파일들 전체를 처리할 수 있도록 전환. 이에 따라서 주어진 파일 디렉토리에 있는 비디오 영상을 순차적으로 처리할 수 있게 됨. 이렇게 처리된 파일들은 각각 비디오의 이름이 쓰여진 csv로 저장됨.  
5/16: <https://developers.google.com/mediapipe/solutions/vision/pose_landmarker>   
랜드마크 마킹하기, 이를 가져와서 순차적으로 마킹함으로써 사람이 읽기 쉽도록 함.  
매 프레임 timeframe 마킹하기, 맨 앞의 Frame Data를 매 프레임의 수를 집어넣어서 주어진 비디오에서 이러한 Detection이 늦게 시작되었을 경우 그 프레임에서부터 시작할 수 있도록 구현.

5/22: CSV 모션 예제 추출, 모션 데이터 비교에 집중하기 시작  


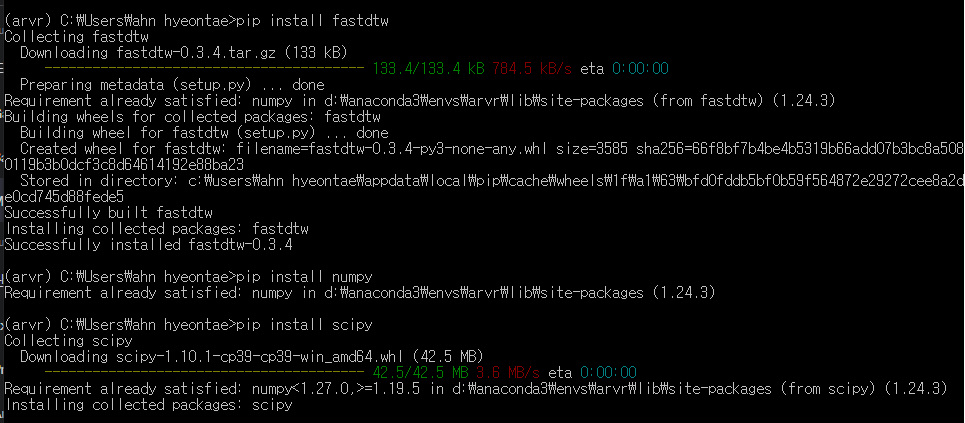
1. 추출된 모션 데이터를 다른 모션 데이터와 비교할 수 있는 방법 마련, 및 이의 중간점을 계산할 수 있도록 하기

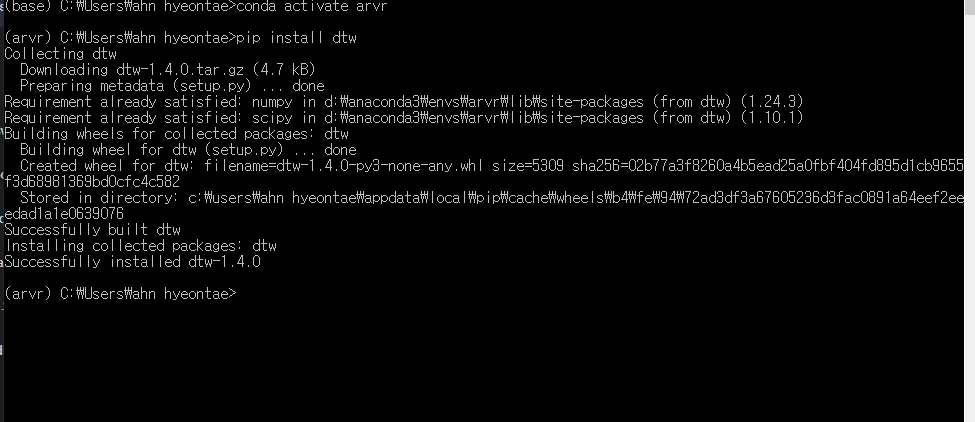
5/24: 우선적으로 단순 중점을 찾을 수 있는 방법을 마련. 기존 mediapipe csv extraction에서 필수적인 점들의 중점을 계산한 Centre column을 새로 제작하여 집어넣음.

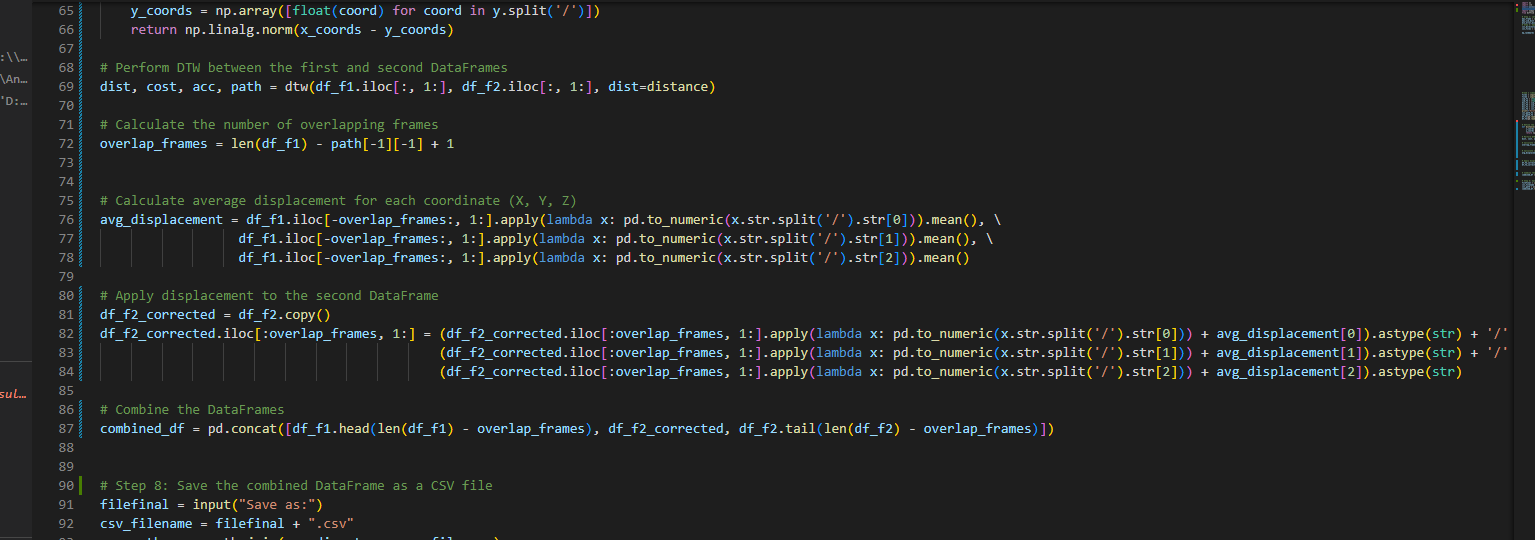
5/31: 기존 방식 테스트;   


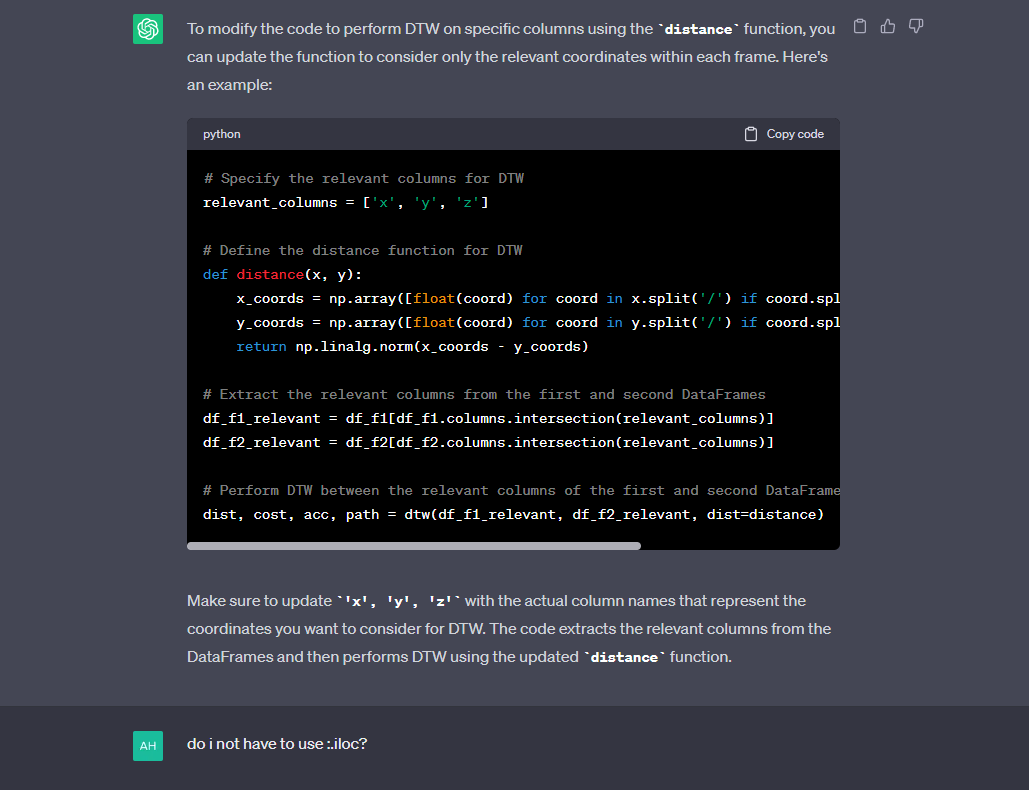
현재 CSV 파일 읽을 때 오류 발생. 이를 고치기 위해 ChatGPT와 상담 중. 이와 별개로, 최종 발표 도중에 알게 된 fastDTW(Dynamic Time Warping)의 적용을 검토

6/1: Scipy, FastDTW Install



6/3: DTW로 바꾸기  




DTW 적용 시도. split을 통해 저장되어 있는 x,y,z 좌표의 값을 각각 나눠서 두 점 사이의 거리를 계산할 수 있도록 함.  
이때, 이를 계산하기가 힘든 것 때문에 csv의 포맷을 바꾸는 것을 고려 중.  


ChatGPT를 통해 DTW 사용 시도  
6/4: DTW 에러, 다시 기본 방식으로 적용해보고자 함. 아쉽게도 Index Error가 발생, 이를 디버깅하기 위한 시간이 부족. DTW와 일반 Euclidian 방식으로 하는 것 둘 다 이러한 에러가 발생하는 것으로 보아, csv에서 data를 import할 때 생기는 오류인 것으로 추정됨.

1. A + A 후반부+B 시작부 + B 형태의 모션 데이터 만들어보기
2. 해당 모션 데이터를 Animation으로써 Unity Engine에 도입해보기

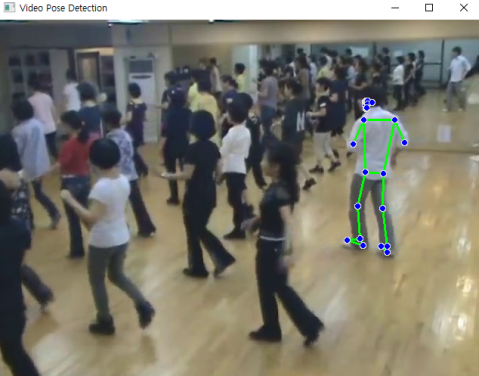
목적:

기존에 존재하는 춤의 ‘단위’별로 나눈 영상을 가지고 있을 때, 과연 이 두 영상을 어떻게 연결하면, 즉 춤의 포즈를 어떻게 연결하는 것이 제일 좋은지에 관하여 프로젝트를 진행하고 싶었다. 이떄 필요한 것은 우선적으로 춤의 포즈를 추출하는 프로그램(Mediapipe, ZED) 등, 그리고 추출된 춤의 포즈를 비교할 수 있는 프로그램(numpy 혹은 DTW 등으로 해결 가능), 그리고 전체의 춤을 몇 초 단위로 쪼개서 합성해야 제일 자연스러워지는지에 대한 프로그램(pandas를 통해 구현 가능할 듯 함)으로 구성될 것이다. 이때, 두 춤은 이러한 목적을 위해 정자세로 찍음으로써 크기 조절과 같은 문제는 없을 수 있개 할 수 있을 것이다.

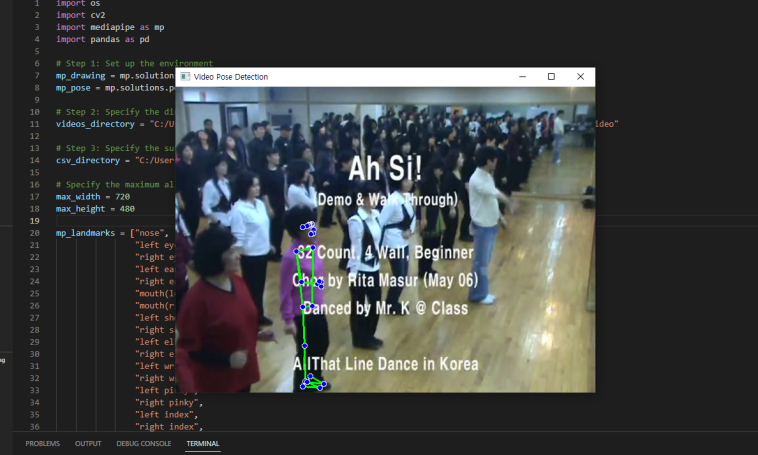
결과:

6/4: 꽤나 오랜 시간 동안 작업했으나 Mediapipe의 기능을 잘 파악 못하여 좀 늦어졌던 것이 아쉬움. 이와는 별개로, 개인 작업을 하다 보니 작업의 양과 질이 떨어지기에 두 Mediapipe 자료를 비교/합성하는 용도의 코드를 만들 지 못한 것이 아쉬움. 이번 학기에 혼자서만으로 아쉬운 양의 작업이었지만, 다른 이들과 소통하며 Mediapipe의 사용법, DTW, pandas, cv2와 같은 다양한 Python Library를 배울 수 있었다. 비록 처음 다루어 보는 것이라 많은 문제점이 있었지만 그러한 만큼 유익한 시간이었다고 생각한다.

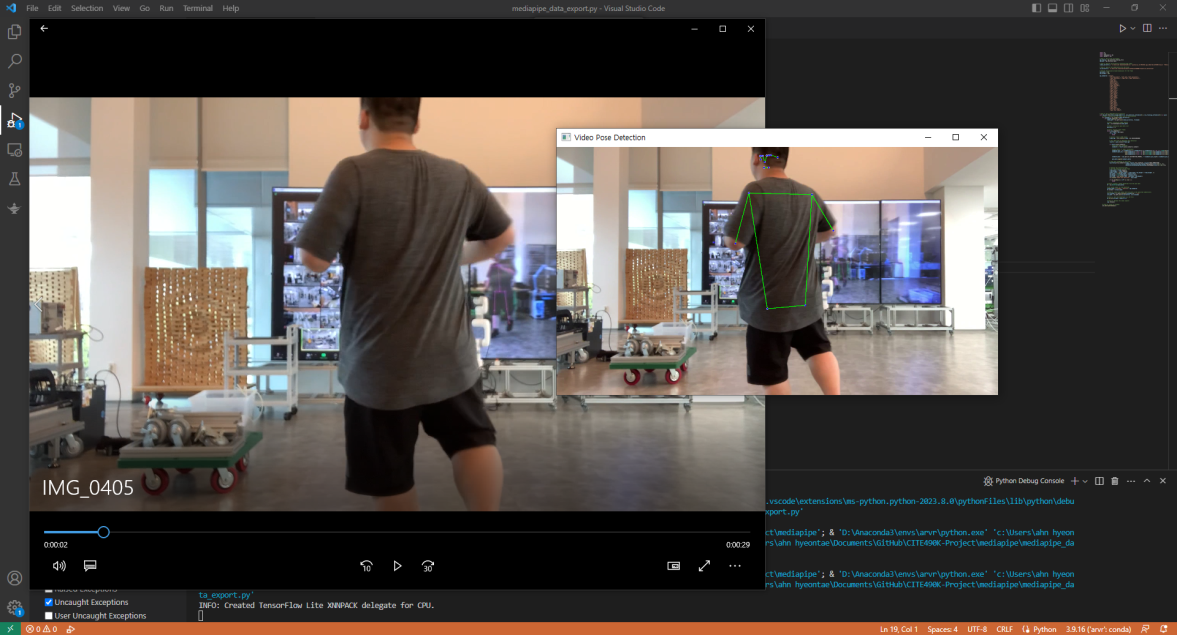
6/9: 추가 활동의 사진(화면 녹화가 제대로 되지 않아 개별로 녹화 및 제공)



Mediapipe data의 처리가 제대로 되는 것을 확인할 수 있음.

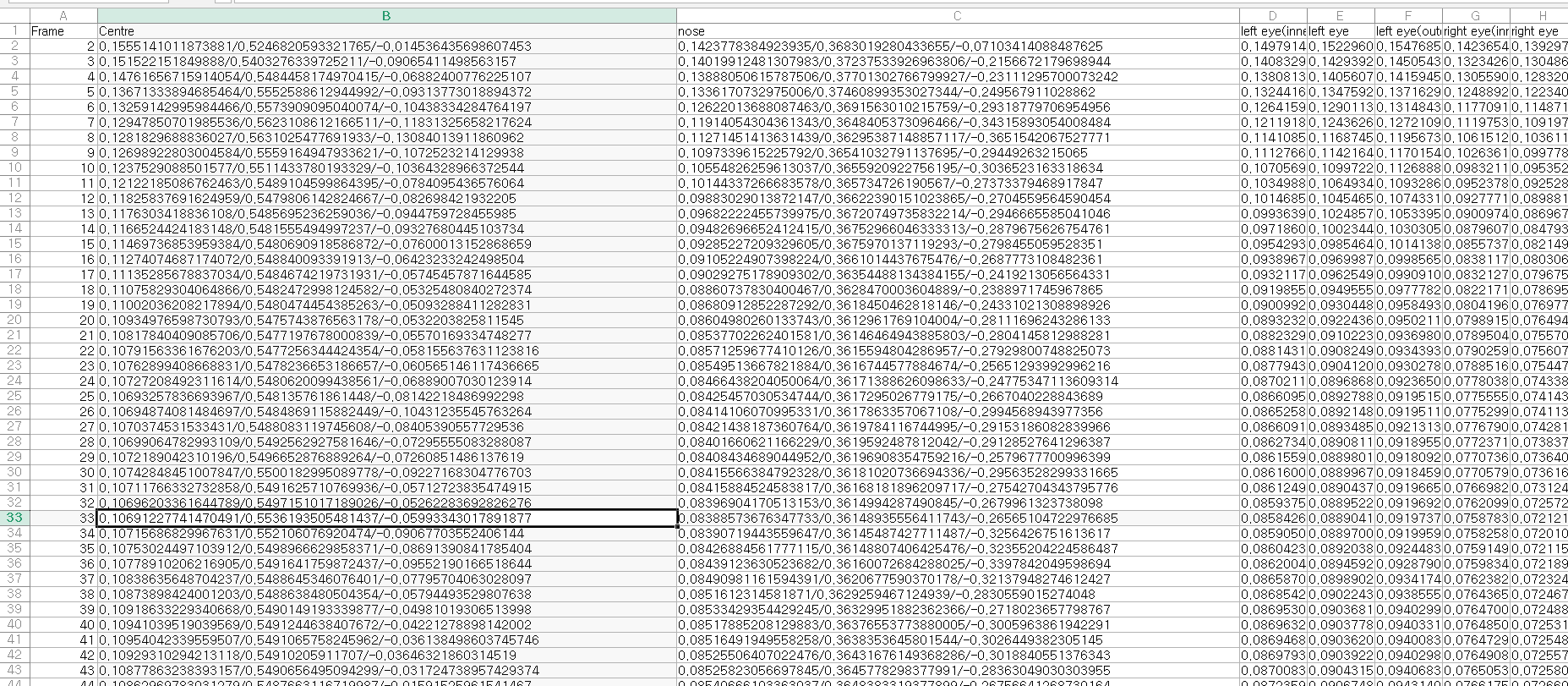


이와 함께, 디스플레이의 편의성을 위해 비디오가 특정 크기 이상일 경우 자동으로 알맞게 줄여주는 편의성 능력을 도입

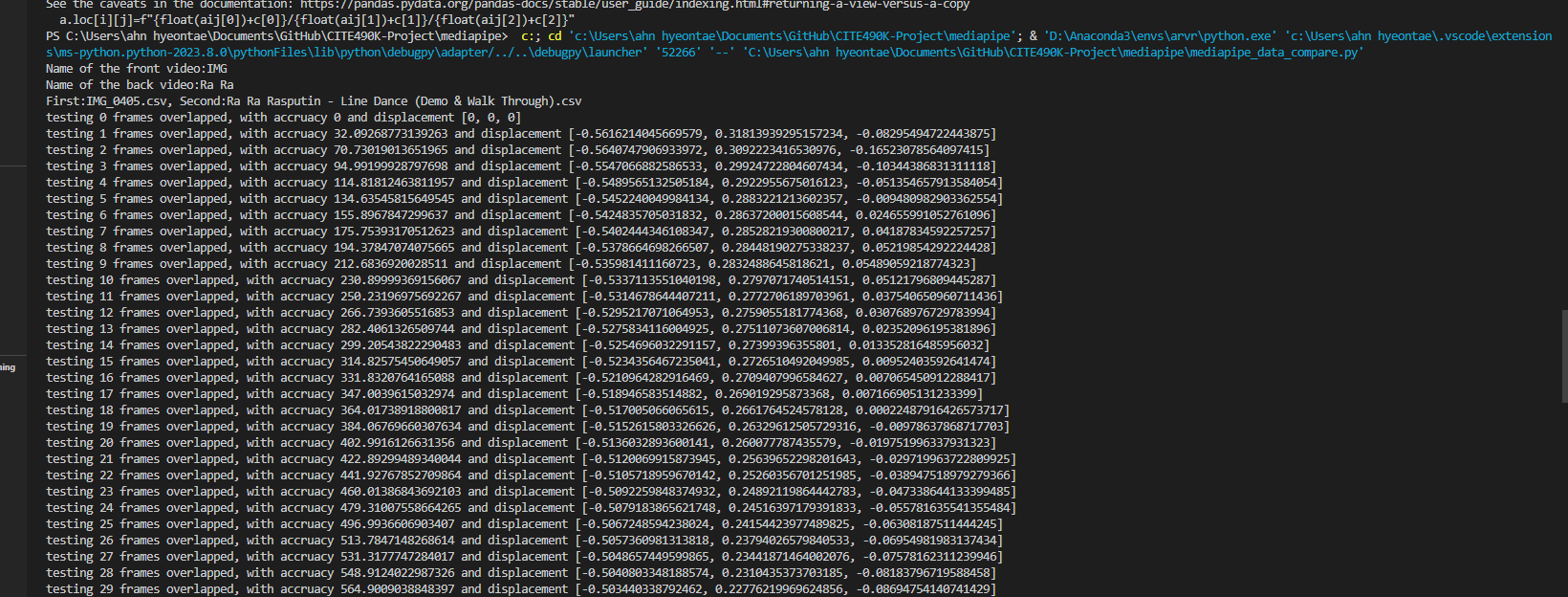


(왼쪽의 경우 기존 비디오, 오른쪽의 경우 Pose Detection을 돌리고 난 후의 비디오이다.)

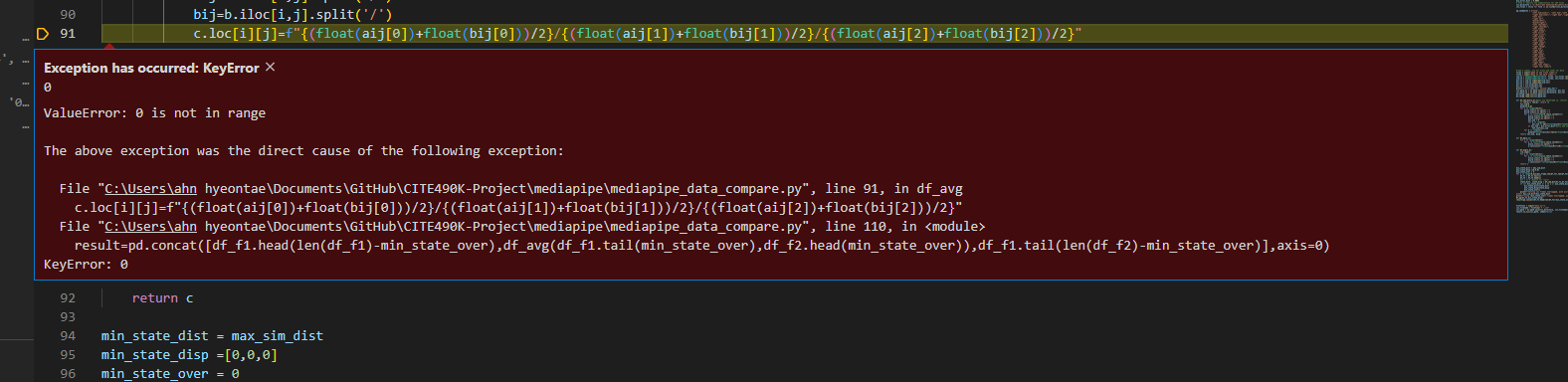
이후 처리를 거친 mediapipe는 x/y/z의 형태로 각 점의 프레임마다의 움직임이 기록된다.



두 mediapipe csv를 비교할 때에는 편의성을 위해 os library를 통해 자동으로 검색 및 색적을 할 수 있도록 디자인하였다.



위 2줄에서 입력을 받아 두 csv의 앞-뒤 차이를 분석하기 시작. 아쉽게도 이를 구현하는 것에 좀 오류가 발생해 겹치는 부분이 별로 없는 초반부는 잘 판단을 못하는 것으로 파악됨.



구현상 오류로 추정되나 아쉽게도 이 문제점을 찾아내지는 못함.

첨부된 3개의 동영상은 각각 mediapipe\_data\_export의 동작, 해당 python 파일의 동작 시에 cv2를 통해 보이는 비디오, mediapipe\_data\_compare의 동작 내용을 담고 있습니다.