Realtime pose replication for 3D Avatars Using Human Pose Estimation

Sai Tarun Sathyan , Amanraj Lnu

Abstract

Realtime pose estimation has become an increasingly popular field of study due to its wide range of applications, including in the fields of motion capture, sports analysis, gaming, and human-computer interaction. However, traditional motion capture techniques involve expensive equipment, such as motion capture suits and studios, which limit its accessibility.

The goal of this project is to develop an affordable motion capture system that uses real-time pose estimation to replace the expensive equipment required for traditional motion capture. We aim to compare and evaluate the accuracy and performance of various state-of-the-art pose estimation libraries, including PoseNet, MediaPipe, and OpenPose, and determine which is best suited for real-time motion capture applications.

We recorded human movements using a simple webcam and tracked key points using each of the three libraries. We evaluated the performance and accuracy of each library by comparing the results of the tracked key points with ground truth data. Our evaluation criteria included frame rate, accuracy, and robustness in different lighting conditions and camera angles.

Once we identified the most accurate and efficient pose estimation library, we used the tracked key points to animate a 3D avatar in real time using software such as Unreal Engine or Blender. Our approach provides an affordable motion capture solution that avoids the need for expensive equipment and studios, making motion capture accessible to a wider audience.

The findings of this project are significant for developers and creatives looking to incorporate motion capture into their projects but lack the budget for expensive equipment. By leveraging the power of real-time pose estimation, our system provides a cost-effective alternative to traditional motion capture techniques. We also provide visualisations of the tracked key points and the resulting animations, allowing developers to easily understand the capabilities and limitations of each library.

In conclusion, our project provides a comprehensive evaluation of the performance and accuracy of various state-of-the-art pose estimation libraries for real-time motion capture applications. By developing an affordable motion capture system using real-time pose estimation, we hope to make motion capture accessible to a wider audience and contribute to the democratisation of this technology.

1 INTRODUCTION

Human pose estimation is a computer vision technique that involves detecting and tracking the key points of the human body in images or videos. The ability to accurately estimate human pose has numerous applications, including in the fields of motion capture, sports analysis, gaming, and human-computer interaction. Pose estimation algorithms have come a long way in recent years, with many state-of-the-art methods achieving high accuracy and real-time performance.

There are several types of pose estimation techniques, including 2D pose estimation, 3D pose estimation, and multi-person pose estimation. 2D pose estimation involves detecting and tracking the key points of the human body in a 2D image, while 3D pose estimation involves estimating the 3D positions of the key points in space. Multi-person pose estimation extends 2D or 3D pose estimation to detect and track multiple people simultaneously.

The key points that are typically tracked in human pose estimation include the head, neck, shoulders, elbows, wrists, hips, knees, and ankles. Some methods also track additional key points, such as the eyes, ears, and fingers. There are several state-of-the-art pose estimation algorithms such as:

- PoseNet - PoseNet is a deep learning-based approach that uses a convolutional neural network to estimate pose in real-time. It is optimised for single-person pose estimation and can run efficiently on mobile devices. Some of its strengths include its real-time performance, high accuracy, and ability to operate on low-power devices. However, its performance can be limited in multi-person scenarios, and it may struggle with occlusions or complex poses.
- MediaPipe - MediaPipe is a cross-platform framework that provides high accuracy and robustness for pose estimation in various lighting conditions and camera angles. It supports both single-person and multi-person pose estimation and provides APIs for integrating pose estimation into applications. Some of its strengths include its robustness to challenging conditions, high accuracy, and multi-person support. However, it may require more computational resources than some other algorithms and may not perform as well on low-powered devices.
- OpenPose - OpenPose is a popular open-source library that provides state-of-the-art performance for multi-person pose estimation. It uses a deep learning-based approach and can track up to 135 keypoints per person. Some of its strengths include its high accuracy, multi-person support, and ability to handle occlusions and complex poses. However, its real-time performance may be limited, and it may require a powerful GPU for optimal performance.
- MoveNet - MoveNet is a lightweight pose estimation model that is optimised for mobile and embedded devices. It supports both single-person and multi-person pose estimation and can detect body, hand, and facial landmarks. Some of its strengths include its real-time performance, low computational cost, and ability to detect multiple types of landmarks. However, its accuracy may be lower than some other algorithms, especially in challenging lighting conditions or with occlusions.
- MaskRCNN - Mask R-CNN is a deep learning-based method for multi-person pose estimation that predicts both the object mask and pose keypoints. It extends the R-CNN framework by adding a mask prediction branch to extract features for each individual person and achieve high accuracy.
- Yolov7 Pose Estimator - This model uses a single neural network to detect people in an image and estimate their key points simultaneously, achieving high accuracy and real-time performance. It has a relatively small model size and can be run on low-power devices, making them suitable for real-world applications.

To compare the performance of these algorithms, we evaluated the accuracy and speed of each method on a benchmark dataset, including the number of keypoints tracked, accuracy, and speed. We provide a table summarising these results to aid developers and researchers in selecting the most appropriate pose estimation method for their application.

In this paper, we present a comprehensive evaluation of state-of-the-art pose estimation algorithms and demonstrate their practical use in real-time 3D avatar animation. By comparing the performance of different algorithms, we aim to provide insights into the strengths and weaknesses of each approach and help developers and researchers make informed decisions in selecting the most appropriate algorithm for their needs.

2 RELATED WORK

1. **BlazePose(On-device Real-time Body Pose tracking) by Valentin Bazarevsky, Ivan Grishchenko, Karthik Raveendran, Tyler Zhu, Fan Zhang, Matthias Grundmann**:BlazePose is a convolutional neural network (CNN) architecture designed specifically

for human pose estimation in real-time on mobile devices. It is lightweight and capable of producing 33 body keypoints for a single person with over 30 frames per second on a Pixel 2 phone, making it ideal for real-time applications such as fitness tracking and sign language recognition.THis paper include's a new body pose tracking technique and a lightweight neural network that uses both heatmaps and regression to estimate keypoint coordinates.

2. **Integrating Human Body MoCaps into Blender using RGB Images by Jordi Sanchez-Riera and Francesc Moreno-Noguer:**This study's methodology attempts to simplify and lower the price of conventional Motion Capture (MoCap) systems. The authors suggest an algorithm that can accomplish this with just a single Red-Green-Blue (RGB) camera, acting as a MoCap system.

3. **Application of 3D Human Pose Estimation for Motion Capture and Character Animation by Anastasiia Borodulina:**The thesis presents a framework that can create motion capture (mocap) data from a regular RGB video and use it to animate a 3D character based on the person's movement in the original video.

4. **"Real-time 3D Human Pose Estimation with a Single RGB Camera" by S. Moon et al.:** This paper proposes a real-time 3D pose estimation method using a single RGB camera. The system uses a deep neural network to estimate the user's pose and then maps the estimated pose to a 3D avatar to animate it in real-time. The authors report that their system achieves high accuracy and can run at real-time frame rates.

3 PROBLEM DESCRIPTION

The problem addressed in this project is the high cost of motion capture technology, which typically requires specialised suits and studio setups. The goal is to develop a more cost-effective approach using real-time pose estimation to animate a 3D avatar without the need for expensive equipment. The project will compare and evaluate different state-of-the-art pose estimation libraries, including PoseNet, MediaPipe, and OpenPose, to determine which is the most accurate and efficient at real-time pose estimation. The input data for this project will be video recordings from a simple webcam, which will capture the user's movements for real-time animation.

4 PROPOSED SOLUTION

The proposed solution for this project is to use real-time pose estimation to track the movement of a human body and then transfer the tracked key points to a 3D avatar to animate it in real-time. The project will compare and evaluate the performance of different state-of-the-art pose estimation libraries such as PoseNet, MediaPipe, and OpenPose to determine the most accurate and efficient approach for real-time pose estimation. The project will use a simple webcam to capture the user's movements, eliminating the need for expensive motion capture suits and studios. After identifying the most efficient pose estimation method, the project will port the tracked keypoints to either Unreal Engine or Blender to animate the 3D avatar in real-time. The output will be a cost-effective motion capture system that can be used in a variety of contexts, such as gaming, film, and virtual reality without the need for expensive equipment or studios.

These are the pose estimators we will be testing : Pose Net Media Pipe Open Pose Move Net MaskRCNN Yolov7 pose estimator

After finding the most efficient one out of the list we will try to attach this to unreal engine/blender to obtain key points of a person standing in front of a webcam and map these key points onto a 3d character.

5 EXPERIMENT RESULT

5.1 TESTING

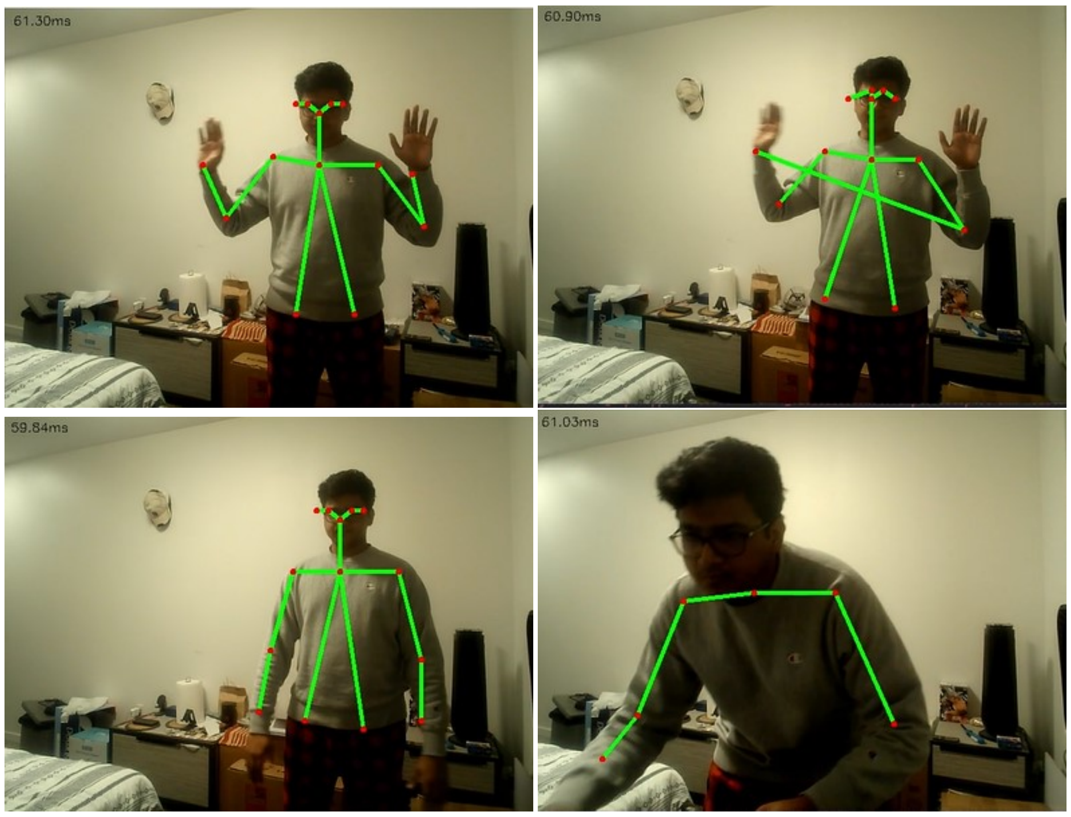


Figure 1: Open Pose Testing

5.1.1 Open Pose Pose Estimation

Device used: Works on CPU and GPU
Key Points Tracked: Returns 17-135 key points

- Works well on CPU, works in real time using CPU alone
- The detections are mostly accurate, breaks apart sometimes during overlapping/complex movements
- Fails to detect legs while wearing checked pants / due to bad lighting
- Fails to detect limbs while close to the camera (waist up)

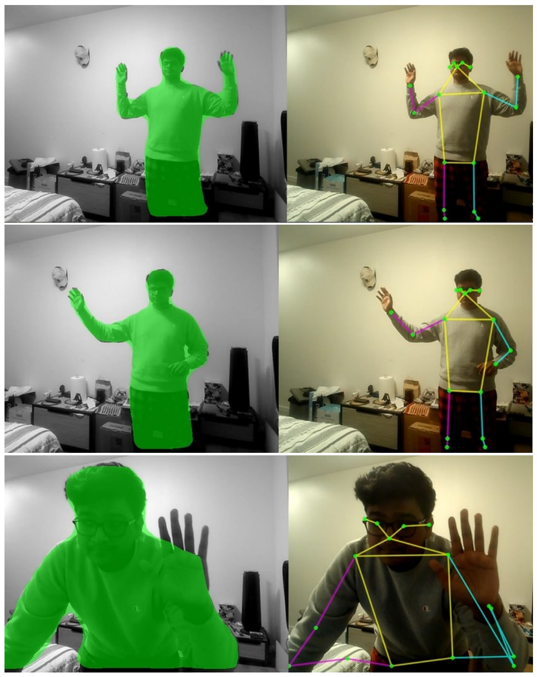


Figure 2: Mask RCNN Pose Estimation

5.1.2 Mask RCNN Testing

Device used: Works on cpu and gpu
Key Points Tracked: Returns 17 key points

- Extremely accurate , even when key points overlap and during lowlight conditions.
- Works well with multiple people in the frame
- Also works as a segmentation algorithm
- Detects legs properly and also detects limbs when close to the camera
- Sometimes detects phantom limbs or joints that do not exist
- It is extremely slow on CPU and fails to work in real-time
- Needs GPU to work fast for real-time detections

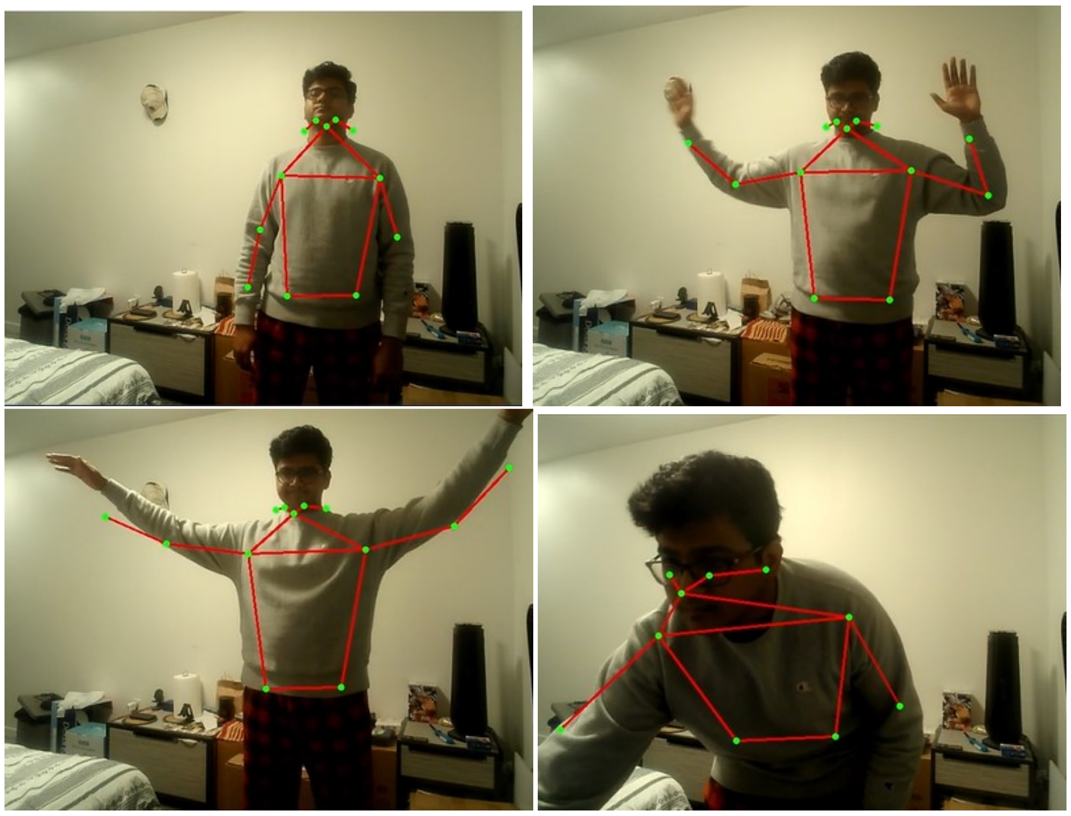


Figure 3: Movenet Pose Estimation

5.1.3 MoveNet Testing

Device used: Works on cpu and gpu
Key Points Tracked: Returns 17 key points

- Mostly accurate, even during low light conditions.
- Works in real time with CPU alone.
- It's good at identifying limbs when close to the camera.
- Fails at properly tracking the eyes, nose, ears if the subject is wearing glasses and in low light conditions.
- Fails to detect legs while wearing checked pants or due to bad lighting.

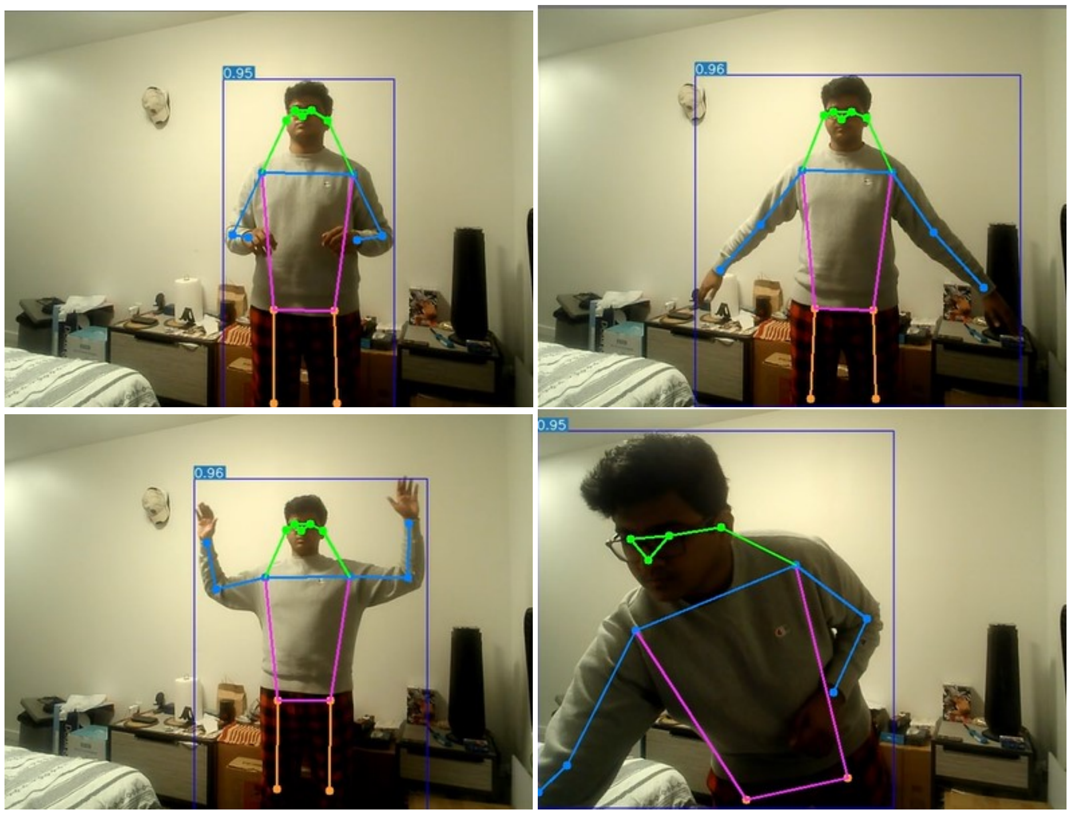


Figure 4: YOLOv7 Pose Estimation

5.1.4 Yolov7 Testing

Device used: Works on cpu and gpu
Key Points Tracked: Returns 17 key points

- Extremely accurate, even during low light conditions.
- Works moderately well on CPU (<=12 FPS).
- Works really well on GPU.
- It's good at identifying limbs when close to the camera.
- You need a GPU for it to work smoothly in real time.

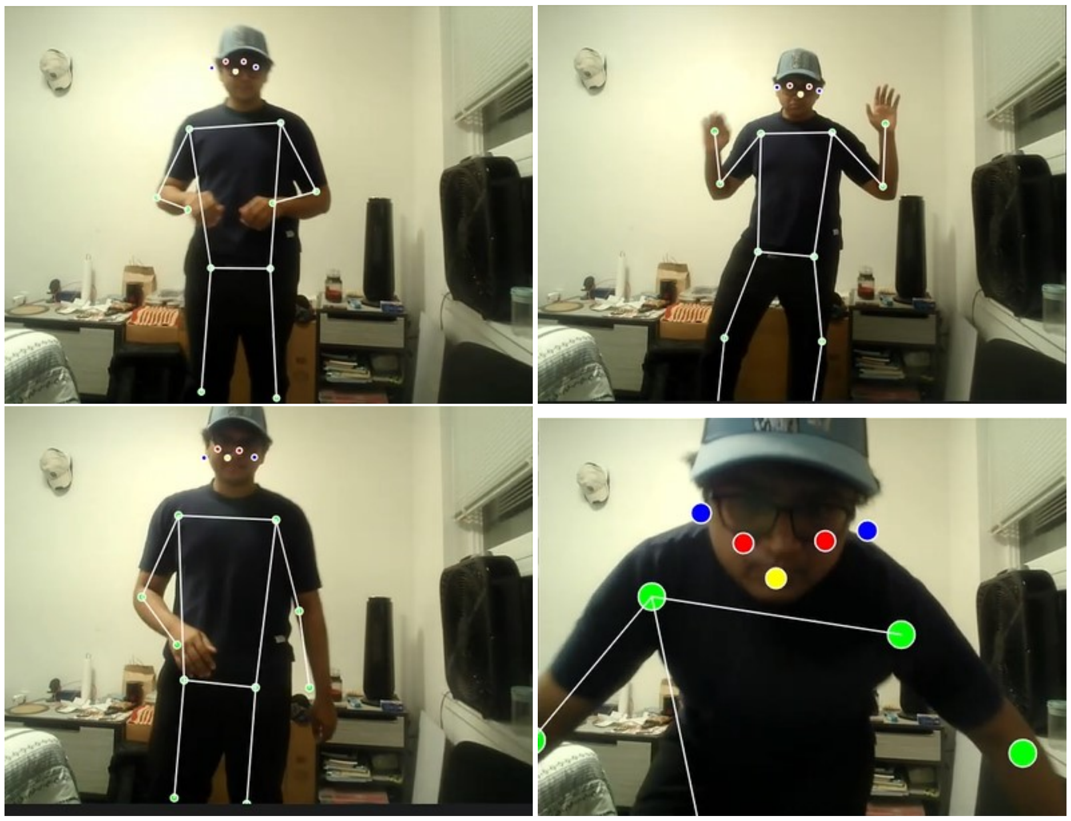


Figure 5: PoseNet Testing

5.1.5 Pose Net Testing

Device used: Works on cpu, does not use gpu
Key Points Tracked: Returns 17 key points

- Lightweight pose estimator, works with very little computing resources.
- Extremely accurate, even during low light conditions.
- Works moderately very well on CPU.
- Does not work on/use GPU.
- It's good at identifying limbs when close to the camera.
- It is not the optimal choice if you have a decent PC but the best when using it on a low-end computer.

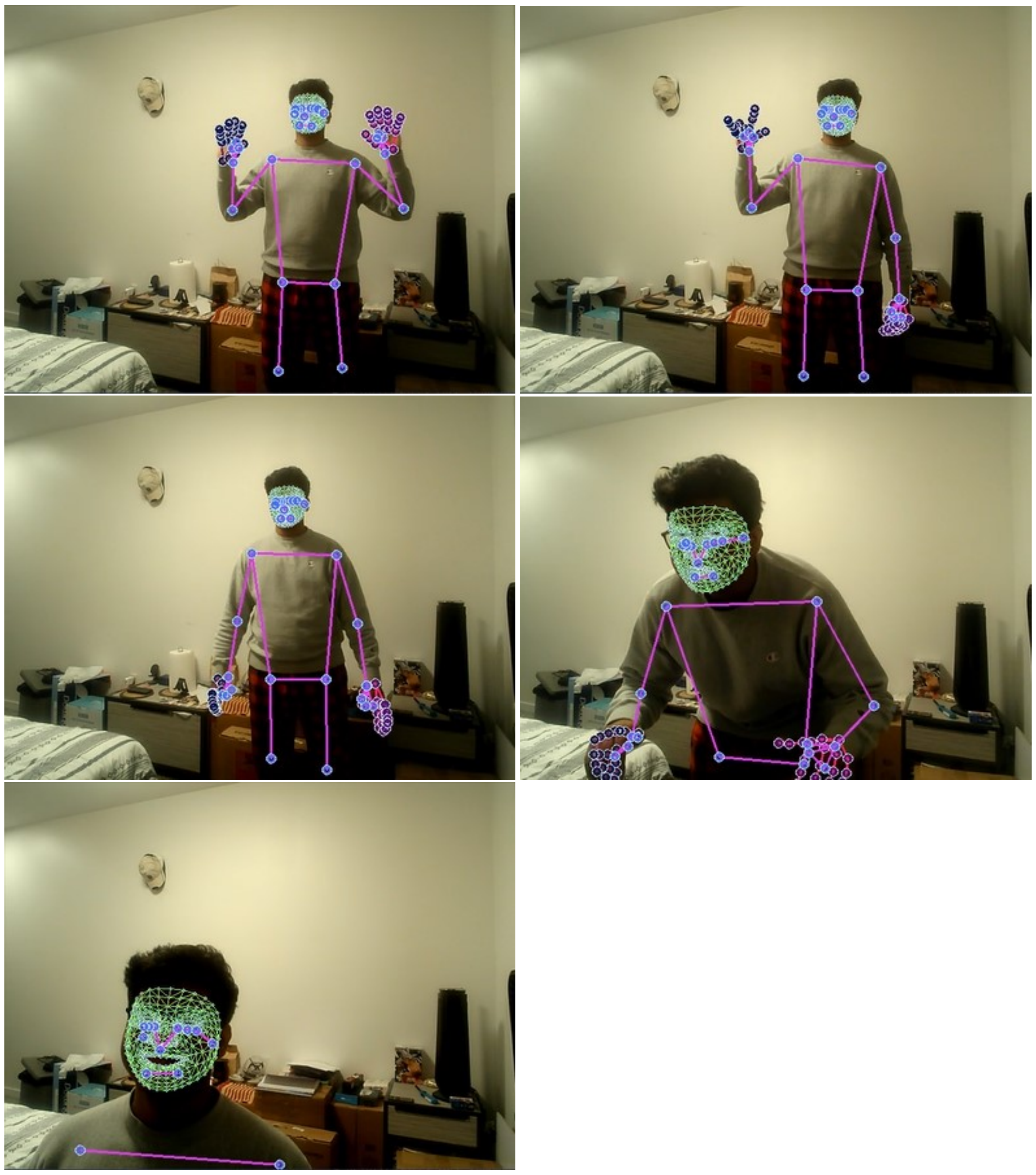


Figure 6: Mediapipe Pose Estimation

5.1.6 Media Pipe Testing

Device used: Works on cpu only
Key Points Tracked: Returns 135 key points

- Works extremely well with CPU resources alone.
- Provides tracking for facial expression and fingers as well.
- Does not require GPU.
- The most accurate pose estimator so far.
- Works well close up and in low light conditions.

5.2 RESULTS

Pose Estimator	Devices used	No. Key points	Works Real time	Subjects Tracked
Open Pose	CPU/GPU	17-135	Yes/CPU	1 person
MaskRCNN	CPU/GPU	17	No	N people
Move Net	CPU/GPU	17	Yes/CPU	1 person
Yolov7	CPU/GPU	17	Yes/GPU	N people
Pose Net	CPU	17	Yes/CPU	1 person
Media Pipe	CPU	135	Yes/CPU	1 person

Table 1: Comparison of different Pose Estimators

5.2.1 Chosen Method For Pose Estimation

6 DISCUSSION

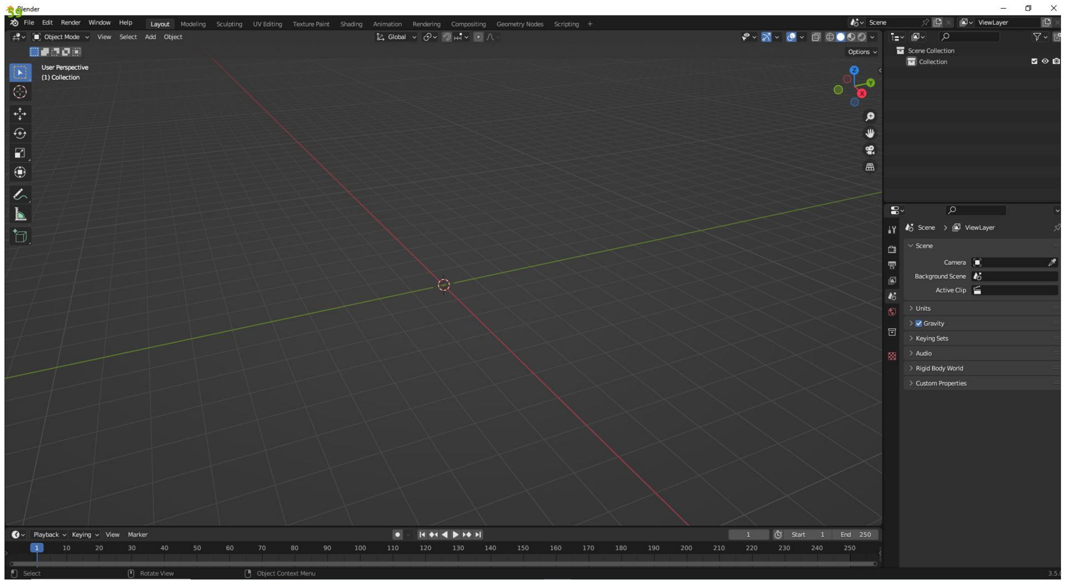


Figure 7: BLENDER

6.1 BLENDER

Blender is a powerful cross-platform, open-source software which can handle the whole creation pipeline - from character design and rendering to movie making and game development. Moreover, its functionality includes different kinds of simulators, various animation tools, instruments for use of mocap data, Python scripting and many others. For working with Python code Blender has a Python application programming interface (API) that provide the opportunity of working with native Blender libraries and tools.

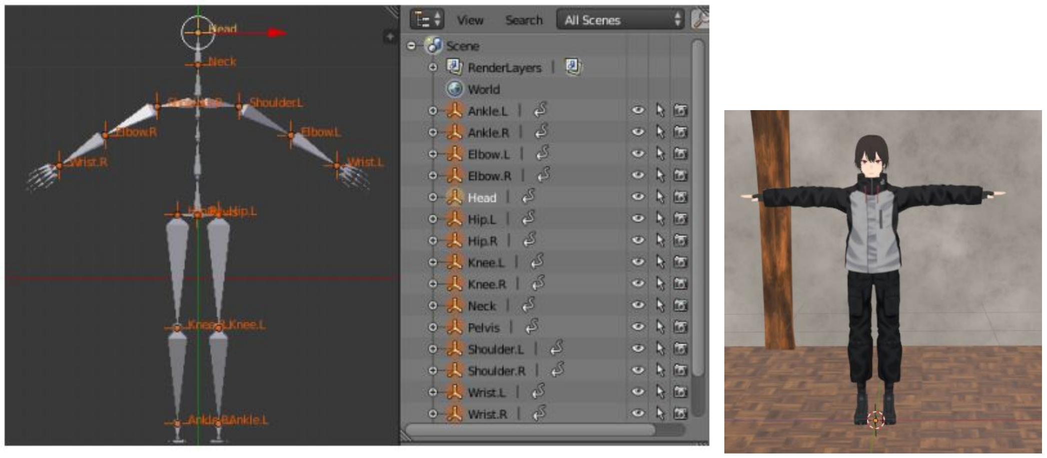


Figure 8: ARMATURE

6.1.1 Mesh and Armature

Mesh is merely a shell, and animating a collection of vertices for a video game or film can be challenging because the character artist must move every single vertex, even if some models have tens of thousands or even hundreds of thousands. Use of armatures is a quick and efficient fix for this problem. Most of the time, the skeleton or armature is made up of a hierarchical tree-like arrangement of bones. Every bone in this system is linked to the one before it, which means that if the parents bones move, the child ones will follow. These character skeletons typically resemble a standard skeleton in appearance.

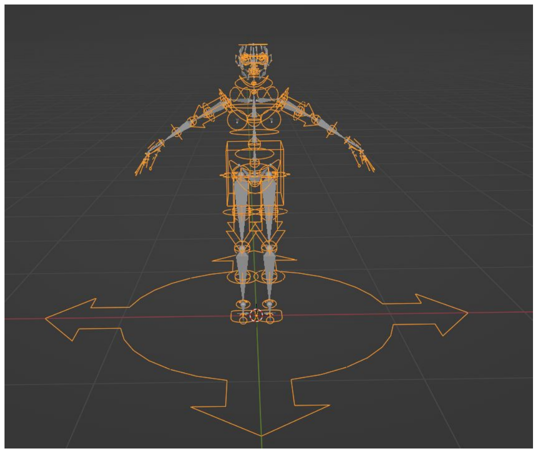


Figure 9: RIG

6.1.2 Rigging and Skinning

The act of constructing a character's skeletal structure is referred to as rigging, which involves not only the bone structure but also constraints, object modifiers, and other components. The end result is an armature, which is used to manipulate the mesh. To achieve this, the skin must be linked to the character's skeleton through a process known as skinning. In this process, groups of vertices are associated with their respective bones to enable movement. As a result, when a bone moves, the vertices assigned to it will follow suit and create the animation. Typically, vertices are assigned to the nearest bone and have a vertex weight that determines the degree of influence that each rig element has on each vertex. The larger the weight value, the more the vertex's translation will depend on the motion of the specific bone. At joint locations, vertices may be affected by multiple bones.

6.2 MediaPipe

A cross-platform, open-source framework for creating pipelines for computer vision and machine learning is called MediaPipe. It was created by Google and has a wide range of uses, including robotics, gesture recognition, augmented reality, and video analysis.

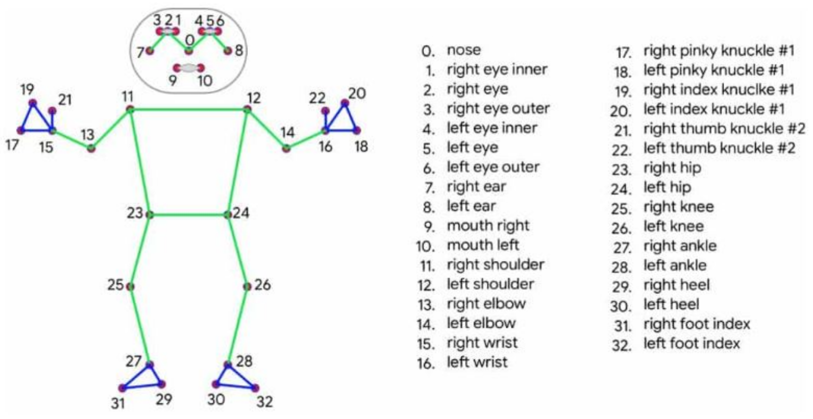
A modular and adaptable architecture is offered by MediaPipe for building ML and CV pipelines. It comprises a variety of pre-built parts that may be combined to build unique pipelines, such as feature extraction, neural network inference, and data augmentation. It is simple to optimize pipelines for particular devices and use cases with the aid of MediaPipe's real-time performance analysis and debugging tools.

Model	FPS	AR	PCK@0.2
OpenPose (body only)	0.41	87.8	83.4
BlazePose Full	102	84.1	84.5
BlazePose Lite	312	79.6	77.6

Table 2: Comparison of pose estimation models on the Yoga dataset

Google created BlazePose (Full Body), a posture detection model that can calculate the (x,y,z) coordinates of 33 skeleton keypoints. A powerful tracking algorithm is also a part of MediaPipe BlazePose, allowing it to identify body parts across frames even when they are obscured or just partially visible. In practical situations, this results in smoother and more precise stance predictions.

A Detector and an Estimator are two machine learning models that make up BlazePose. The Estimator inputs a 256x256 resolution image of the discovered person and outputs the keypoints while the Detector removes the human region from the input image.



0. nose	17. right pinky knuckle #1
1. right eye inner	18. left pinky knuckle #1
2. right eye	19. right index knuclke #1
3. right eye outer	20. left index knuckle #1
4. left eye inner	21. right thumb knuckle #2
5. left eye	22. left thumb knuckle #2
6. left eye outer	23. right hip
7. right ear	24. left hip
8. left ear	25. right knee
9. mouth right	26. left knee
10. mouth left	27. right ankle
11. right shoulder	28. left ankle
12. left shoulder	29. right heel
13. right elbow	30. left heel
14. left elbow	31. right foot index
15. right wrist	32. left foot index
16. left wrist	

Figure 10: Keypoints and there names Using BlazePose

The 33 keypoints are output by BlazePose in the sequence listed below. This is greater than the COCO dataset's 17 keypoints, which are frequently utilized, but less than 135 by openpose, making it the sweet spot. The Single-Shot Detector (SSD) is the basis for the detector's architecture. It produces a bounding box (1,2254,12) and a confidence score (1,2254,1) from an input image (1,224,224,3). The bounding box's 12 constituent parts of the form (x, y, w, h, kp1x, kp1y,..., kp4x, kp4y), where kp1x to kp4y are additional keypoints. There are 2254 elements total, each of which requires its own anchor, anchor scale, and offset to be used.

The Detector can be used in two different ways. The bounding box in box mode can be identified by its size (w,h) and position (x,y). In alignment mode, the bounding box with rotation can be anticipated by using (kp1x,kp1y) and (kp2x,kp2y) to compute the scale and angle.

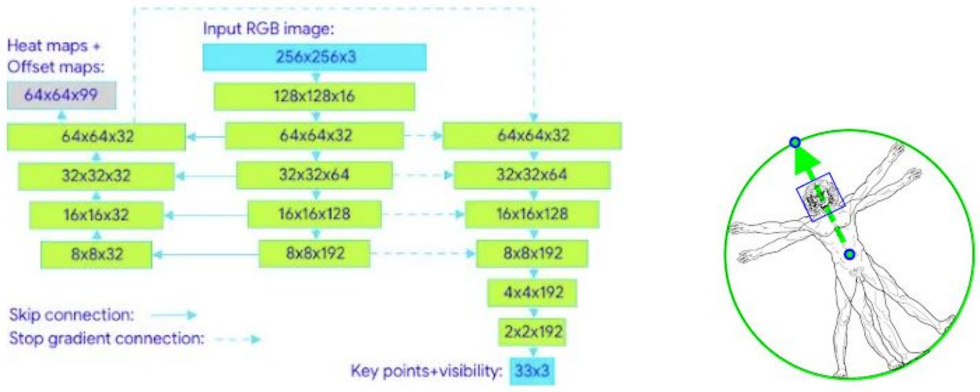


Figure 11: Tracking Network Architecture: Regression with Heatmap Supervision

The Estimator produces (1,195) landmarks as its first output and (1,1) flags as its second result. The 165 components that make up a landmark are (x, y, z, visibility, presence) for every 33 keypoints.

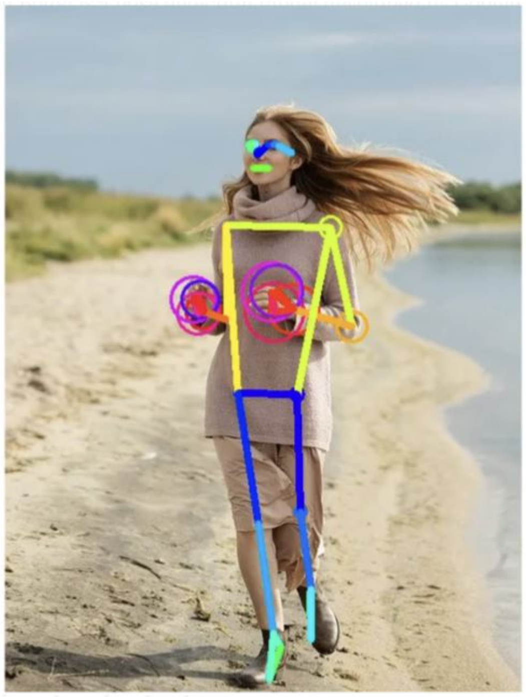


Figure 12: BlazePose With Circles denoting z value relative strength

The keypoints are between the subject's hips and the camera when the z-value is negative and behind the hips when the z-value is positive. The z-values are based on the subject's hips.

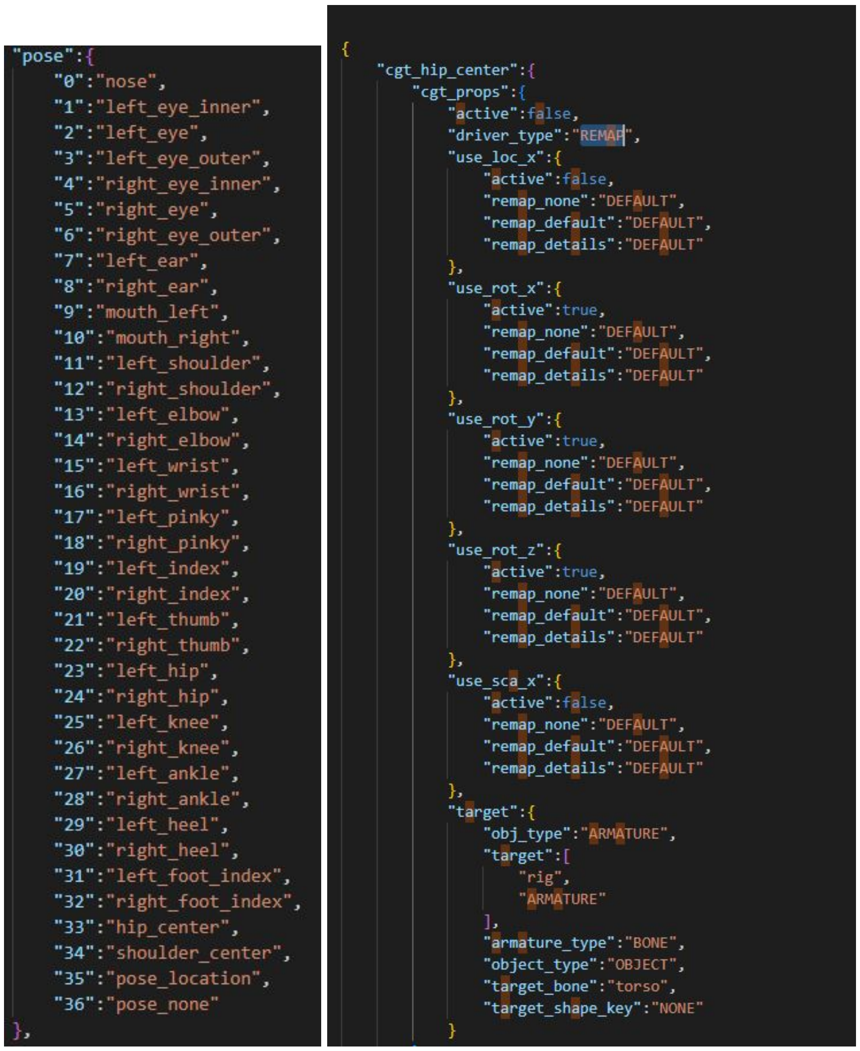
A sigmoid function is used to transform the visibility and presence, which are kept in the $[min\_float, max\_float]$ range, to probability. The visibility function returns the likelihood that any keypoints are present in the frame and not blocked by other objects. The probability of keypoints being present in the frame is returned by presence.

Additionally, the upper body can be estimated using the BlazePose (Upper Body). MediaPipe initially only made available the upper body model; later, they added the full body model. The complete body and upper body models have distinct specs; for instance, the upper body model's detector resolution is 128x128.

6.2.1 Depth and Bone Reamapping Using Plugin

Similar to X and Y coordinates, the Z coordinate measures distances in "image pixels" and is based on the subject's hips, which serve as the origin of the Z axis. Positive values are behind the hips, while negative values are between the hips and the camera. The Z coordinate scale is comparable to the X and Y scales, but it has a different nature because it was created by fitting synthetic data to the 2D annotation rather than by human annotation. Keep in mind that Z is not metric but accurate.

We have mediapipe bone names and we have to remap them to a rig bone name as can be seen in fig 14 which have different format and then to drivers which are 3D points that change over time when actions and poses are being recorded or streamed . By giving a joint's (x, y) coordinates on the 2D



```
"pose":{
    "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",
    "33":"hip_center",
    "34":"shoulder_center",
    "35":"pose_location",
    "36":"pose_none"
},
```

```
{
    "cgt_hip_center":{
        "cgt_props":{
            "active":false,
            "driver_type":"REMAP",
            "use_loc_x":{
                "active":false,
                "remap_none":"DEFAULT",
                "remap_default":"DEFAULT",
                "remap_details":"DEFAULT"
            },
            "use_rot_x":{
                "active":true,
                "remap_none":"DEFAULT",
                "remap_default":"DEFAULT",
                "remap_details":"DEFAULT"
            },
            "use_rot_y":{
                "active":true,
                "remap_none":"DEFAULT",
                "remap_default":"DEFAULT",
                "remap_details":"DEFAULT"
            },
            "use_rot_z":{
                "active":true,
                "remap_none":"DEFAULT",
                "remap_default":"DEFAULT",
                "remap_details":"DEFAULT"
            },
            "use_sca_x":{
                "active":false,
                "remap_none":"DEFAULT",
                "remap_default":"DEFAULT",
                "remap_details":"DEFAULT"
            },
            "target":{
                "obj_type":"ARMATURE",
                "target":[
                    "rig",
                    "ARMATURE"
                ],
                "armature_type":"BONE",
                "object_type":"OBJECT",
                "target_bone":"torso",
                "target_shape_key":"NONE"
            }
```

Figure 13: Medipaipe and Blender Human Rig Remapping

plane of the camera picture and its estimated depth on the z-axis, MediaPipe BlazePose can calculate its location in 3D space. In situations where depth information is crucial, like in augmented reality or motion capture applications, this enables more precise and realistic representations of human position.

6.3 Demonstartion

We demonstrate the replication of pose data on mesh and rig using a plugin that converts the bone name from mediapipe to human rig in blender .It can is stored in video that can be viewed as the output.

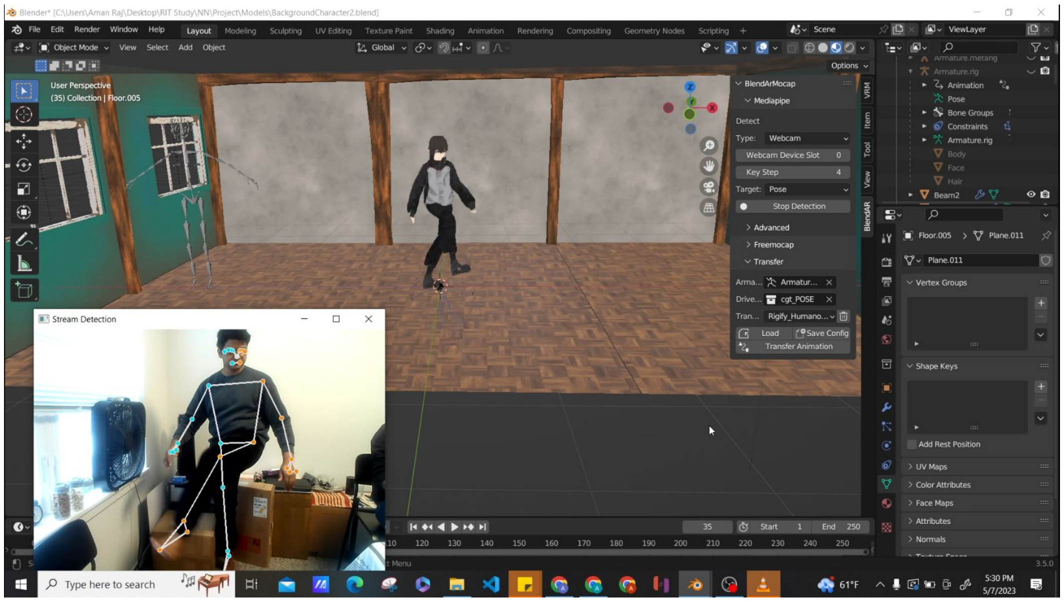


Figure 14: Blender Mediapipe Demonstration

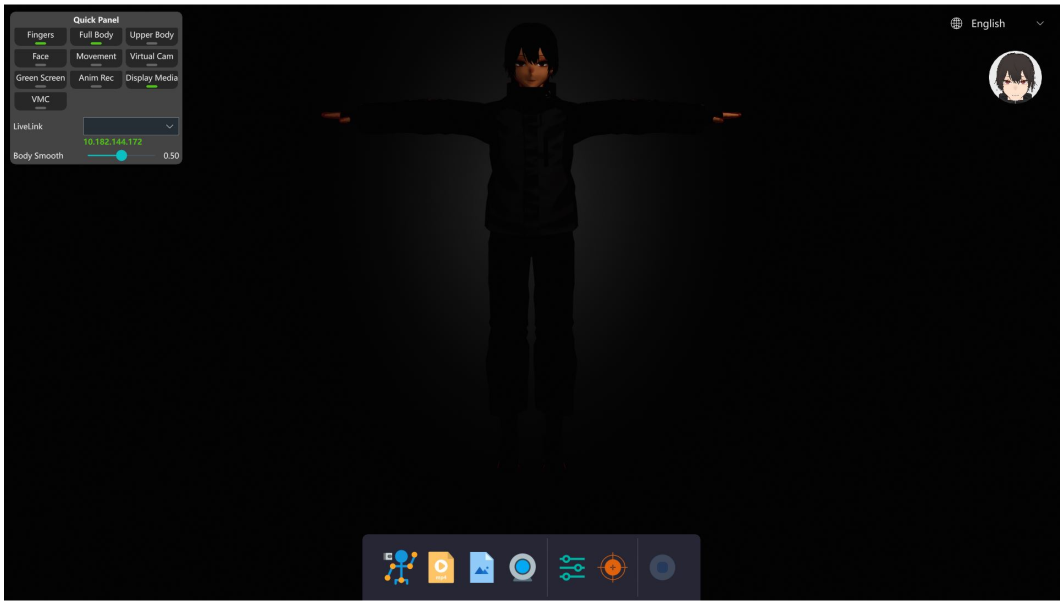
6.4 PROBLEMS AND LESSON LEARNED

We had problems linking the mediapipe to Blender,unreal engine and unity as well as making standardised rig like Unreal Engine .We underestimated the unreal engine and 3d software difficulty level for transferring data and uniform rig structure for models in different softwares.As the number of bones in a model can differ so we either remap mediapipe output to bones but that can leaves us with so many other calculation required for missing bones.Luckily we found a plugin for mediapipe that was open-source to convert mediapipe to 3D model containing human meta rig (default rig in blender).I still took us long time to learn blender and how to rig a character with different skeleton and rig to transform rig to blender human rig which could accept the 3D points recorded by the plugin and stored in drivers(which contains the translation ,rotation and smoothing data).

7 CONCLUSION AND FUTURE WORK

In this project, we have demonstrated the potential of real-time pose estimation as a cost-effective alternative to traditional motion capture techniques. By comparing and evaluating the performance and accuracy of various state-of-the-art pose estimation libraries, we have identified MediaPipe with BlazePose Lite as the most efficient and accurate library for real-time motion capture applications. We have also shown how a simple webcam can be used to capture human movements and track key points to animate a 3D avatar in real time using software such as Blender. Our approach provides an affordable and accessible motion capture solution that can benefit developers and creatives looking to incorporate motion capture into their projects without breaking the bank.

Future work for this project could include exploring other pose estimation libraries, such as MoveNet or Yolov7 Pose Estimator, and comparing their performance and accuracy to the ones used in this project. Additionally, the use of two cameras for depth estimation could be investigated to enhance the accuracy of the pose estimation results.Another potential area for future work is to explore the use of Unreal Engine for real-time motion capture applications. Unreal Engine provides advanced graphics capabilities and a user-friendly interface for creating immersive experiences. By integrating pose estimation into Unreal Engine, developers and creatives can create high-quality animations and simulations without the need for expensive motion capture equipment.



LiveLink 10.182.144.172
Body Smooth 0.50

Figure 15: Unreal Engine Example

We know it can be done as we have some examples for unreal engine and unity.

References

[1] Bazarevsky, V., Grishchenko, I., Raveendran, K., Zhu, T., Zhang, F. and Grundmann, M., 2020. Blazepose: On-device real-time body pose tracking. arXiv preprint arXiv:2006.10204.
[2] Sánchez Riera, J. and Moreno-Noguer, F., 2020. Integrating human body mocaps into Blender using RGB images. In 2020 13th International Conference on Advances in Computer-Human Interactions (ACHI) (pp. 285-290). International Academy, Research, and Industry Association (IARIA).
[3] Borodulina, A., 2019. Application of 3D Human Pose Estimation of Motion Capture and Character Animation (Doctoral dissertation, Master's thesis, University of Oulu, 6 2019).