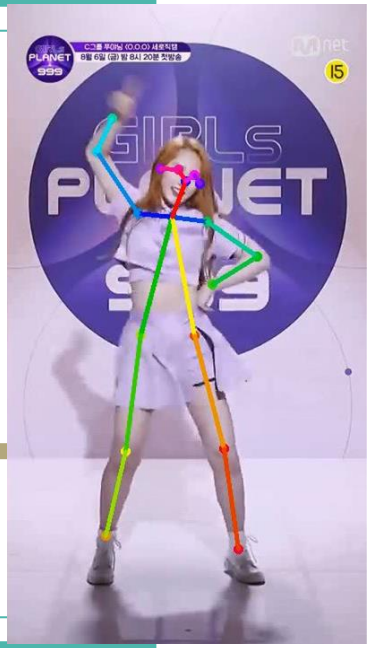
ECE595 CVES

DancePose Assistant: human pose estimation and skeleton-based dance evaluation

Group 5
Tsam Ying Lam, Harry Lin, Yinghan Long

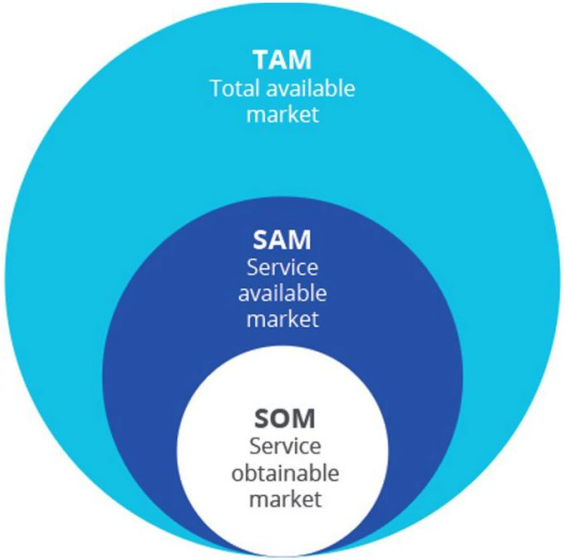


Player

Teacher

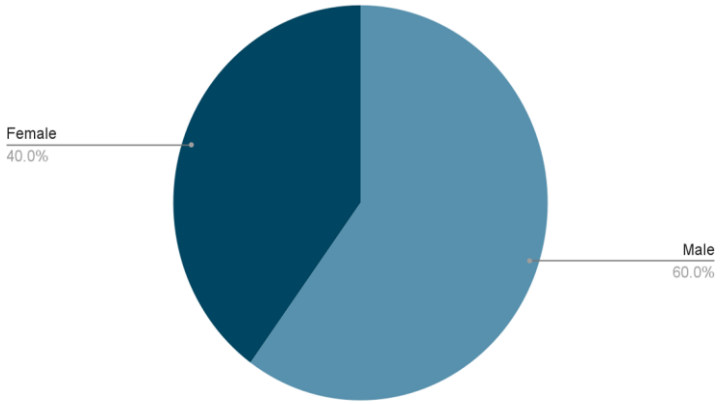
Problem Statement

- The problem
  - Inaccurate human pose estimation in motion sensing games
  - Lack pose correction and tutoring
- Impact
  - 61 million people
- Market
  - TAM - 2.69 billion: video game players worldwide
  - SAM - 156.4 million: US video game players
  - SOM - 61 million: motion sensing game players



Customer Discovery

- Who are they? total 15
  - 60% male, 40% female
  - >70% mid-class income
  - 23 - 30 years old
  - 66.7% motion sensing game players

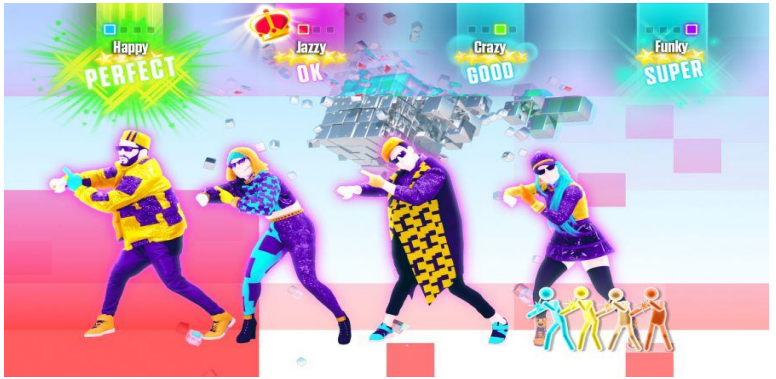


Customer Discovery

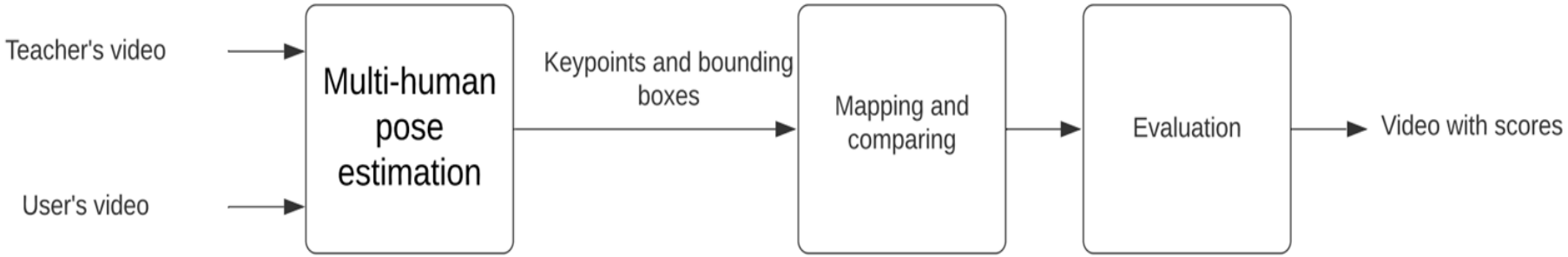
- How do they deal with inaccurate pose estimation?
  - Reset sensors
  - Repeat their actions
- Their ideal motion sensing games?
  - More accurate estimation
  - Good haptic feedback

Existing Solutions

- Motion sensors and accelerometers
  - Position, acceleration, angular velocity, etc.
  - Easy to “cheat”
- Position detection systems
  - Computer vision
  - Limited detection



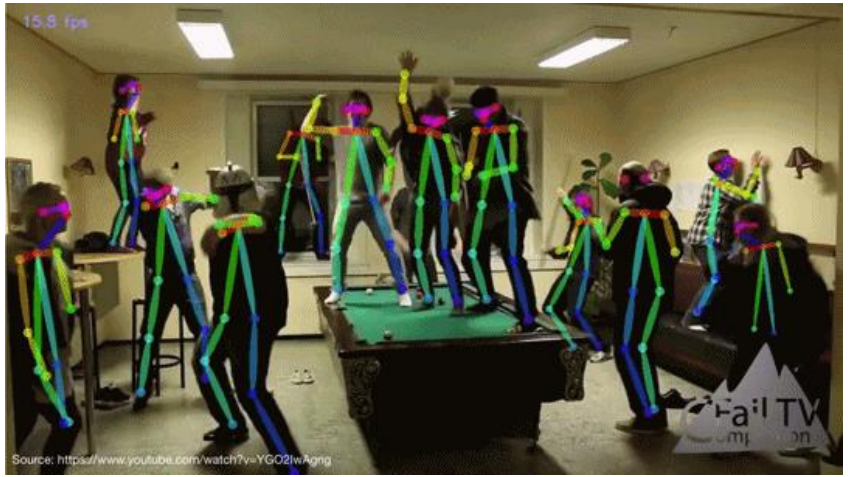
Proposed Solution



Score scale: Perfect, excellent, good, miss

Technology Used: OpenPose

- Open pose
  - VGG19
  - Bottom-up analysis
  - Greedy phrasing algorithm



model name	mAP on COCO val2017 Dataset	Inference Time (GPU)
OpenPose Pytorch [1]	0.653	8.8 fps with 19 people

[1] https://github.com/tensorboy/pytorch_Realtime_Multi-Person_Pose_Estimation

Key points

Nose = 0

Neck = 1

RShoulder = 2

RElbow = 3

RWrist = 4

LShoulder = 5

LElbow = 6

LWrist = 7

RHip = 8

RKnee = 9

RAnkle = 10

LHip = 11

LKnee = 12

LAnkle = 13

REye = 14

LEye = 15

REar = 16

LEar = 17

Background = 18 (Not used)

- source from MNET

Expected Functions

- Multi-person dance pose detection in videos
- Pose evaluation with given benchmark by mapping key points
- Realization with limited computation power on RPi

Projecting key points for comparison

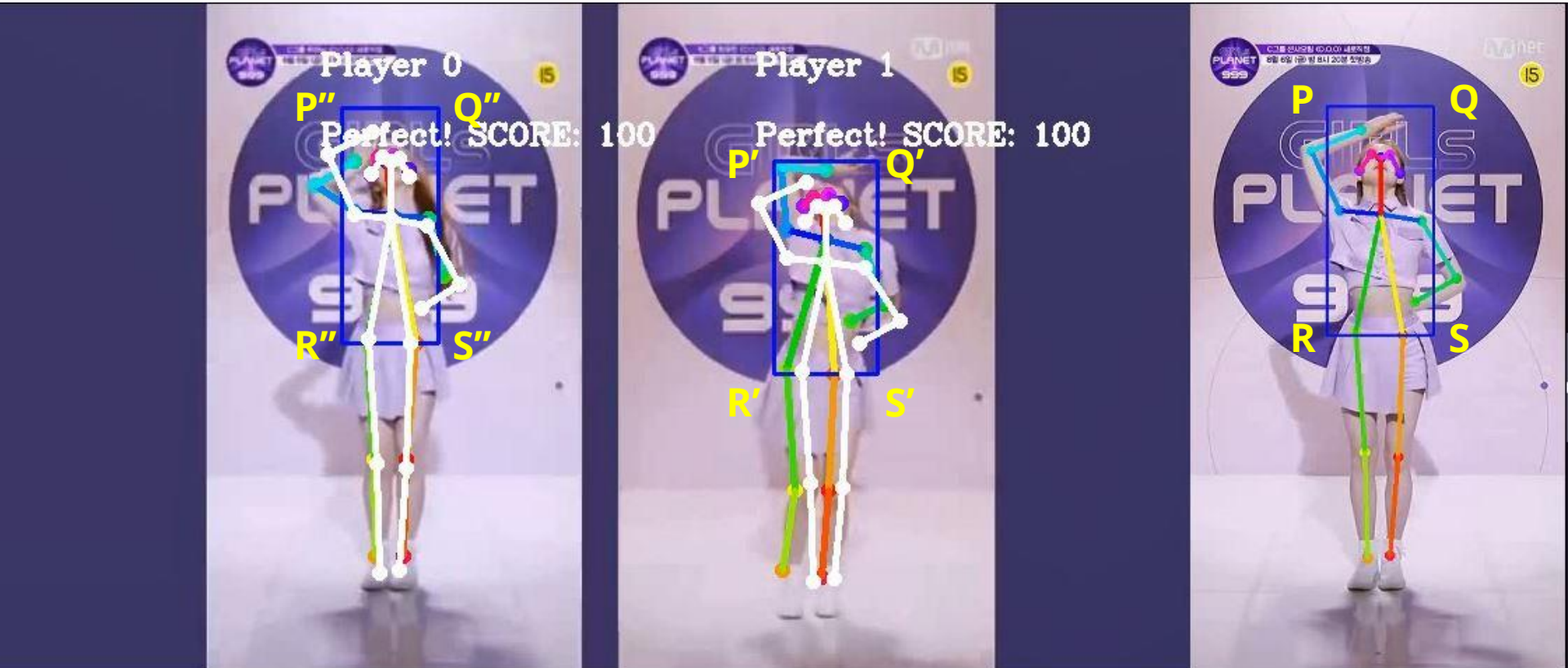
Find homography with corresponding points

Given a point x in a planar scene and its corresponding pixel x' in the image plane, we can write $x' = Hx$, assuming that $x = (x, y, 1)^T$, $x' = (x'_1, x'_2, x'_3)^T$, with $H = \begin{pmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{pmatrix}$. h_{33} can be set to 1 since we use homogeneous coordinates.
Let $x' = x'_1/x'_3, y' = x'_2/x'_3$. Then we can get

$$x' = \frac{h_{11}x + h_{12}y + h_{13}}{h_{31}x + h_{32}y + 1}$$

$$y' = \frac{h_{21}x + h_{22}y + h_{23}}{h_{31}x + h_{32}y + 1}$$

Using bounding boxes as corresponding points



Solve homography H

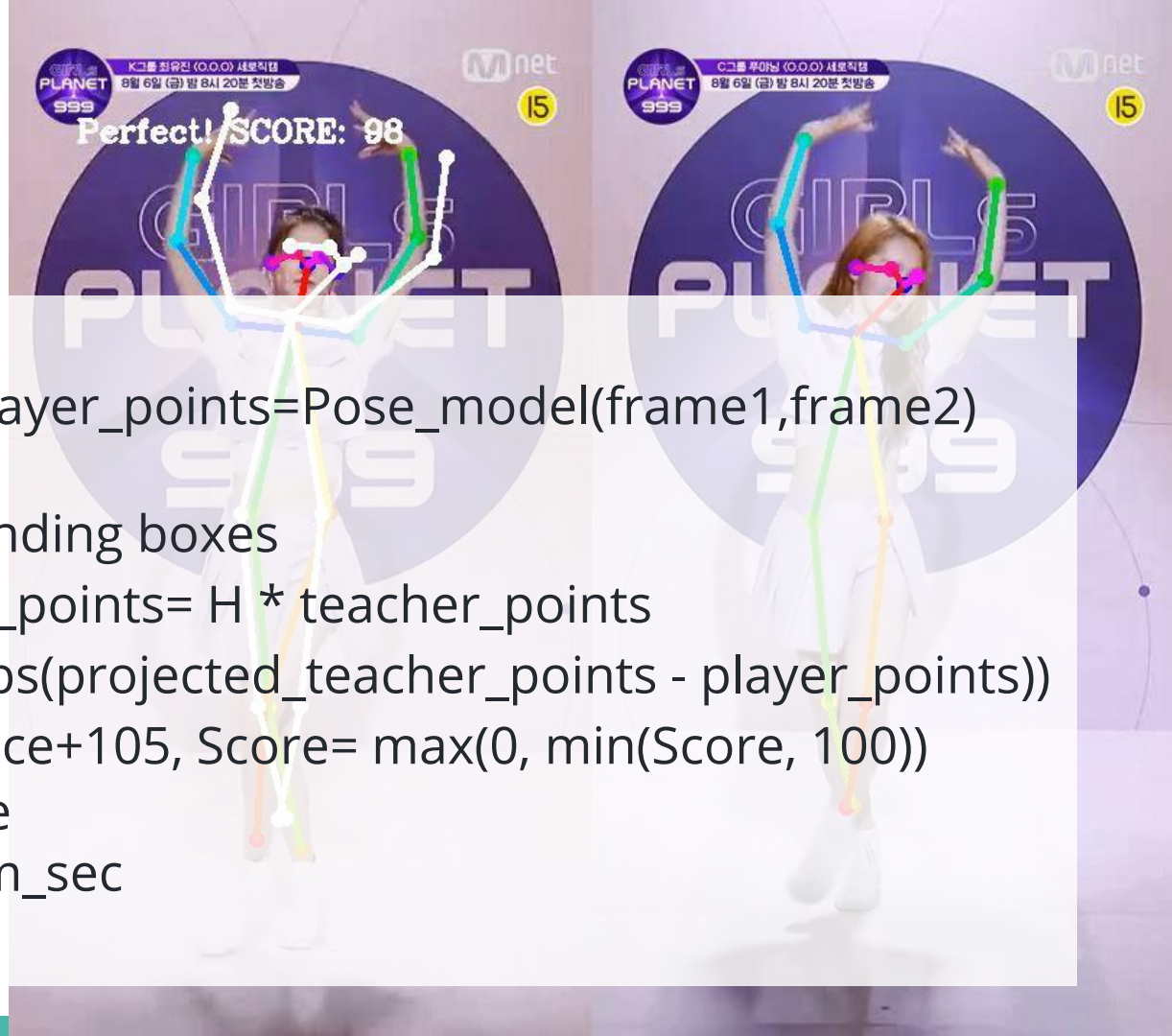
To solve H, we use four groups of corresponding points (**PQRS of bounding boxes**) to solve 8 unknown parameters in H.

$$\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x_1x_1' & -y_1x_1' \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -x_1y_1' & -y_1y_1' \\ x_2 & y_2 & 1 & 0 & 0 & 0 & -x_2x_2' & -y_2x_2' \\ 0 & 0 & 0 & x_2 & y_2 & 1 & -x_2y_2' & -y_2y_2' \\ x_3 & y_3 & 1 & 0 & 0 & 0 & -x_3x_3' & -y_3x_3' \\ 0 & 0 & 0 & x_3 & y_3 & 1 & -x_3y_3' & -y_3y_3' \\ x_4 & y_4 & 1 & 0 & 0 & 0 & -x_4x_4' & -y_4x_4' \\ 0 & 0 & 0 & x_4 & y_4 & 1 & -x_4y_4' & -y_4y_4' \end{bmatrix} \cdot \begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \end{bmatrix} = \begin{bmatrix} x_1' \\ y_1' \\ x_2' \\ y_2' \\ x_3' \\ y_3' \\ x_4' \\ y_4' \end{bmatrix}$$

$$\Rightarrow A \cdot h = b$$

$$\Rightarrow h = A^{-1}b$$

Dance Pose Evaluation



For a frame every second:

- Teacher_points, player_points=Pose_model(frame1,frame2)

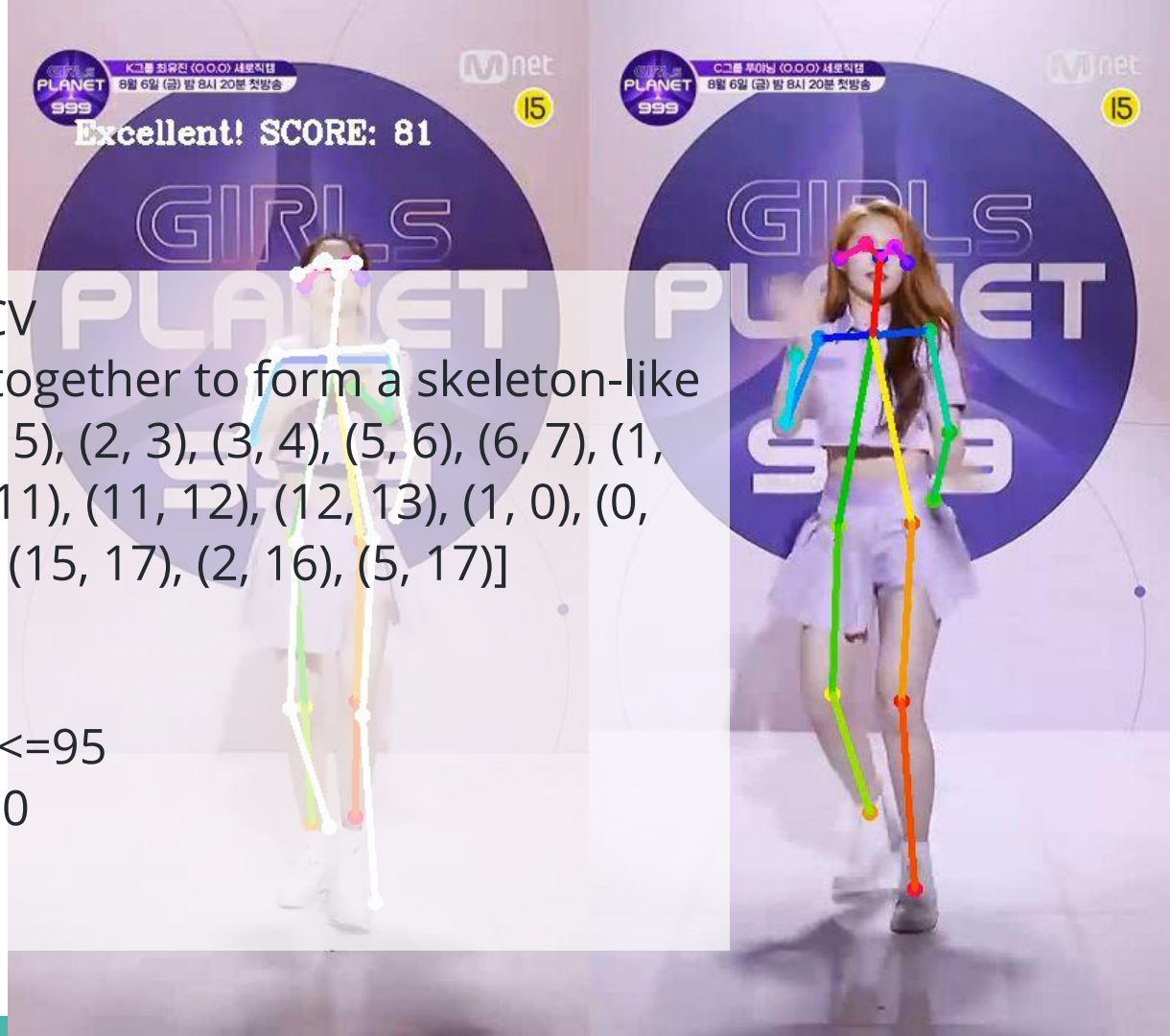
For each player:

- Solve H using bounding boxes
- Projected_teacher_points= H * teacher_points
- distance= mean(abs(projected_teacher_points - player_points))
- Score= -0.5*distance+105, Score= max(0, min(Score, 100))
- total_score+=score

avg_score=total score/num_sec

Video Rendering

- Plot poses using OpenCV
  - Connect the joints together to form a skeleton-like structure: [(1, 2), (1, 5), (2, 3), (3, 4), (5, 6), (6, 7), (1, 8), (8, 9), (9, 10), (1, 11), (11, 12), (12, 13), (1, 0), (0, 14), (14, 16), (0, 15), (15, 17), (2, 16), (5, 17)]
- Show scores
  - Perfect: score>95
  - Excellent: 80<score<=95
  - Good: 60<score<=80
  - Miss: score<=60



Demo Video

Conclusion

- Implement all expected functions
- Run on CPU, GPU, and Raspberry Pi
- Need further improvement on the speed
  - Quantization
  - Lightweight models

model name	Raspberry Pi Run Time (s)	CPU Run Time (s)	GPU Run Time (s)
DancePose Assistant	124.32	7.86	0.94