4.3 Computer Vision

Computer vision is an interdisciplinary scientific field that deals with how computers can gain high-level understanding from digital images or videos. in our case we are using computer vision technology for understanding how we can use the human pose coordinates for deciding a perticular pose is correct or not.

4.3.1 Mediapipe Blazepose

MediaPipe BlazePose is a ML solution for high-fidelity body pose tracking, inferring 33 3D landmarks and background segmentation mask on the whole body from RGB video frames utilizing our BlazePose research that also powers the ML Kit Pose Detection API. Current state-of-the-art approaches rely primarily on powerful desktop environments for inference, whereas method achieves real-time performance on most modern mobile phones, desktops/laptops, in python and even on the web.

ML Pipeline

The solution utilizes a two-step detector-tracker ML pipeline, proven to be effective in the MediaPipe Hands and MediaPipe Face Mesh solutions. Using a detector, the pipeline first locates the person/pose region-of-interest (ROI) within the frame. The tracker subsequently predicts the pose landmarks and segmentation mask within the ROI using the ROI-cropped frame as input. Note that for video use cases the detector is invoked only as needed, i.e., for the very first frame and when the tracker could no longer identify body pose presence in the previous frame. For other frames the pipeline simply derives the ROI from the previous frame's pose landmarks.

Models

The detector is inspired by mediapipe's own lightweight BlazeFace model, used in MediaPipe Face Detection, as a proxy for a person detector. It explicitly predicts two additional virtual keypoints that firmly describe the human body center, rotation and scale as a circle. Inspired by Leonardo's Vitruvian man, we predict the midpoint of a person's hips, the radius of a circle circumscribing the whole person, and the incline angle of the line connecting the shoulder and hip midpoints.

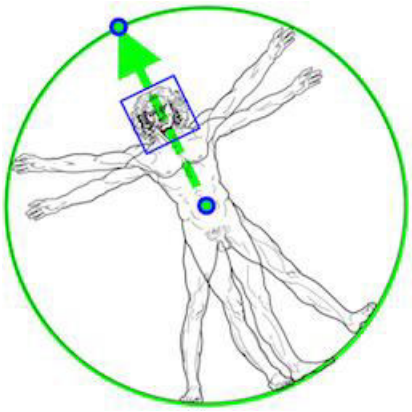
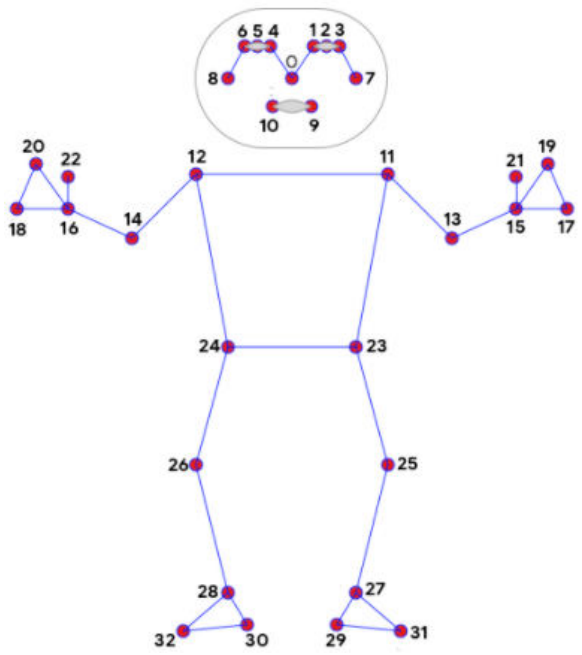


Fig. Vitruvian man aligned via two virtual keypoints predicted by BlazePose detector in addition to the face

bounding box.

Pose Landmark Model (BlazePose GHUM 3D)

The landmark model in MediaPipe Pose predicts the location of 33 pose landmarks (see figure below).



0. nose
1. left_eye_inner
2. left_eye
3. left_eye_outer
4. right_eye_inner
5. right_eye
6. right_eye_outer
7. left_ear
8. right_ear
9. mouth_left
10. mouth_right
11. left_shoulder
12. right_shoulder
13. left_elbow
14. right_elbow
15. left_wrist
16. right_wrist
17. left_pinky
18. right_pinky
19. left_index
20. right_index
21. left_thumb
22. right_thumb
23. left_hip
24. right_hip
25. left_knee
26. right_knee
27. left_ankle
28. right_ankle
29. left_heel
30. right_heel
31. left_foot_index
32. right_foot_index

Fig. pose landmarks.

Optionally, MediaPipe Pose can predicts a full-body segmentation mask represented as a two-class segmentation (human or background).

Output

A list of pose landmarks. Each landmark consists of the following:

x and y: Landmark coordinates normalized to [0.0, 1.0] by the image width and height respectively.

z: Represents the landmark depth with the depth at the midpoint of hips being the origin, and the smaller the value the closer the landmark is to the camera. The magnitude of z uses roughly the same scale as x.

visibility: A value in [0.0, 1.0] indicating the likelihood of the landmark being visible (present and not occluded) in the image.

Another list of pose landmarks in world coordinates. Each landmark consists of the following:

x, y and z: Real-world 3D coordinates in meters with the origin at the center between hips.

visibility: Identical to that defined in the corresponding pose_landmarks.

4.3.2 Video Input

we take the live web-cam video field as an input and after implementing the deep learning on this video we will get some raw data. then we prerocess this data and use some mathematical rules. then we train a model using machine learning and neural networks. after training the model we convert this model in tfjs model. for implementing in the browser.

4.3.3 Video Output

Computer Vision is one of the most important part in our scenario. the overall input and output is totally based on computer vision. after training and conversion of out model. we took this model to a suitable environment. ie on browser. in our case we uploaded this converted model to github in order to access globally. then in our front end code we write code for accessing web-cam and displaying it on browser. after loading the model successfully we take the input video feed and preprocess it using some mathematical rules. for this we write some functions which will do mathematical operations.

after applying these functions, we give the data to our trained models continuesly which will take some more ram from the system. after doing either classification or regression we will get the final output as classification or data points for correct position. then according to this result we will display the suggessions in our web-app.