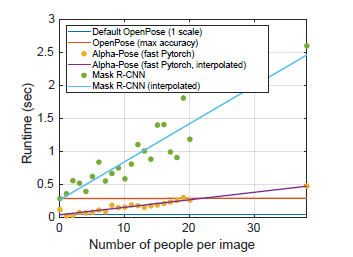
2) 모델 선정 이유

\*[그림1], [그림2]는 "OpenPose: realtime multi-person 2D pose estimation using Part Affinity Fields." 논문에서 인용하였습니다.

Multi-person pose estimation model 중에서 가장 유명한 모델이 Openpose와 Alphapose 그리고 Mask R-CNN이 있습니다. 그 중에서 아래 [그림1]을 보시면 Openpose모델이 하나의 이미지에서 사람의 수가 늘어나도 실행속도는 낮아지는 것을 알 수 있습니다. 그래서 저희는 Frame안에 많은 사람들이 들어와도 감지를 빠른 속도에서 할 수 있는 Openpose모델을 최종적으로 선택하였습니다.



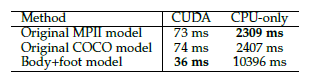
[그림 1]

3) 모델 생성

Mp4형식의 공부 동영상을 Openpose모델에 input으로 넣어주어 output으로 주요 신체부위의 좌표값은 JSON파일의 형식으로 Skeleton의 영상은 AVI형식으로 나오게 됩니다.

3-1) BODY+foot vs. COCO vs. MPII Model 비교

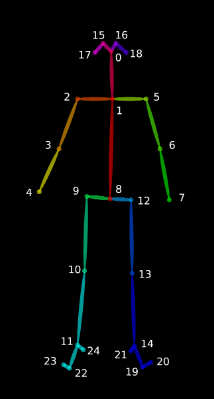
신체 특징을 뽑아내는 모델은 다음과 같이 3가지가 있습니다.



[그림2]

[그림2]를 보시면 MPII, COCO, Body+foot에서 CUDA와 CPU 환경일 때 각각 모델의 실행속도를 나타낸 표입니다. 보시다시피 Body+foot모델 즉, 신체와 발을 포함해 detection하는 모델이 나머지 두 모델들보다 속도가 약 2배정도 빠른 것을 확인할 수 있습니다. 그러한 이유로 저희는 CUDA환경내의 Body+foot모델을 최종적으로 선정하게 되었습니다.

3-2) Body Keypoints 추출

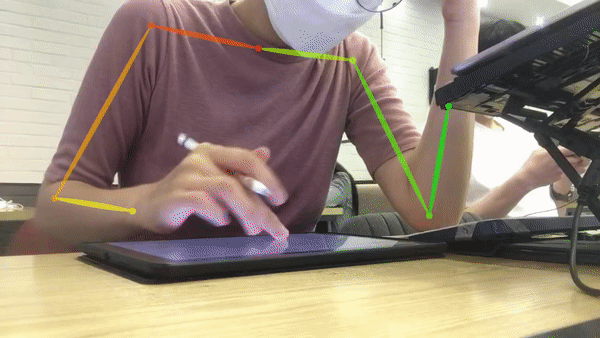
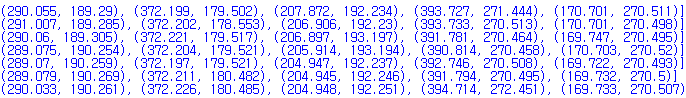


[그림3]

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 0 :Nose | 1 :Neck | 2 :RShoulder | 3 :RElbow | 4 :RWrist |
| 5 :LShoulder | **6 :LElbow** | **7 :LWrist** | **8 :MidHip** | **9 :RHip** |
| 10 :RKnee | **11 :Rankle** | **12 :LHip** | **13 :LKnee** | **14 :LAnkle** |
| 15 :REye | **16 :LEye** | **17 :REar** | **18 :LEar** | **19 :LBigToe** |
| 20 :LSmallToe | **21 :LHeel** | **22 :RBigToe** | **23 :RSmallToe** | **24 :RHeel** |

[그림4]

앞에서 언급했듯이 body+foot모델을 사용했고 그 구조와 신체의 keypoints들은 [그림3] [그림4]에서 볼 수 있습니다. 여기서 저희는 상반신 모델만을 이용해 공부 자세를 정의하기 때문에 필요한 keypoints의 좌표값들만 추출하였습니다.

[그림5] [그림6]

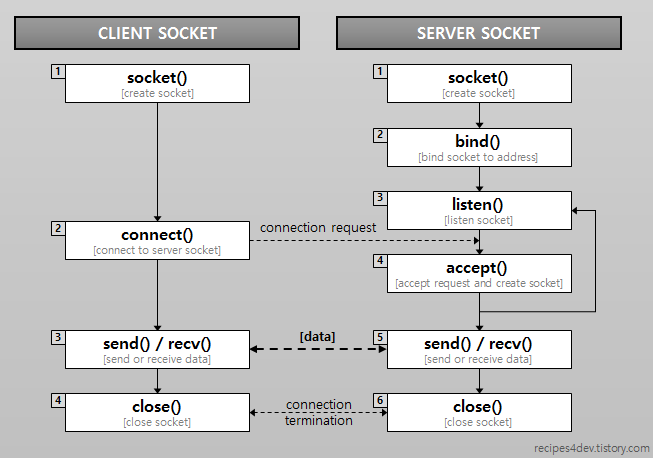
[그림5]에서 보이듯이 차례로 각각 왼쪽, 오른쪽 어깨와 목 그리고 왼쪽, 오른쪽 손목을 frame에서 위치 좌표값(x,y)의 값으로 데이터를 뽑아 [그림6]과 같이 리스트안에 튜플이 들어간 형식으로 가공해주었습니다. 그리하여 매 frame마다 신체의 위치를 나타내는 좌표값들이 데이터로 생성되고, 그 값들을 활용하여 공부 자세를 추정하는 모델을 생성하고자 하였습니다. 하지만 그러기 위해서는 공부자세를 정의하고 기준에 맞는 임계치들을 정의하는 과정이 필요했습니다. 그 부분은 아래에서 설명하겠습니다.

* **소켓 통신**

1. 소개

Server와 Client가 특정 Port를 통해 연결을 성립하고 있어 실시간으로 양방향으로 통신을 하는 방식을 말합니다. 소켓에는 기본적으로 TCP와 UDP의 통신 방법이 존재합니다. 그 중 저희가 채택한 것은 TCP로 그 구조에 대해서 알아보겠습니다.

1. TCP 구조



[그림7]

[그림7]과 같이 다음과 같이 이루어져있으며 구조의 흐름은 다음과 같습니다.

1) socket함수를 이용해 객체를 생성 2) Server에서 포트를 열고 값을 기다리기 위해서 bind함수 사용 3) bind함수에서 호스트 이름과 포트번호를 입력 받음 4) listen함수를 사용해 Client가 해당 포트에 들어오기까지 대기 5) Client가 해당 포트로 접속하면 accept함수를 사용해 값을 받음 6) 이후에 Server에서 다시 Client에 send함수를 사용해 전달 7) Client에서 다시 recv함수를 사용해 해당 값을 받음

이와 같이 서로 양방향의 소통을 하면서 Server와 Client간의 데이터를 주고받을 수 있는 플랫폼이 Socket통신이며 저희는 APP(즉, Server)에 데이터를 전송하는 방식으로 이러한 흐름으로 TCP를 사용하였습니다.