Real-Time 3D Avatar Modeling for AR using Human Pose and Actions in Resource-Constrained Web Environments

S.R. Galappaththy

Real-Time 3D Avatar Modeling for AR using Human Pose and Actions in Resource-Constrained Web Environments

Sandul Renuja Galappaththy

Index No: 20000545

Supervisor: Dr. Damitha Sandaruwan

Submitted in partial fulfillment of the requirements of the B.Sc in Computer Science Final Year Project (SCS4224)

Declaration

I certify that this dissertation does not incorporate, without acknowledgment, any material previously submitted for a degree or diploma in any university and to the best of my knowledge and belief, it does not contain any material previously published or written by another person or myself except where due reference is made in the text. I also hereby give consent for my dissertation, if accepted, be made available for photocopying and for inter-library loans, and for the title and abstract to be made available to outside organisations.

Candidate Name: Sandul Renuja Galappaththy

Date: 30-06-2025

Signature of Candidate

This is to certify that this dissertation is based on the work of
Mr. S. R. Galappaththy

under my supervision. The thesis has been prepared according to the format stipulated and is of acceptable standard.

Principle/Co- Supervisor's Name: Dr. Damitha Sandaruwan

Date: _____

Signature of Supervisor

Abstract

Real-time 3D avatar animation using standard webcams offers immense potential for immersive communication and interaction within web-based Augmented Reality (AR). However, existing solutions often struggle with accessibility and generalisation, either requiring specialised hardware or relying on a single, high-fidelity pose estimation input that may be too resource-intensive for many web environments. This creates a significant gap, as these systems cannot easily adapt to the diverse quality and format of data from various readily available pose estimators or gracefully handle common real-world challenges like partial user visibility.

This research addresses these limitations by employing a Design Science Research methodology to design, implement, and evaluate a novel, generalized middleware pipeline for real-time, webcam-based 3D avatar animation, operating entirely within standard web browsers. The core contribution is a modular JavaScript-based architecture centered around a biomechanically-aware canonical pose representation aligned with the VRM humanoid standard. The methodology involves developing an adaptive input adapter for heterogeneous data (from MoveNet, BlazePose, YOLO-Pose, etc.), a pose processor for heuristic 2D-to-3D lifting and robust inference of occluded joints using data-driven priors from H36M, and a flexible retargeting module.

Experimental results demonstrate the pipeline's ability to successfully process diverse inputs and drive plausible, full-body avatar animations in real-time. Notably, the system generates coherent motion even from sparse 2D keypoint data where simpler direct mapping would fail. Performance analysis indicates viability on desktop/laptop browsers and feasibility on mobile devices with lighter-weight estimators. This research presents a significant step towards more accessible and flexible real-time avatar systems for the web platform, providing a practical and extensible foundation for future advancements in web-based embodied interaction.

Contents

List of Figures

List of Tables

Acronyms

API Application Programming Interface. 27

AR Augmented Reality. ii, 1–5, 7, 8, 14–16, 19, 28–30, 32, 49, 52, 53, 55

BFS Breadth-First Search. 39, 40, 44

CG Computer Graphics. 1

CNN Convolutional Neural Network. 8, 14

CPU Central Processing Unit. 24

CUDA Compute Unified Device Architecture. 27

CV Computer Vision. 1

EMA Exponential Moving Average. 45

FK Forward Kinematics. 56

FoV Field of View. 16

FPS Frames Per Second. 50

GAN Generative Adversarial Network. 10, 18

GPU Graphics Processing Unit. 24, 27

HMR Human Mesh Recovery. 10

IK Inverse Kinematics. 34, 48, 56

ML Machine Learning. 1, 48

MR Mixed Reality. 16

NMS Non-Maximum Suppression. 34

PAF Path Affinity Fields. 34

RGB Red, Green, Blue. 1, 8, 10, 14, 15

RGBD Red, Green, Blue, Depth. 13

SDK Software Development Kit. 12

SMPL Skinned Multi-Person Linear Model. 8–12, 14, 15, 18, 30

VR Virtual Reality. 1, 3, 7, 12, 14, 27–30, 55

Wasm WebAssembly. 25, 33, 50, 56, 58

XR eXtended Reality. 5

1 Introduction

In today's world, the various ways we see, hear, and interact with the digital worlds are constantly changing and evolving. The advancements and achievements in the fields of Computer Graphics (CG), Computer Vision (CV), and Virtual Reality (VR)/Augmented Reality (AR) play a major role in creating these digital environments that we can feel and interact with just like our physical world (Al-Ansi et al. (2023)). These fields which span a huge domain of knowledge have interested people for a long time and have a significant potential for research as well.

By combining Computer Graphics (CG) and Computer Vision (CV) technologies, VR and AR make it possible to create experiences that feel very real. VR immerses users in entirely synthetic virtual environments, while AR overlays digital elements onto the real world, blurring the lines between the physical and digital worlds. With the rapid development of these technologies, there is a growing demand for realistic and immersive experiences that seamlessly blend digital content with the physical world. One key aspect of this is the ability to capture and model real human appearance, pose and actions into life-like 3D avatars that can be rendered and animated in VR or AR environments. Applications of these technologies range from virtual conferencing and remote collaboration to gaming, training simulations, and beyond (Mazuryk & Gervautz (n.d.)).

1.1 Research Background

Motion capture and human pose estimation has been active areas of research for decades, with extensive work on both marker-based and marker-less techniques using various camera setups (Caserman et al. (2020)). The advent of depth sensors like the Kinect in the 2010s enabled new marker-less approaches to full-body motion capture based on 3D data (Husein & Ciawi (2021), Huang et al. (n.d.)).

More recently, advances in CV and Machine Learning (ML) have made it possible to estimate 3D human pose from just monocular Red, Green, Blue (RGB) video in real-time. Google's MediaPipe is a leading technology in this area, using machine learning models trained on massive datasets to accurately predict 3D skeleton positions from video input (Lin et al. (2023)).

In parallel, progress has been made in photogrammetric scanning of real people and objects to generate extremely high quality 3D models and textures.Huang et al. (n.d.) developed workflows to combine this with motion capture to create animatable photo-real digital humans.

1.2 Research Motivation

Despite these advancements, many widely studied approaches to avatar-based interaction rely on specialized hardware, such as depth cameras or multi-camera setups, which limits their accessibility and scalability (Alexiadis et al. (2017), Charles (2016)). Furthermore, many software solutions focus on a single part of the problem—such as high-fidelity pose estimation or photorealistic character

modeling—without providing an integrated, end-to-end pipeline that is both accessible and efficient.

This research is motivated by the potential to bridge this gap, empowering a broader audience with an immersive way to interact within AR environments. The core motivation is to enable these experiences on resource-constrained platforms like standard smartphones and laptops, using only a common webcam, without requiring any special hardware or software installations. By choosing the web browser as the target environment, the solution becomes universally accessible. This could unlock novel applications in remote collaboration, virtual social platforms, and interactive entertainment, allowing anyone to represent themselves with a dynamic 3D avatar in real-time.

1.3 Research Aims and Objectives

1.3.1 Aim

Developing an efficient, widely-accessible, end to end solution which can run on smartphone browsers with no special equipment, to replicate real human expressions, pose and movements in a 3D avatar that can be rendered and animated in real-time within a resource-constrained environment.

1.3.2 Objectives

- To identify and evaluate optimal real-time human pose estimation techniques that balance accuracy and performance for deployment within a web browser.
- To develop a novel and generalized algorithmic pipeline that can process heterogeneous pose data (e.g., sparse 2D, dense 3D) and transform it into a standardized canonical representation, inferring missing data using biomechanical priors.
- To implement a retargeting system that maps the canonical pose to animate standard 3D avatar formats (e.g., VRM) without significant latency or visual artifacts.
- To evaluate the performance and qualitative realism of the complete pipeline when rendering the animated avatar within a web-based AR context.

1.4 Research Questions

1. What are the trade-offs between accuracy and performance for real-time human pose estimation algorithms when deployed in a resource-constrained web environment?
2. How can a generalized transformation pipeline be designed to robustly map diverse pose estimation outputs (sparse 2D to dense 3D) to a canonical rep-

resentation for animating a standard 3D avatar, particularly when handling occluded or missing joint data?

3. To what extent can a purely web-based, monocular system achieve plausible real-time avatar animation, and what are the key limitations in terms of performance and realism for browser-based AR experiences?

1.5 Summary of the Chapter

This chapter introduced the rapidly advancing field of real-time 3D avatar animation, driven by progress in computer vision, graphics, and immersive AR/VR technologies. The core motivation for this research is to bridge the accessibility gap, moving beyond systems that require specialized hardware or high-fidelity input. I aim to develop an efficient, web-based pipeline enabling users to animate standard 3D avatars (like VRM models) using only a common webcam, directly within resource-constrained browser environments on devices like smartphones or laptops. This could significantly enhance applications in remote collaboration, virtual social platforms, and entertainment by allowing more people to achieve embodied digital representation.

To guide this effort, the research objectives focus on evaluating suitable real-time pose capture techniques, designing a novel and generalized transformation pipeline that incorporates biomechanical plausibility and can handle diverse or incomplete input data, and assessing the feasibility of rendering the animated avatar in web-based environments and speciallt AR interactions. The key research questions explore the optimal capture methods for constrained environments (RQ1), the design of this generalized mapping pipeline (RQ2), and the resulting trade-offs in realism, performance, and user experience (RQ3). The subsequent chapters will explore the relevant literature and detail the specific methodology developed to address these questions.

2 Research Approach

1. **Method Selection and Adaptation:** Determine the most suitable motion capture technique (e.g., using MediaPipe for skeletal point extraction) and adapt it to the specific requirements of capturing human expressions, pose, and movements using a smartphone camera.

2. **Data Extraction:** Employ the chosen motion capture technique to extract human expressions, pose, and movements from real-time camera feeds, ensuring the captured data can be transferred and animated in real-time.

3. **Avatar Selection and Modeling:** Identify a real-time animatable 3D avatar that can be controlled via game engine scripts to display movements and poses. Optionally, this step may involve creating a custom avatar model to enhance realism.

4. **Algorithm Development:** Design and implement an efficient algorithm for mapping input points (captured skeletal points) to the output (3D avatar), accommodating various scenarios and conditions.

5. **Adaptation for Mobile AR:** Optimize the rendering process to ensure the animatable model can be effectively displayed within a browser-based mobile AR environment.

6. **Evaluation:** Assess the performance, realism, and user experience of the entire solution, identifying areas for improvement and refining the system accordingly.

2.1 Scope

2.1.1 In Scope

The following areas will be covered under the scope of this research.

Capturing human pose and motion: This involves experimenting with state of the art methods proposed in existing literature and selecting the optimal methods for the purpose, with a special focus on resource usage, then adapting the methodology to efficiently capture human expressions, pose and motion using a webcam or a smartphone camera. This data collection and processing should be doable in an efficient manner with minimal latency and performance overhead, and it should support operating in browser based (resource constrained) environments.

Mapping to a digital animatable 3D model, on-device within the browser: Developing algorithms and techniques for mapping captured human behavior to virtual representations in real-time with minimal latency and high fidelity. As transformation and mapping algorithm will be developed to map the input points to the output. Also this process, has to be optimized to work efficiently under various conditions such as missing input points, keypoint/coordinate system mismatches, occlusions, etc. The realism or the dynamic nature of the

avatar is not strictly considered for the study, however, the viability of making the avatar be dynamic will be considered.

Real-time Rendering on web browser XR environment: Ensuring the efficient rendering of the animatable model within a resource constrained environment, such as a smartphone-based AR environment running on a web browser.

2.1.2 Out of Scope

The following will not be covered under the scope of this research.

Creating ultra realistic human models: This research will focus on accurately representing the pose and actions, not on generating visually realistic avatars. Creating synthetic human behaviours: This research will concentrate on replicating real human actions, not on simulating artificial behaviours. Working with multiple people: This research will focus on accurately representing a single character and not capturing motion and replicating for multiple people, or any interactions between people.

2.2 Thesis Structure

This thesis is organized into seven chapters, each designed to logically guide the reader from the initial problem statement through to the final conclusions. The structure has been chosen to clearly separate the research methodology from the technical implementation, providing a clear and rigorous account of the work undertaken.

- **Chapter 1: Introduction** This chapter introduces the research field, presents the background and motivation for the study, and clearly defines the research aims, objectives, and core research questions that guide the entire project.

- **Chapter 2: Literature Review**
  This chapter provides a critical review of relevant academic literature and existing technologies. It analyzes the state of the art in human pose estimation, 3D avatar modeling, and real-time web graphics to identify the specific research gap that this thesis aims to address.

- **Chapter 3: Research Methodology**
  This chapter outlines the high-level research approach. It introduces the Design Science Research (DSR) paradigm adopted for this project and describes the structured phases of the research plan, from initial technology evaluation to the final system evaluation.

- **Chapter 4: System Design and Development**
  This chapter presents the technical core of the thesis: the design and implementation of the novel software pipeline. It details the system architecture, the algorithms used for data processing and inference, the data-driven priors, and the technologies used to build the functional artifact.

- **Chapter 5: Results and Analysis**
  This chapter presents the empirical results obtained from testing the developed system. It provides performance benchmarks, such as Frames Per Second (FPS), and a qualitative analysis of the animation quality, directly addressing the research questions with concrete data and observations.

- **Chapter 6: Discussion**
  This chapter interprets the significance of the results. It discusses the key contributions of the research, the implications of the findings, and provides a frank acknowledgment of the system's limitations.

- **Chapter 7: Conclusion and Future Work**
  The final chapter summarizes the key outcomes and contributions of the thesis. It reflects on how the research questions were answered and proposes specific, actionable directions for future research that can build upon the foundation laid by this work.

2.3 Summary of the Chapter

This chapter has set the stage for the research presented in this thesis. It began by introducing the exciting and rapidly advancing field of real-time 3D avatar animation and established the core motivation for this project: to improve the accessibility and flexibility of these technologies by creating a solution that works on standard hardware within a web browser. The specific aims, objectives, and research questions that will guide this investigation have been clearly defined. Finally, the overall structure of the thesis has been outlined, providing a roadmap for the subsequent chapters which will detail the literature review, methodology, system design, results, and conclusions of this work.

3 Literature Review

To understand the current state of the art and identify potential research gaps within the problem domain of real-time 3D avatar modeling in web environments, a thorough literature review was conducted. This review focuses on key areas including human pose and motion capture, 3D human modeling and representation, and the specific challenges related to real-time performance and user experience in web-based AR and VR environments.

Most of the existing research related to this problem domain can be categorised into a few categories according to their main goal and scope of study as:

1. Estimating pose, capturing motion
2. Capturing and modelling
3. Capturing and animating
4. Modelling and animating
5. Capturing, modelling and animating

This research focuses on achieving this in real-time within the resource-constrained environment of web browsers, specifically targeting mobile devices. This requires a deep understanding of efficient human pose estimation, effective 3D human modeling, and performant real-time animation techniques suitable for web deployment. Therefore, the following review examines the evolution and current state-of-the-art across these interconnected domains, drawing heavily on recent advancements in deep learning and computer vision, to contextualize the research questions and identify prominent methodologies and challenges.

3.1 Estimating pose, capturing motion

Accurately capturing human pose and motion in real-time is the foundational step for animating 3D avatars. Therefore, Human Pose and Motion Capture are topics that have been the focus of many studies over the years. The field has evolved significantly, driven by the need for more accessible and less intrusive methods compared to traditional techniques.

Human pose estimation has traditionally been achieved through marker-based systems, which require the attachment of physical markers or sensors to the subject's body. Marker-based systems are known for their high accuracy and have been widely used in the entertainment industry for creating realistic character animations in movies and video games. However, marker-based systems have several limitations. They require specialized equipment and controlled studio environments, making them expensive and impractical for many applications. To overcome these limitations, there has been a growing interest in marker-less motion capture and 3D pose estimation methods. Traditional marker-based motion capture, while offering high fidelity (Caserman et al. (2020)), requires specialized

labs, equipment, and often manual processing with tools like OpenSim, making it impractical for widespread AR applications (Huang et al. (n.d.)).

Early marker-less approaches relied on multiple calibrated cameras to reconstruct 3D pose from silhouettes or visual hulls. Microsoft Kinect, introduced in 2010, popularized marker-less motion capture by providing real-time depth sensing capabilities. Many works have used Kinect for 3D pose estimation in gaming, robotics, and healthcare applications (Desmarais et al. (2020), Ji et al. (2020*a*)).

These consumer depth sensors enabled real-time 3D skeletal tracking and demonstrated the potential for direct user movement mapping in virtual environments as demonstrated in studies by Charles (2016). However, depth sensors possess limitations regarding range, ambient light interference, and device ubiquity.

Monocular Deep Learning Approaches:

In the past few years, focus has largely shifted to Monocular Human Pose Estimation (MHPE) using standard Red, Green, Blue (RGB) cameras, fueled by deep learning breakthroughs since 2014. Comprehensive surveys by Liu, Bao, Sun & Mei (2021), Chen et al. (2020)and Ji et al. (2020*b*) categorize the rapid advancements. Key paradigms include: **2D vs. 3D Estimation:** Many methods first estimate 2D keypoints (using techniques like OpenPose, or heatmap regression and then "lift" these to 3D space, leveraging robust 2D datasets. Others attempt **direct 3D regression** from image features.

Many of these approaches that estimate 3D pose from a single RGB image rely on a parametric 3D human body model, such as Skinned Multi-Person Linear Model (SMPL) (Loper et al. (n.d.)), to constrain the space of possible poses. Bogo et al. (2016) proposed one of the first methods to estimate 3D pose and shape from a single image by fitting the SMPL model to 2D joint detections. Remarkably, fitting only 2D joints produces plausible 3D body shapes. However, monocular 3D pose estimation still faces challenges in terms of accuracy, robustness to occlusion, and generalisation to unseen poses and subjects.

Significant advancements in real-time monocular 3D pose estimation have been made using Convolutional Neural Network (CNN). Mehta et al. (2017) introduced VNect, a system predicting 2D heatmaps and 3D joint positions in real-time from a single image. Silva et al. (2019) proposed TensorPose, a streamlined CNN architecture for real-time multi-person pose estimation, predicting both 2D heatmaps and 3D positions. Other widely used methods and frameworks in this space include OpenPose (Cao et al. (2016)), known for its robust 2D multi-person detection, and DensePose (Güler et al. (2018),) which maps all human pixels to a 3D surface model.

Achieving bio-mechanically accurate pose estimation from monocular video, beyond simplified parametric models, was also explored by methods like BioPose (Koleini et al. (2025)). This study, published in 2025, uses neural inverse kinematics and refinement techniques. It pushes further towards anatomical accuracy by integrating a biomechanical skeleton model (like OpenSim) with mesh recovery (MQ-HMR) and Neural Inverse Kinematics (NeurIK), but it has high complexity.

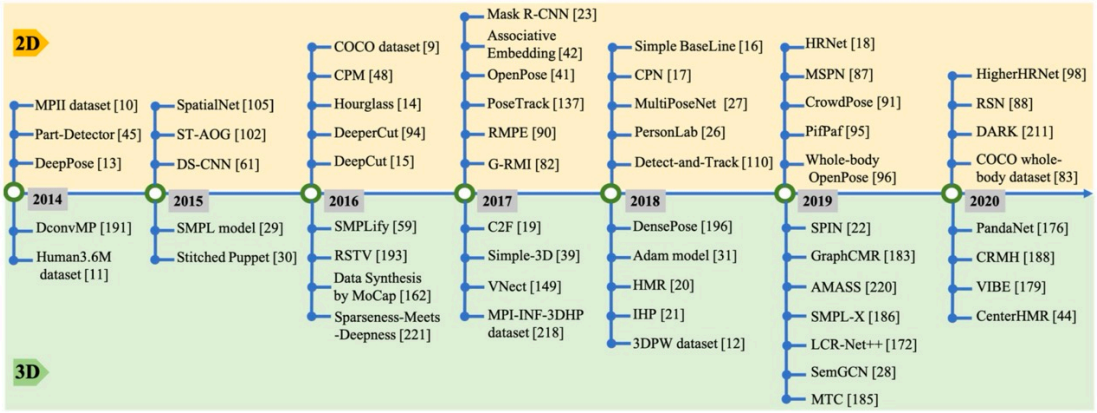


Figure 1: Milestones, idea or dataset breakthroughs, and the state-of-the-art methods for 2D (top) and 3D (bottom) pose estimation from the year 2014 to 2021. (From: Liu, Bao, Sun & Mei (2021))

When considering the studies from recent years, Google's MediaPipe is a notable achievement, specially for resource constrained environments. MediaPipe is an open-source framework that provides a set of tools for building multimodal applied machine learning pipelines, including pose estimation. One of the key advantages of MediaPipe is its ability to perform real-time pose estimation on resource-constrained devices such as smartphones and embedded systems. This is achieved through a combination of efficient model design, optimized inference, and hardware acceleration. (Kim et al. (2023)). However, despite its strengths, MediaPipe's pose estimation module still has some limitations. Lin et al. (2023) suggested ways improve the accuracy of MediaPipe's 3D human posture detection and their improved system achieves over 90% accuracy in multi-pose recognition tests.

3.2 Capturing and modelling

A few studies have focused extensively on the steps of capturing the appearance of the human subject and then creating a realistic 3D model.

Avatar Appearance and Realism:

While this project mainly focuses on pose and motion mapping, avatar appearance significantly impacts user experience (Caserman et al. (2020), Piumsomboon et al. (2018)). This scope of study also seems highly promising, as an efficient web-based solution, (possibly also making use of recent developments in GenAI and Vision Models) has lots of usecases in the modern day. Creating personalized 3D avatars that can be animated by captured motion is an active research area. Some studies focus on reconstructing detailed 3D models from images or video and then rigging them for animation, often by transferring weights from a generic model like SMPL. Li et al. (2019*a*) proposed a pipeline to reconstruct 3D human avatars

from a single RGB image by fitting a parametric model and warping it based on the silhouette, using a GAN called InferGAN to infer occluded textures. These methods demonstrate pipelines for generating personalized, animatable geometry based on a canonical representation derived from models like SMPL.

A significant advancement in single-view human digitisation came with the development of implicit function representations, particularly Pixel-Aligned Implicit Functions (PIFu) (Saito et al. (2019)) and its high-resolution successor, PIFuHD (Saito et al. (2020)). PIFuHD specifically addresses the challenge of reconstructing detailed geometry (like clothing folds, facial features, and fingers) from high-resolution (e.g., 1k) single images, which previous methods struggled with due to memory limitations forcing downsampling. While computationally intensive compared to skeletal tracking, PIFuHD demonstrates the potential of implicit functions for achieving state-of-the-art single-view reconstruction fidelity, representing a key benchmark in detailed avatar geometry generation from images.

Generative models are also being explored to create diverse and animatable human avatars. AvatarGen proposed by Zhang et al. (2022) generates animatable human avatars with varied appearances using only 2D images for training. It utilizes a coarse human body model as a proxy to warp the observation space into a standard avatar under a canonical space, learning pose-dependent deformations with a dedicated network. This approach highlights using a canonical space as an intermediate representation for generative modeling of animatable characters.

Generating realistic geometry and texture, especially for unseen parts from single-view input, is challenging. Methods like those reconstructing clothed humans from video (Alldieck et al. (2018)), generating textures via GANs (Li et al. (2019*a*)), using 3D scanning (Huang et al. (n.d.)), or employing generative models like AvatarGen (Zhang et al. (2022)) aim for high fidelity but often involve significant offline processing or computational cost, potentially conflicting with real-time requirements and web browser environment constraints.

3.3 Capturing and animating

Several other studies have focused on capturing the human motion and animating a generic model to represent the human, without worrying about modelling to create a realistic avatar. To animate a 3D avatar using captured pose and motion data, a suitable 3D model representation is required. A key challenge is mapping the captured pose and motion (often represented as joint rotations or 3D joint positions) to a generic avatar model and animating it in real-time.

Parametric Models: The Skinned Multi-Person Linear Model (SMPL) model (Loper et al. (n.d.)) is central to many Human Mesh Recovery (HMR) approaches (Bogo et al. (2016)). It provides a low-dimensional, controllable representation of human shape ($\boldsymbol{\beta}$ parameters) and pose ($\boldsymbol{\theta}$ parameters, joint rotations). Its differentiability allows end-to-end training, and its parameters serve as a form of generic, canonical pose representation. Fitting SMPL to 2D keypoints or image features is a common strategy (Bogo et al. (2016)). However, SMPL has limitations; it's based on a limited set of scans, primarily of minimally clothed individuals, and

might not capture the nuances of diverse body shapes, clothing, or highly dynamic poses accurately (Koleini et al. (2025), Bogo et al. (2016)).

Direct Mesh and Volumetric Representations: Alternatives include directly regressing mesh vertices (Lin et al. (2020*a*)), GraphCMR/Pose2Mesh referenced by Liu, Xu, Habermann, Zollhofer, Bernard, Kim, Wang & Theobalt (2021)) or using volumetric representations (Li et al. (2021)). METRO (Lin et al. (2020*a*)) demonstrates the power of transformers for direct mesh regression, learning complex vertex relationships without SMPL's structural constraints.

Detection-based vs. Regression-based methods: Detection-based methods often predict heat maps representing joint likelihoods Liu, Bao, Sun & Mei (2021), Chen et al. (2020), offering robustness, while regression-based methods directly output coordinates, which can be simpler but potentially less robust to ambiguity.

Joint-based vs. Mesh-based representations: Joint-based Representation: If 3D joint locations/rotations are the primary output (e.g., from MediaPipe, MMPose, or some direct methods), these need to be re-targeted to the target avatar's skeleton. This involves solving kinematic chains and handling differences in skeleton topology and proportions (Li et al. (2021)). This requires a robust re-targeting algorithm suitable for real-time execution.
Mesh-based Representation: If a full mesh is reconstructed, animating a different target avatar requires mesh deformation techniques or re-targeting based on an underlying skeleton extracted from the mesh.

Parametric body models, particularly SMPL, have become a de facto standard. SMPL provides a compact representation of human body shape and pose, controlled by low-dimensional parameters. It disentangles shape and pose variations and is designed with a skeletal structure and blend weights that are compatible with standard graphics pipelines (like linear blend skinning), making it highly suitable for animation. The model represents the body in a canonical (T-pose or A-pose) rest state, and pose-dependent deformations are applied based on joint rotations.

While parametric models are prevalent, alternative representations exist. Some methods directly regress 3D mesh vertices. METRO by Lin et al. (2020*b*), for instance, uses a transformer-based approach to reconstruct 3D human pose and mesh vertices end-to-end from a single image, modeling interactions between vertices and joints without strictly relying on a parametric model structure. However, parametric models offer advantages for animation due to their inherent rigging structure.

The process of transforming captured pose and motion data into avatar animation typically involves mapping the estimated joint rotations or positions to the corresponding joints of the avatar's skeleton. If the avatar is based on a model like SMPL, the estimated SMPL pose parameters directly drive the animation via blend skinning. For custom or reconstructed models, weights are often transferred

from a fitted SMPL model to enable animation. The canonical pose serves as the reference state, and captured motion applies transformations relative to this pose, ensuring consistent animation across different models mapped to the same underlying structure.

Charles (2016) demonstrated a system for real-time full-body motion capture and mapping onto a virtual avatar using the Microsoft Kinect depth sensor and the Unity game engine. Their approach uses the Kinect Software Development Kit (SDK) to extract the user's skeleton data, which is then mapped onto a generic humanoid avatar in the Unity environment. The system allows for real-time control of the avatar's movements based on the user's actions, enabling interactive experiences in VR.

Alexiadis et al. (2017) present a method to map real-time body movements to a virtual environment using Kinect and Oculus Rift. It enables basic interactions with virtual objects, enhancing the VR experience. A study by Husein & Ciawi (2021) again demonstrate the use of Kinect V2 and Unity Engine for marker-less motion capture and animation. It highlights the ease of capturing motion and creating animations using depth sensors and game engines.

When we focus on more recent research that introduce novel pose estimation techniques which challenge state-of-the-art methods, the following studies are prominent. A research paper published in 2020, by Kocabas et al. (2019) introduces **VIBE: Video Inference for Human Body Pose and Shape Estimation**, which is a method that addresses the limitations of existing video-based 3D human pose and shape estimation approaches. Prior video-based methods often fail to produce accurate and natural motion sequences due to a lack of ground-truth 3D motion data for training. VIBE tackles this by leveraging a large-scale motion capture dataset (AMASS) along with unpaired, in-the-wild 2D keypoint annotations. Extensive experiments show that VIBE outperforms previous state-of-the-art methods on challenging 3D pose estimation benchmarks, demonstrating the benefits of using video over single-frame approaches.

In 2021, Choi et al. (2020) presented **TCMR: Beyond Static Features for Temporally Consistent 3D Human Pose and Shape**, which is a system designed to produce temporally consistent and smooth 3D human motion from video. Unlike prior video-based methods that strongly rely on the static features of the current frame, TCMR effectively leverages temporal information from past and future frames. It achieves state-of-the-art performance in terms of both temporal consistency and per-frame 3D pose and shape accuracy, outperforming previous video-based methods.

A study by Zhang et al. (2021), in 2023, introduced a novel way to align 3D models to humans in videos called **PyMAF: Pyramidal Mesh Alignment Feedback for Well-Aligned Body Model Regression**. It is a regression-based approach for recovering parametric 3D body models from monocular images. A key challenge in regression-based methods is achieving well-aligned results, as minor errors in the predicted model parameters can lead to noticeable misalignment between the estimated mesh and the input image.

A recent research published in 2024, by Shen et al. (2024) called **WHAM:**

World-grounded Humans with Accurate Motion describes a method that accurately and efficiently reconstructs 3D human motion in a global coordinate system from video. WHAM addresses several key limitations of previous 3D human pose estimation approaches, for example; It learns to lift 2D keypoint sequences to 3D using motion capture data and fuses this with video features to integrate motion context and visual information. Extensive experiments show WHAM outperforming prior video-based and image-based 3D human pose estimation methods.

FrankMocap: A Monocular 3D Whole-Body Pose Estimation System, presented by Rong et al. (2021), presents a fast and accurate whole-body 3D pose estimation system that can produce 3D face, hands, and body simultaneously from monocular images. Most existing 3D pose estimation methods focus on a single body part, neglecting the fact that subtle movements of the face, hands, and body are essential for capturing authentic human motion. FrankMocap takes a modular approach, first running separate 3D pose regression models for the face, hands, and body, and then composing the outputs via an integration module.

Toolkits like **MMPose** (Sengupta et al. (2019)), a Pytorch-based open-source library, provide a rich collection of algorithms for various pose estimation tasks, including 2D and 3D whole-body estimation, offering a flexible platform for research and development in this area.

Overall, these studies demonstrate the feasibility and potential of real-time motion capture and animation. By focusing on capturing and mapping the user's movements onto a generic avatar, they prioritize real-time interactivity and ease of use over photo-realistic appearance. However, the use of generic avatars may limit the sense of embodiment and presence experienced by the user, as the avatar does not fully match their individual appearance and body shape.

3.4 Capturing, modelling and animating (Realistic 3D representations)

A few studies have focused on all three steps that can be identified in the problem. They propose an end to end solution to capture motion, create a realistic model and animate the model.

Huang et al. (n.d.) proposed a novel approach for efficiently transforming an actor into a photorealistic, animated character using 3D scanning, motion capture, and free viewpoint video for integration in Unity. One of the main advantages of this approach is its ability to generate high-fidelity, interactive 3D character models that closely resemble the original actors, including their facial expressions and body movements.

Alexiadis et al. (2017) described an integrated platform for live 3D human reconstruction and motion capturing, targeting tele-immersion and future 3D applications. Their system combines multiple RGBD sensors, a novel calibration method, a robust and fast 3D reconstruction approach based on volumetric Fourier transform, and a generic skeleton tracking method using multiple depth streams. One of the key contributions of this work is the development of an end-to-end

pipeline for real-time 3D human reconstruction and motion tracking using commodity hardware (e.g., Kinect).

A survey by Caserman et al. (2020) provides a comprehensive overview of recent advances in full-body motion reconstruction for immersive VR applications. The authors analyze a wide range of techniques and systems for capturing, modeling, and animating human avatars, including marker-based and marker-less motion capture, 3D scanning and reconstruction, and real-time avatar control. One of the key findings of the survey is the growing trend towards marker-less and low-cost motion capture solutions, such as those based on depth sensors (e.g., Kinect) or monocular RGB cameras. An important trend identified in the survey is the increasing focus on creating personalized and realistic avatars that match the user's appearance and body shape.

Overall, these studies demonstrate the significant progress and potential of end-to-end solutions for capturing, modeling, and animating 3D human avatars in immersive applications. By combining advanced motion capture, 3D scanning, and animation techniques, these approaches enable the creation of realistic and interactive virtual characters that can closely match the user's appearance and movements.

3.5 Closely Related Work

Out of the comprehensive body of literature reviewed, several studies stand out as particularly relevant to the specific goals and constraints of this research project. These works offer critical insights into key aspects of the proposed system, from pose estimation and avatar representation to user experience in interactive AR.

Firstly, the field of real-time pose estimation for interactive applications is directly addressed by studies like **TensorPose: Real-time Pose Estimation for Interactive Applications** by Silva et al. (2019). TensorPose's focus on efficient CNN architectures for real-time multi-person pose estimation is highly relevant, as achieving high frame rates and temporal coherence is paramount for a responsive AR experience. Similarly, efforts to improve existing efficient frameworks, such as **Detection of 3D Human Posture Based on Improved MediaPipe** by Lin et al. (2023), are significant. This work highlights the strengths of Google's MediaPipe Kim et al. (2023) as a mature framework for resource-constrained environments, while also identifying its limitations in 3D pose detection and proposing improvements. These studies inform the selection and potential optimisation of the initial pose capture component, emphasizing the trade-offs between accuracy and real-time performance on devices with limited computational power, like those targeted by web browsers.

Regarding the challenge of creating and animating 3D human representations, the literature on parametric models and their use is highly pertinent. The SMPL: A Skinned Multi-Person Linear Model paper by Loper et al. (n.d.) is foundational, presenting a model widely used as a generic representation due to its compatibility with standard animation pipelines and its ability to disentangle pose and shape. Methods that leverage SMPL for 3D human avatar creation, such as **3D Human Avatar Digitization from a Single Image** by Li et al. (2019*a*) and **Video**

Based Reconstruction of 3D People Models by Alldieck et al. (2018), are closely related to the modeling aspect. These studies demonstrate pipelines for reconstructing personalized geometry and texture from minimal input (single image or video) by mapping to a canonical SMPL structure, providing insights into potential approaches for generating or adapting avatar models. While these methods may not be real-time in their entirety, they highlight the importance of the canonical representation for animating diverse avatars. Furthermore, generative models like **AvatarGen: a 3D Generative Model for Animatable Human Avatars** by Zhang et al. (2022), show the potential for creating diverse, animatable avatars from 2D data, again leveraging a canonical space, offering alternative modeling strategies.

The core challenge of mapping captured pose and motion to a generic avatar for real-time animation is explored in studies focusing on real-time movement mapping. Early work like **Real-Time Human Movement Mapping to a Virtual Environment** by Charles (2016) and related efforts using Kinect and Unity demonstrated the feasibility of capturing motion and animating a generic avatar in real-time. These works, while often relying on dedicated applications and depth sensors rather than web browsers and monocular RGB, established the fundamental pipeline of pose capture, skeleton mapping, and real-time animation. They inform the design of the transformation model by illustrating how captured joint data can drive an avatar's rig.

Addressing the complexities of handling diverse pose data and different skeleton formats, which is crucial if the project aims for generalisation or uses data from multiple sources, the paper **Learning 3D Human Pose Estimation from Dozens of Datasets using a Geometry-Aware Autoencoder to Bridge Between Skeleton Formats** by Sárándi et al. (2022) is particularly insightful. This approach to handling data inconsistencies and discovering a robust intermediate representation is highly relevant to the concept of mapping captured pose to a generic, canonical representation that is independent of the specific pose estimator's output format.

While not solely focused on pose estimation, the study **BioPose: Biomechanically accurate 3D Pose Estimation from Monocular Videos** by Koleini et al. (2025), is relevant as it represents the forefront of monocular 3D pose estimation, pushing towards biomechanical accuracy by integrating mesh recovery, neural inverse kinematics, and anatomical constraints. Although its complexity likely exceeds the resources available in a web environment, it highlights the limitations of simpler parametric models like SMPL in capturing true anatomical motion. This provides valuable context for evaluating the achievable level of realism when using more efficient, web-friendly pose estimation techniques and simpler avatar models.

Finally, research directly addressing avatar representation and user experience in mobile AR is highly relevant. The paper **Towards Avatars for Remote Communication using Mobile Augmented Reality** by Murugan et al. (2021) explores the design space of avatars for mobile AR remote communication, considering dimensions like body part visibility and movement type. It explicitly

discusses challenges related to limited pose data (often just phone tracking) and inferring body movements, which are constraints directly applicable to a web-based mobile AR system. Their investigation into how different avatar types impact social presence provides insights into the user experience aspects of the project.

Similarly, **Mini-Me: An Adaptive Avatar for Mixed Reality Remote Collaboration** by Piumsomboon et al. (2018) focuses on adaptive avatar design for MR collaboration, demonstrating how avatars can be designed to enhance social presence and convey non-verbal cues in AR, despite limitations like the limited FoV of AR headsets. This research offers valuable perspectives on designing the avatar and the overall AR experience to maximize realism and user interaction within the system's constraints.

In summary, these related works provide a strong foundation and direct parallels to the proposed research. They offer solutions and insights into efficient pose estimation, canonical avatar representation and animation, and the specific challenges of delivering a compelling user experience within the context of mobile AR, particularly in resource-constrained settings. Examining their methodologies and findings is crucial for guiding the design and implementation of the real-time 3D avatar modelling system in a web environment.

1. Real-Time Human Movement Mapping to a Virtual Environment (Charles (2016))
2. An integrated platform for live 3D human reconstruction and motion capturing (Alexiadis et al. (2017))
3. Video Based Reconstruction of 3D People Models (Alldieck et al. (2018))
4. A Process for the Semi-automated Generation of 3D Avatars from Images (Huang et al. (n.d.))
5. 3D Human Avatar Digitization from a Single Image (Li et al. (2019*a*))
6. End-to-End Human Pose and Mesh Reconstruction with Transformers (Lin et al. (2020*b*))
7. Ray3D: ray-based 3D human pose estimation for monocular absolute 3D localization (Zhan et al. (2022))
8. AvatarGen: a 3D Generative Model for Animatable Human Avatars (Zhang et al. (2022))
9. Real-Time Interaction for 3D Pixel Human in Virtual Environment (Deng et al. (2023))
10. Detection of 3D Human Posture Based on Improved MediaPipe (Lin et al. (2023))

No.	Year	Real-time	Platform	Equipment	Body Scope	Pipeline Stage Focus	Output
1	2016	Yes	PC + Unity	Kinect, Oculus Rift	Full Body	All (Capture, Model, Animate)	3D Pose
2	2016	Yes	PC Network (CUDA GPUs)	Multiple RGBD Cameras	Full Body + Env	All (Capture, Reconstruct, Animate)	3D Reconstruction
3	2018	No	PC	Monocular RGB Video	Full Body (Clothed)	Modeling (Reconstruction)	3D Mesh + Texture
4	2019	No	PC + Unity	3D Scanner, MoCap System	Full Body	Modeling + Animation Prep	3D Mesh + Rig
5	2019	Near Real-time	Mobile App + Server	Smartphone RGB Camera	Full Body	Modeling (Geometry + Texture)	3D Mesh + Texture
6	2021	Near Real-time (12–24 fps)	PC (GPU)	Monocular RGB Image	Full Body + Hands	Modeling (Mesh + Joints)	3D Mesh + Joints
7	2022	No	PC	Monocular RGB Camera	Full Body	Pose Estimation	3D Pose (Absolute)
8	2022	No (Generative)	PC (GPU)	2D Image Dataset	Full Body (Clothed)	Modeling (Generative)	3D Mesh + Texture
9	2023	Yes	PC + Unity + MediaPipe	Monocular Webcam	Upper Body + 2D Head	All (Capture, Model, Animate)	3D Pixel Avatar
10	2023	Yes	PC	Monocular Webcam	Full Body	Pose Estimation (Improved MediaPipe)	3D Pose
11	2022	No (Training Focus)	PC	Monocular RGB Camera	Full Body	Pose Estimation (Cross-Dataset)	3D Pose
12	2024	Yes (Claimed)	PC	Monocular RGB Video	Full Body	All (Capture, Model, Animate)	3D Biomechanical Pose + Mesh
13	2018	Yes (Claimed)	PC	Monocular RGB Image	Full Body	All (Capture, Model, Animate)	3D Mesh + SMPL Params
14	2021	Yes (Baseline Method)	PC	2D Pose Data	Full Body	Pose Estimation (Lifting + Normalization)	3D Pose
15	2016	Yes	PC Network	Custom Depth Cameras	Full Body + Env	All (Capture, Reconstruct, Animate)	3D Reconstruction

Table 1: Comparison of Existing Research on 3D Human Modeling and Animation

11. Learning 3D Human Pose Estimation from Dozens of Datasets using a Geometry-Aware Autoencoder to Bridge Between Skeleton Formats (Sárándi et al. (2022))

12. BioPose: Biomechanically-accurate 3D Pose Estimation from Monocular Videos (Koleini et al. (2025))

13. End-to-end Recovery of Human Shape and Pose (Kanazawa et al. (2017))

14. A Baseline for Cross-Database 3D Human Pose Estimation (Rapczyński et al. (2021))

15. Holoportation: Virtual 3D Teleportation in Real-time (Orts-Escolano et al. (2016))

3.6 Synthesis and Research Gap

The review of the existing literature reveals a field that is advancing at a rapid pace. We've seen powerful deep learning techniques emerge that can understand

and reconstruct human motion from a simple video feed. These include impressive real-time methods like **VIBE** (Kocabas et al. (2019)), **TCMR** (Choi et al. (2020)), **PyMAF** (Zhang et al. (2021)), and **WHAM** (Shen et al. (2024)), which can estimate full 3D joint positions and even detailed meshes. At the same time, we have practical, efficient frameworks like Google's **MediaPipe** (Kim et al. (2023)) and comprehensive toolkits like **MMPose** (Sengupta et al. (2019)) that are specifically designed to run on the devices we use every day, including our phones.

In parallel, the way we create and use 3D models for animation has also matured. Parametric models like the **SMPL** model have become a common language for researchers, providing a controllable and standardized way to represent the human body. This makes it easier to apply the captured motion to different characters. We've also seen amazing research into creating personalized avatars from just a single photo, using techniques like Generative Adversarial Networks (GANs) to create realistic and animatable characters (Alldieck et al. (2018), Li et al. (2019*b*), Zhang et al. (2022)). All the individual components have advanced separately: we have ways to capture motion, ways to represent the body, and powerful graphics engines in our browsers to display the results.

However, when we look closer at how these pieces fit together, we see significant gaps. The primary challenge and the central motivation for this thesis lie in bridging the gap between the world of high-end academic research and the world of practical, accessible, real-world applications, especially within the unique and restrictive environment of a web browser.

3.6.1 The Conflict Between Performance and Quality in Real-Time Systems

A major theme emerging from the literature is the constant battle between performance and quality. This is especially true for real-time systems that need to run on resources-constrained devices such as smartphones. On one hand, we want our avatars to be driven by the most accurate and detailed motion capture data possible. On the other hand, the methods that produce this high-quality data are often very slow and require powerful computers.

This brings us to the first major hurdle: choosing a pose estimation method. Google's MediaPipe framework is a fantastic example of a solution designed for performance. It offers pipelines that can run efficiently on a phone's CPU, making it one of the few practical choices for a web-based application (Kim et al. (2023)). However, as studies like Lin et al. (2023) have pointed out, this efficiency can come at the cost of accuracy. The 3D pose data from such lightweight models may not be as precise as that from more complex methods. This trade-off is fundamental: to achieve the speed needed for a smooth real-time experience on the web, we often have to accept input data that is less than perfect.

Furthermore, the fundamental challenges of seeing in 3D from a 2D image don't disappear, even with powerful models. Problems like depth ambiguity (is an arm bent forward or is it just short?), self-occlusion (a user's arm hiding their torso), and sensitivity to different types of clothing or poor lighting are persistent issues in monocular vision (Liu, Shen, Wang, Chen, ching Cheung & Asari (2021),

Chen et al. (2020)). Even the most advanced research systems struggle with these problems. This means that any practical system built for real-world use cannot assume it will receive a perfect, complete 3D skeleton at all times. It must be designed to be resilient and to handle messy, incomplete data gracefully.

3.6.2 The Challenge of Heterogeneity: Mismatched Skeletons and Diverse Data

The problem is compounded by a lack of standardization. Different research projects and datasets often use different skeleton formats. For example, some datasets define 17 keypoints, while others use 25 or more, and the specific definition of joints like the "hip" or "spine" can vary. As research by Sárándi et al. (2022) and Rapczyński et al. (2021) has shown, simply combining data from these different sources requires careful "harmonization" work to align the different skeleton structures.

This issue directly impacts our goal of building a flexible web application. One user, on a powerful laptop, might be able to run a pose estimator that outputs a detailed 33-point skeleton. Another user, on an older phone, might need to use a much lighter model that only outputs 17 keypoints, and only in 2D. How can a single application handle both of these inputs? Handling these skeleton mismatches and ensuring a smooth animation transfer in both cases is a significant challenge. Any robust system needs a way to work with these different "languages" of pose data. This problem is also highlighted in research that tries to generate a full-body pose from very limited data, such as just the phone's position in mobile AR (Murugan et al. (2021)). This shows that there is a need for intelligent systems that can infer a plausible full motion from a sparse input.

3.6.3 The Web Browser as a Challenging Environment

Finally, deploying a complete avatar pipeline in a web browser introduces its own unique set of severe constraints. While technologies like WebGL provide the ability to render 3D graphics, the performance is not the same as a native application. As a detailed study by Bi et al. (2023) on WebXR performance shows, browser-based 3D applications are limited by JavaScript execution speed, browser overhead, and the wide variety of device capabilities, especially on mobile.

Their study provides crucial benchmarks that highlight the "resource-constrained" nature of the web. They found that while rendering simple 3D scenes is feasible, performance degrades very quickly as the complexity of the 3D models and the scene increases. Critically, their findings suggest that often the primary bottleneck for a web AR application is not the camera feed itself, but the 3D rendering load. This means that every part of our pipeline—from running the pose estimation model (using libraries like TensorFlow.js) to processing the data and finally rendering the avatar—must be extremely efficient. We cannot afford to waste any computational resources.

This brings the user experience into sharp focus. The quality and responsiveness of the avatar directly influence the level of immersion and realism for the

user. While highly realistic avatars created from 3D scans look amazing, they are often too complex to be rendered smoothly in a web browser in real-time (Huang et al. (n.d.), Caserman et al. (2020)). This forces a compromise: simpler, more "cartoony" avatars are easier to animate, but may reduce the user's sense of embodiment or "presence" (Charles (2016)). Finding the right balance between realism and performance is a key challenge for any web-based system.

3.6.4 Identifying the Research Gap

After analyzing the literature, a clear picture emerges. The field is full of powerful, specialized solutions, but there is a lack of integration between them, especially for the web platform. The academic, high-fidelity methods are too slow for the web, and the fast, web-friendly methods are not flexible or robust enough on their own.

This leads us to the **significant research gap** that this project addresses: there is a lack of a readily available, end-to-end pipeline specifically designed to work within the web browser that can **flexibly handle diverse pose estimation inputs** and **intelligently generate a complete, plausible animation from potentially imperfect or sparse data.**

Specifically, the gap lies in the absence of a single, cohesive system that does the following:

- **Adapts to Different Inputs:** Current solutions are usually hard-coded to one specific input source. There is no "universal translator" or middleware that can take data from a variety of common pose estimators (e.g., a 17-point 2D skeleton from MoveNet, a 33-point 3D skeleton from BlazePose) and convert them into a single, standardized format. This forces developers into a rigid choice and limits user accessibility.

- **Intelligently "Fills in the Blanks":** A web-based system must expect imperfect data. It needs a way to handle this, for example, by "lifting" 2D keypoints into 3D space, or by estimating the position of an occluded arm. While complex AI models can do this, they are too slow for the browser. The gap is the need for a system that can perform this inference using lightweight, efficient techniques based on biomechanical rules and heuristics.

- **Provides a Fully Integrated and Optimized Solution:** While a developer could technically try to stitch together a pose estimator, a 3D library, and their own custom logic, this is a massive undertaking. There is no existing, open-source pipeline that combines all these pieces and is optimized to work together efficiently under the demanding constraints of the web browser.

Therefore, this research project aims to fill this specific gap. It is not about inventing a new pose estimator. It is about designing, implementing, and evaluating the crucial **middleware**—the smart "glue"—that connects the pieces. By creating a generalized transformation model, this work seeks to make real-time 3D avatar

animation more robust, flexible, and most importantly, accessible to anyone with a webcam and a web browser.

4 Research Methodology

The primary goal of this research is to create and evaluate a novel software artifact—a generalized pipeline for real-time 3D avatar animation. Given this objective, the project adopts the Design Science Research (DSR) methodology. DSR is a well-established paradigm in technology and information systems research that focuses on the design, implementation, and evaluation of artifacts intended to solve specific, identified problems. This approach is inherently iterative and practical, directly aligning with the engineering challenges of this thesis. The research was executed in a series of structured phases, allowing for an adaptive approach where findings from earlier stages directly informed the direction of subsequent work.

4.1 Overall Research Approach

The research process began with an initial plan to identify a single, optimal pose estimation technique and build a solution around it. However, early investigations, detailed in Chapter 5, revealed significant practical challenges related to the diversity of available tools and their performance trade-offs. This led to a crucial pivot in the research strategy. The focus shifted from optimizing for a single input to designing a flexible, generalized system capable of handling multiple, heterogeneous inputs.

The refined research plan was structured into three main phases:

1. **Problem Analysis and Technology Evaluation:** This phase involved a deep dive into existing solutions to understand the technical landscape. It combined a thorough literature review (Chapter 2) with hands-on technical experiments to identify the capabilities and limitations of available tools for pose estimation, 3D modeling, and web-based rendering. Generated code

2. **Artifact Design and Development:** Based on the gaps identified in the first phase, the core of the research was the design and construction of the software pipeline. This involved defining the system architecture, developing the core algorithms for data processing and inference, and implementing the complete end-to-end solution in a suitable web-based environment.

3. **Evaluation:** The final phase focused on assessing the developed artifact. The goal was to measure its performance, evaluate the quality of the output, and determine its effectiveness in solving the initial problem. Use code with caution.

4.2 Evaluation Plan

To measure the success of the developed pipeline and answer the research questions, a multi-faceted evaluation strategy was designed. The evaluation focuses on three key areas:

1. **Performance Benchmarking:** The real-time viability of the system is critical. The primary metric for evaluation is Frames Per Second (FPS). The FPS of the entire pipeline will be measured on representative hardware (a modern laptop and a smartphone) using different input pose estimators (e.g., lightweight MoveNet vs. heavyweight Holistic) to quantify the system's performance under various loads. Generated code

2. **Functional Correctness and generalisation:** The pipeline's core novelty is its flexibility. Its success will be evaluated by its demonstrated ability to:

    - Successfully ingest and process data from multiple, distinct pose estimators with different skeleton formats (e.g., 17-point 2D, 33-point 3D).
    - Generate a complete, animated output even when provided with sparse or incomplete input data.

3. **Qualitative Assessment of Animation Plausibility:** The final output must be visually acceptable. The quality of the animation will be evaluated through direct observation, focusing on aspects like smoothness, absence of major anatomical errors (e.g., unnaturally stretched limbs), and the general coherence of the motion, especially when inferred from sparse data. Use code with caution.

This structured methodology, combining systematic evaluation with iterative design, ensures that the research process is both rigorous and directly tied to producing a practical, well-validated solution to the research problem.

5 System Design and Development

Following the research methodology outlined in the previous chapter, a novel software pipeline was designed and developed to address the identified research gap. This chapter details the architecture of this artifact, the technical explorations that informed its design, and the implementation of its core components. The system was built from the ground up as a web-native solution, prioritizing modularity, flexibility, and real-time performance.

5.1 Preliminary Experiments

The design of the final pipeline was heavily informed by a series of preliminary experiments aimed at understanding the practical challenges of each part of the system.

5.1.1 Motion Capture Technique Selection and Adaptation

As the initial plan was to select only one ideal technique and build upon it, the first step in the process involved selecting and adapting a motion capture technique that could reliably capture human expressions, poses, and movements using just a webcam. This required evaluating various state-of-the-art pose estimation algorithms, each with unique capabilities and trade-offs in terms of performance, accuracy, and computational requirements.

High-Accuracy Academic Models: Methods like VNect (2017), a pioneering real-time method for 3D pose estimation from a single camera (Mehta et al. (2017)), OpenPose (2017), a robust multi-person 2D detection system (Cao et al. (2016)), and HybrIK (2021), an accurate 3D pose and shape estimation solution (Li et al. (2020)), were reviewed. While these systems represent the state-of-the-art in accuracy, their reliance on C++/Python environments and powerful GPUs made them unsuitable for direct client-side web deployment.

Web-Compatible Real-Time Models: The practical investigation focused on models that could run directly in the browser. MoveNet (2021): This model offers ultra-fast pose detection, making it highly suitable for mobile applications. It supports both WebGL and CPU execution via TensorFlow.js, providing a sparse but very fast 17-point 2D skeleton (*MoveNet: Ultra fast and accurate pose detection model. | TensorFlow Hub* (n.d.)). MediaPipe BlazePose (2020) Holistic (2021): These Google models are designed for on-device inference. BlazePose offers a 33-point 3D skeleton, providing a good balance of detail and performance. Holistic is the most comprehensive, offering 543 landmarks for the body, face, and hands, but this richness comes at a higher computational cost (Kim et al. (2023)). Their native JavaScript availability made them prime candidates.

YOLO-Pose (v8, v11): Extending the popular YOLO family, these models provide fast, 17-point 2D keypoint detection. Integrating YOLO-Pose required a non-trivial conversion process from its native PyTorch format to TensorFlow.js, but it served as another important data point for a fast, sparse input source (Maji et al. (2022)).

The following table summarises the features of above pose estimation techniques.

Year	**Name**	**Method**	**Runs On**	**No. of Points**	**Evaluation**
2017	VNect	C++, Unofficial Python and Tensorflow based implementation	CPU	21 (3D)	
2018	AlphaPose	CNN based - Runs on Linux with PyTorch and torchvision	GPU with PyTorch	136, 26, 17	82.1 mAP on MPII dataset
2020	MediaPipe BlazePose	Tensorflow-JS	CPU, GPU	33 (3D)	
2021	MoveNet	Tensorflow-JS (WebGL or Wasm)	GPU with WebGL or CPU	17 or 33 with BlazePose	30+ fps on mobile devices, 100+ fps on desktops
2021	HybrIK and HybrIK-X	Based on a hybrid inverse kinematics (IK) to convert accurate 3D keypoints to parametric body meshes	GPU with PyTorch		13.2 mm MPJPE and 21.9 mm PVE on 3DPW dataset
2017	OpenPose	Python and C++ APIs, also unofficial Tensorflow based implementation. Windows and Ubuntu	CPU, GPU	135 (2D Keypoints)	
2021	HyperPose	Python, C++	CPU, GPU		
2022	MediaPipe Holistic	Python, JS	CPU, GPU	543 (3D)	20 - 30 fps on mobile devices
2023 (v11)	YOLO-Pose	CNN-based single-stage object detection and keypoint regression	GPU (PyTorch/CUDA), CPU (Python), Browser (TF.js WebGL/Wasm after conversion)	17 (2D, COCO)	Very High FPS (Real-time), Competitive mAP on COCO keypoints (varies by model size)

Table 2: Comparison of Applicable Human Pose Estimation Methods

The critical outcome of this investigation was the empirical confirmation that a trade-off between performance and detail is unavoidable. This finding invalidated the initial plan to select one ”optimal” model and provided the core motivation for designing a generalized pipeline that could adapt to this diversity.

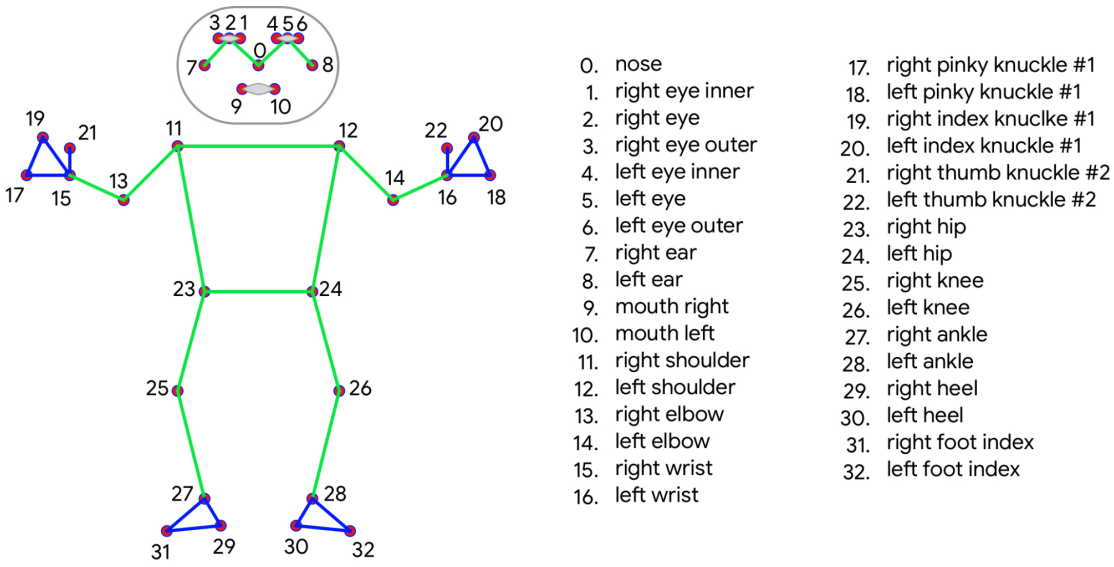


Figure 2: Topology of Blazepose (From: Research (n.d.))

5.1.2 Data Extraction and Avatar Modeling

According to the initial plan, after selecting the MediaPipe BlazePose solution as the primary motion capture method due to its balance of accuracy and efficiency, I focused on real-time data extraction and feeding it into the 3D model, the goal is to be able to transform and map these keypoints to any model, in real-time.

Exploring 3D Models to feed the MediaPipe Keypoints Output After obtaining the output of keypoints from MediaPipe's integrated solution, I started to explore the next step of integrating these outputs with 3D models to drive avatar animations. This exploration revealed the complexity and variety in the 3D modeling ecosystem, primarily due to differing formats, coordinate systems, bone schemas, and rig structures utilized in these models.

During my experiments, I realized that converting MediaPipe's landmarks directly into blendshape weights, which are used in 3D modeling for facial expressions, is a complex task. Blendshapes are essentially predefined deformations that morph a 3D model's mesh into different shapes for expressions and movements, commonly used for animating characters' faces.

Beyond blendshape mapping, a core challenge emerged when considering pose estimators that only provide 2D keypoints (like MoveNet or YOLO-Pose). How can these be 'lifted' to 3D for avatar animation? State-of-the-art research often employs deep learning models specifically trained for this 2D-to-3D lifting task (Rapczyński et al. (2021)). For instance, lightweight networks can infer 3D coordinates from 2D inputs. However, obtaining pre-trained weights suitable for web deployment (e.g., converted to TensorFlow.js) and ensuring real-time performance within browser constraints posed significant challenges. Accessing specific model weights proved difficult, and the computational cost of even lightweight neural networks can be prohibitive on resource-constrained mobile devices. This further steered the methodology towards exploring heuristic and bio-mechanically informed geometric approaches for 3D reconstruction that could run

efficiently in JavaScript.

Approaches for Blendshape Generation: According to a discussion thread on the web (qhanson (2022)), there exists several ways to convert face meshes and keypoints to blendshapes to drive 3D models.

Direct Math Conversion from Mesh Landmarks:

- Kalidokit: An open-source project designed to convert face, hand, and body landmarks from models like MediaPipe into data that can drive 3D models. It provides utilities to map face landmarks into blendshape data for animation. The implementation is done in TypeScript, making it directly usable in browser environments.
- MeFaMo (MediaPipe Face Model): Another project with a similar approach to converting MediaPipe outputs to usable 3D facial animations. The implementation is done in Python.

AI Model-Based Conversion:

- Mocap4Face: A solution that uses facial motion capture data to produce blendshapes for animation. It provides real-time FACS-derived blendshape coefficients, and rigid head pose in 3D space.
- AvatarWebKit: A project focused on creating web-based avatars with facial motion capture capabilities, it does not provide rigid body frame and hand positons.
- NVIDIA's *Maxine AR SDK* is a recent solution, which predicts blendshapes, though it is tailored for use on NVIDIA GPUs with CUDA support.

While these methods provided a starting point, I faced several challenges:

Many of the open-source projects such as Kalidokit and Mocap4Face appeared to be inactive or not well-maintained. NVIDIA's solution, while promising, required hardware constraints (NVIDIA GPUs), limiting its broader applicability. Most of the other projects were archived and had stopped giving API access. However, the Kalidokit code provided a solid foundation for a mathematical transformation model that can map all keypoints for face, hands and body pose from Tensorflow-js or MediaPipe 3D or 2D keypoints to a standard format compatible with VRM based models (explained later), that can be used to animate those models directly.

Discoveries regarding 3D Model Formats

Through my exploration, I noticed that MediaPipe provides normalized landmark outputs for facial, pose and hand features, but integrating these outputs with 3D models wasn't straightforward due to different skeleton topologies.

Skeleton Topologies in Virtual Character Models BlazePose Skeleton Representation: For example, BlazePose, offers a skeleton structure very similar to the standard 17 COCO keypoints but lacks a spine bone, while most virtual character skeletons include it. This makes direct mapping challenging.

Virtual character 3D models typically come in formats like VRM, FBX, and GLTF, each with unique bone structures, coordinate systems, and other parameters.

VRM: A New Schema for Simplifying 3D Avatar Handling

During my explorations, I came across the VRM file format, which was specifically designed to standardize humanoid 3D avatars in VR applications. It offers several benefits:

Research > FYP_20000545 > Codes > 3D-Estimation > Kalidokit > kalidokit > src > PoseSolver > TS calcArms.ts > calcArms

```ts
import Vector from "../utils/vector";
import { clamp } from "../utils/helpers";
import { Results, Side } from "../Types";
import { RIGHT, LEFT } from "./../constants";
import { PI } from "./../constants";

/**
 * Calculates arm rotation as euler angles
 * @param {Array} lm : array of 3D pose vectors from tfjs or mediapipe
 */
export const calcArms = (lm: Results) => {
    //Pure Rotation Calculations
    const UpperArm = {
        r: Vector.findRotation(lm[11], lm[13]),
        l: Vector.findRotation(lm[12], lm[14]),
    };
    UpperArm.r.y = Vector.angleBetween3DCoords(lm[12], lm[11], lm[13]);
    UpperArm.l.y = Vector.angleBetween3DCoords(lm[11], lm[12], lm[14]);

    const LowerArm = {
        r: Vector.findRotation(lm[13], lm[15]),
        l: Vector.findRotation(lm[14], lm[16]),
    };
    LowerArm.r.y = Vector.angleBetween3DCoords(lm[11], lm[13], lm[15]);
    LowerArm.l.y = Vector.angleBetween3DCoords(lm[12], lm[14], lm[16]);
    LowerArm.r.z = clamp(LowerArm.r.z, -2.14, 0);
    LowerArm.l.z = clamp(LowerArm.l.z, -2.14, 0);
    const Hand = {
        r: Vector.findRotation(
            Vector.fromArray(lm[15]),
            Vector.lerp(Vector.fromArray(lm[17]), Vector.fromArray(lm[19]), 0.5)
        ),
        l: Vector.findRotation(
            Vector.fromArray(lm[16]),
            Vector.lerp(Vector.fromArray(lm[18]), Vector.fromArray(lm[20]), 0.5)
        ),
    };

```

Figure 3: Implementation of transformation model in Kalidokit (From: yeemachine on GitHub (n.d.))

Platform Independence: VRM models can be used across various VR/AR platforms and applications. Based on glTF2.0: This format builds on the glTF2.0 standard and adds additional constraints for humanoid characters. Standardized Configurations: The VRM format addresses scale, coordinate systems, and bone structure, making integration across different platforms easier. Avatar Customisation: VRM allows for the customisation of expressions, gaze settings, shaders, and more, all within a single file. This standardisation makes it easier to map MediaPipe outputs to a common avatar format. Licensing and Rights: VRM allows embedding license and usage rights specific to avatars, which is crucial for content creators and VTubers.

Challenges with Other 3D Formats and Skeletons

Mixamo and FBX Models: Many 3D models from platforms like Mixamo use a 65-point bone schema, differing significantly from BlazePose's skeleton structure and also the standard VRM structure. This variation in bone schemas complicates the mapping process.

GLTF and FBX Formats: Generic 3D models in these formats often have unique skeleton setups, making direct animation from MediaPipe outputs complex without intermediate transformations.

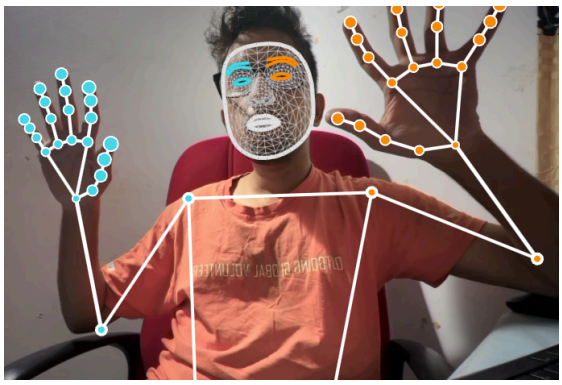


Figure 4: Output from MediaPipe holistic model

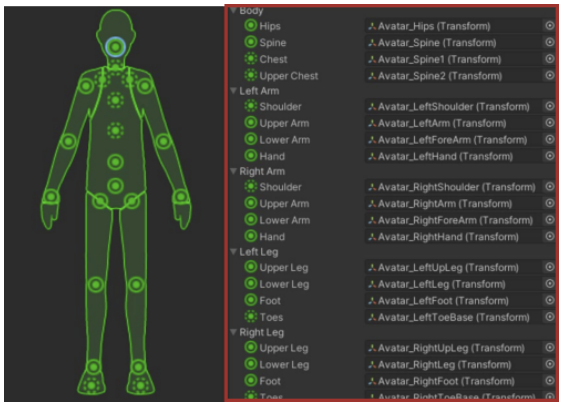


Figure 5: Skeletal points structure from VRM Standard (From HTC-Corporation (n.d.))

The exploration of converting MediaPipe outputs to 3D model blendshapes unveiled a complex landscape with various challenges, primarily due to differing standards in skeleton topology and format specifications. The use of standardized formats like VRM and development of a lightweight end to end framework offer potential paths forward.

5.1.3 Experimenting with Different Types of Rendering in Web Browsers

To render and display the models in web browsers, I also explored different tools and frameworks that are tailored for web-based 3D rendering and AR/VR applications.

WebXR WebXR (Web Extended Reality) is a web API that supports the rendering of virtual and augmented reality experiences in web browsers. It allows users to experience immersive 3D environments directly through their browsers using VR headsets, AR-capable devices, or standard screens for non-immersive experiences. My focus on WebXR was driven by the need to make my avatar animations accessible and interactive within AR experiences utilizing web browsers.

A-Frame A-Frame is a declarative web framework built on top of three.js that makes it easier to create WebXR applications. A-Frame provides an easy-to-use markup-like syntax to define 3D scenes. While A-Frame's ease of use allowed me to quickly prototype 3D models, I discovered that it might lack some advanced customisation and performance optimisation features required for highly detailed and complex avatars.

Figure 6: Structure of sample free 3D model from Mixamo (*Mixamo* (n.d.))

Figure 7: Structure of sample free generic 3D Model

Three.js Three.js is a popular library for rendering 3D content on the web. When used for rendering 3D models on the browser, Three.js provided extensive flexibility and robust rendering capabilities, making it ideal for achieving the level of detail needed here. I found it straightforward to implement WebXR with Three.js for immersive VR/AR experiences. The rich library of utilities and plugins further enhanced my ability to work with complex 3D models, animations, and textures.

5.1.4 Exploring 3D Human Models

In addition to experimenting with pre-rendered humanoid models from various sources such as Mixamo, Sketchfab, Turbosquid etc., I explored the SMPL model which was a 3D human model designed to achieve realistic human body representations.

SMPL provides a highly realistic representation of human body shapes and movements, making it an ideal choice for avatars intended to mimic natural human behavior and appearance. The model can be posed with natural, pose-dependent deformations and even exhibits soft-tissue dynamics, making it efficient for animation while maintaining a high degree of realism. SMPL's simplicity in terms of its low polygon count, clean quad structure, and standard rig makes it accessible to both animators and computer vision researchers (Loper et al. (n.d.)).

SMPL model can be used for personalisation of output based on input data (live video of human), such as adapting body shapes and poses according to a specific human subject. Generating personalized 3D models with dynamic shapes and textures etc. is also a very interesting research topic.

Summary of the Problem and Solutions The key challenge lies in unifying different motion capture standards, each with distinct kinematic topologies. Different virtual characters require different sets of operations to perform the same actions. Additionally, avatar animation applications demand real-time interaction, efficient motion estimation, and immersive environments, making the task computationally expensive.

Figure 8: Rig keypoints structure from VRM Humonoid Spec

The proposed solution is an efficient end to end framework for Real-Time Avatar Animation with any type of 3D humanoid model, that can be run on web browsers, even in smartphones:

- Track human motion using keypoint estimation models for the human body and optionally face and hands.
- Create a flexible middleware like adaptor to convert input keypoints into a topology that can drive virtual character models with any skeleton and rig points structure.
- Convert keypoints and rotation matrices into formats compatible with target avatars.
- Animate the humanoid models with the outputs of the mapping and transformation algorithm (similar to Kalidokit (yeemachine on GitHub (n.d.))) and render them.

5.2 Experimental Setup and Development Environment

After preliminary experiments and the revised synthesis of the research gap, the objective of the project was to conceptualize, develop, and evaluate a generalized real-time pipeline capable of animating 3D humanoid avatars using standard webcam input and a variety of different pose estimation techniques, specifically within the demanding resource constraints of modern web browsers operating on various hardware, including mobile devices.

The inherent challenges associated with monocular computer vision, real-time performance budgets, and the heterogeneity of pose estimation techniques required an iterative and adaptive research methodology. This section details the structured approach I undertook, evolving from initial explorations to the design and implementation of a novel middleware architecture. It covers the experimental setup, the rationale behind key technical decisions, the extensive offline data analysis performed, the specific algorithms developed for each pipeline stage, and acknowledgment of the encountered challenges and limitations.

My investigation began with preliminary experiments aimed at identifying suitable technologies for each facet of the problem: capturing user motion, representing that motion digitally, and animating a 3D avatar. From the outset, the goal was a web-native solution, prioritizing platform independence, real-time responsiveness essential for AR interaction, visual plausibility, and ease of integration.

To facilitate efficient development and iterative testing crucial for this research, I established a modern web-based development environment. A prototype web application served as the core testing platform, built using fundamental web technologies: HTML for structure, CSS for styling, and plain JavaScript (ES6+) for all core logic and pipeline implementation. This choice deliberately avoided reliance on large frontend frameworks to maintain transparency and control over performance characteristics.

The project structure was managed using the Node Package Manager (npm), and Vite was selected as the build tool and development server. Vite's native ES module support and extremely fast Hot Module Replacement (HMR) significantly accelerated the development cycle compared to traditional bundlers, proving invaluable for rapid prototyping and testing of complex pipeline components. For 3D graphics rendering and manipulation within the browser, THREE.js was chosen due to its maturity, extensive features, large community support, and performance capabilities leveraging WebGL. To specifically handle the loading and interaction with humanoid avatars, I integrated the *three.js* and *three-vrm* libraries, which provides robust support for the increasingly popular VRM 1.0 standard – a format specifically designed for interoperable 3D humanoid avatars. All machine learning model inference (for pose estimation) was handled using TensorFlow.js, leveraging its WebGL backend for GPU acceleration where possible.

5.3 Phase 1: Foundational Research Pose Estimation Integration

This initial phase aligned with the original proposal's steps of method selection and data extraction, but evolved significantly based on preliminary findings.

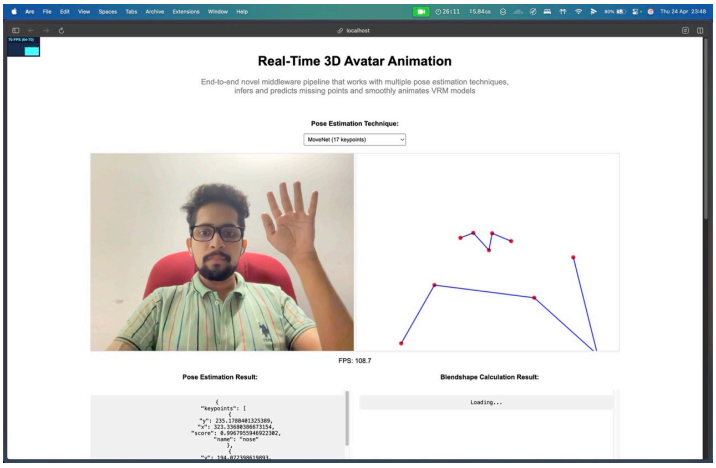


Figure 9: Developed prototype web-app (on a computer)

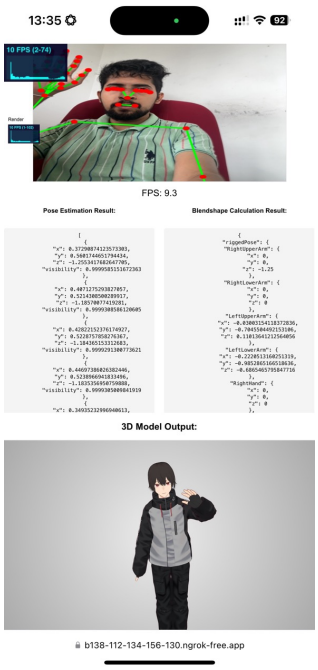


Figure 10: Developed prototype web-app (on a smartphone)

5.3.1 Exploring Pose Estimation Techniques for Web Environments

I began by surveying and experimentally evaluating a range of real-time human pose estimation algorithms accessible from a web context. The literature presents a spectrum of approaches, from complex, high-accuracy methods often requiring significant computational power, to lightweight models optimized for edge devices.

My investigation included:

MediaPipe Suite (BlazePose, Holistic): These solutions by Google (Lugaresi et al. (2019), Kim et al. (2023)) were prime candidates due to their design for on-device inference and direct JavaScript/WebAssembly (Wasm) availability. BlazePose offers efficient 33-keypoint 2D/3D body tracking. Holistic provides a comprehensive solution combining pose, face (468 landmarks), and hands (2x21 landmarks), totalling 543 landmarks. Integrating Holistic provided rich data but confirmed the potential performance bottleneck on lower-spec devices due to its computational load. I successfully integrated both, noting the difference between Holistic's normalized image landmarks (poseLandmarks) and its root-relative pseudo-meter world landmarks (poseWorldLandmarks).

TensorFlow.js Models (MoveNet): I integrated MoveNet (specifically, the faster 'Lightning' variant) *MoveNet: Ultra fast and accurate pose detection model. | TensorFlow Hub* (n.d.)) via TensorFlow.js. Its speed and simplicity in providing 17 standard COCO-format 2D keypoints made it an excellent baseline for testing the pipeline's ability to handle sparse, 2D-only input.

YOLO-Pose: To further test generalisation and incorporate another popular efficient architecture, I worked on the complex task of integrating a YOLO-Pose model within the browser. As pre-trained TensorFlow.js versions were not readily available,

this involved obtaining the model weights (e.g., PyTorch), exporting to ONNX format, converting the ONNX model first to a TensorFlow SavedModel, and finally using the tensorflowjs converter tool to generate a web-compatible GraphModel (model.json and shard files). This multi-step conversion process was complex and required careful dependency management but ultimately provided another 17-keypoint 2D input source for the browser.

Other Methods Considered: I also researched methods like VNect (Mehta et al. (2017)), OpenPose (Cao et al. (2016)), HybrIK (**?**), and frameworks like AlphaPose (Fang et al. (2022)). While highly capable, their primary implementations in Python/C++ and reliance on specific GPU libraries (like TensorRT for HyperPose) made them unsuitable for direct, real-time deployment within the target browser environment, without significant, complex porting efforts beyond the scope of this project. However, studying their architectures (e.g., heatmap regression, Path Affinity Fields (PAF), analytical IK integration in HybrIK, Non-Maximum Suppression (NMS) techniques in AlphaPose) provided valuable conceptual understanding.

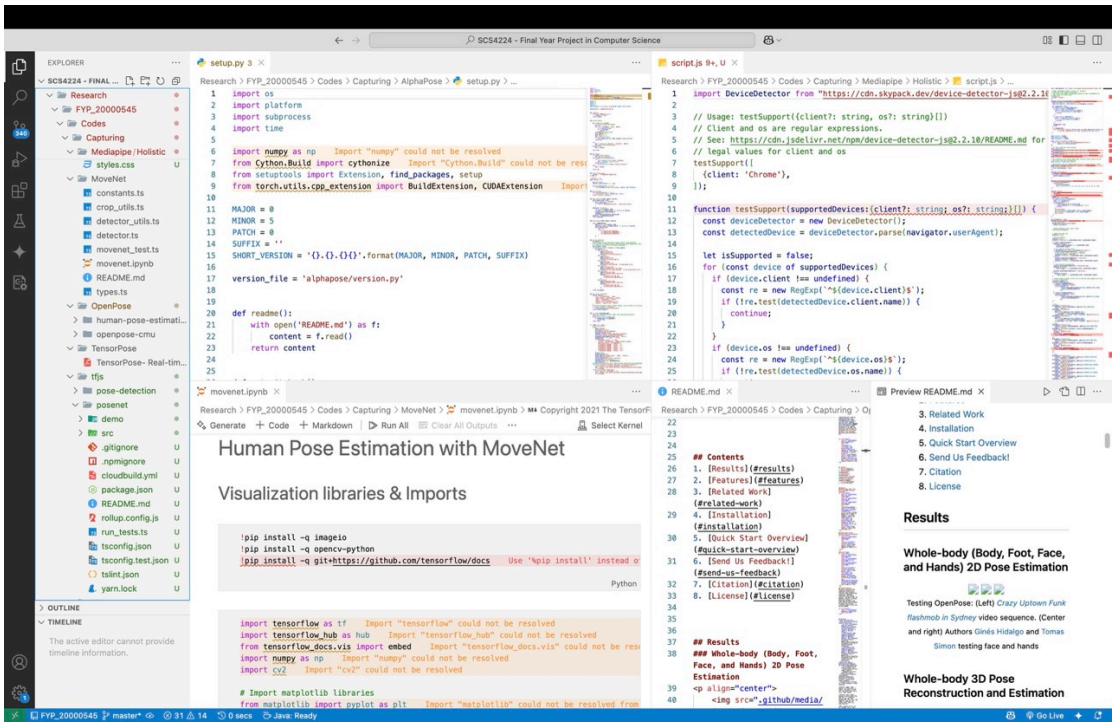


Figure 11: Exploring code level implementation of various pose estimation techniques

5.3.2 The Shift Towards generalisation

Initial experiments confirmed the feasibility of driving a VRM avatar using MediaPipe Holistic and existing mapping logic (similar conceptually to libraries like Kalidokit). While functional, simply replicating this pathway offered limited novelty. Kalidokit, for instance, is tightly coupled to MediaPipe's specific 543-landmark output and primarily

targets VRM avatars for output. The key realisation was the significant heterogeneity among readily available web-compatible pose estimators. They varied drastically in:

- Keypoint count (17 vs 33 vs 543).
- Skeleton topology (COCO vs BlazePose/Holistic definitions differ, especially around hips/spine).
- Coordinate systems (2D pixels, 2D normalized [0,1], 3D pseudo-meters root-relative, 3D normalized image space).
- Confidence metrics (score vs visibility).

Further complicating the selection process is the diversity in skeleton formats output by different models. Common standards include COCO (17 keypoints, used by MoveNet, YOLO-Pose, PoseNet), MPII (16 keypoints), H36M (17 or 32 keypoints, often used in research datasets), and the more detailed BlazePose (33 keypoints) and MediaPipe Holistic (543 landmarks). While some overlap exists (e.g., basic limb joints), differences in joint definitions (especially around the hips and spine) and the sheer number of points pose significant integration challenges (see Fig. 12). Early quantitative comparisons, such as those by Jo & Kim (2022), highlighted trade-offs: evaluating OpenPose, PoseNet, and MoveNet variants on mobile-centric tasks, they found MoveNet 'Lightning' offered the highest speed, while PoseNet achieved higher accuracy (97.6%) on their dataset compared to MoveNet Thunder (80.6%), MoveNet Lightning (75.1%), and OpenPose (86.2%, though it uniquely supported multi-person). OpenPose, with its detailed Body25 model (including neck, mid-hip, and foot keypoints), offers high granularity but often demands more computational resources. Attempts to directly port complex C++ based models like OpenPose to the web proved challenging due to dependencies and performance hurdles (Tejo (2024)). Related research explores advanced techniques like VIBE (Kocabas et al. (2019)) for temporal smoothness or geometry-aware autoencoders to bridge skeleton formats, but these often require significant computational power unsuitable for our target web environment. Projects like MMPose (Sengupta et al. (2019)) and MMHuman3D *MMHuman3D: OpenMMLab 3D Human Parametric Model Toolbox and Benchmark* (2021) offer comprehensive toolkits but are primarily Python-based.

Furthermore, resource constraints are paramount. Users on lower-end devices might need to select a faster, sparser estimator like MoveNet, sacrificing detail for performance. A system locked to Holistic would exclude these users.

This led to a fundamental shift in the research focus: from finding one optimal estimator to designing a flexible, generalized middleware pipeline capable of ingesting input from various pose estimation methods (that can be trained on various datasets). This required developing novel components for input abstraction, 3D lifting, missing data inference, and standardized rigging, forming the core contribution of this work.

This heterogeneity, coupled with the practical challenges of deploying complex models like OpenPose or deep learning-based 2D-to-3D lifters (discussed below) in the browser, solidified the decision to move away from selecting a single 'optimal' estimator. Instead, the research pivoted towards designing a flexible middleware capable of handling this input diversity and compensating for sparser inputs through intelligent processing.

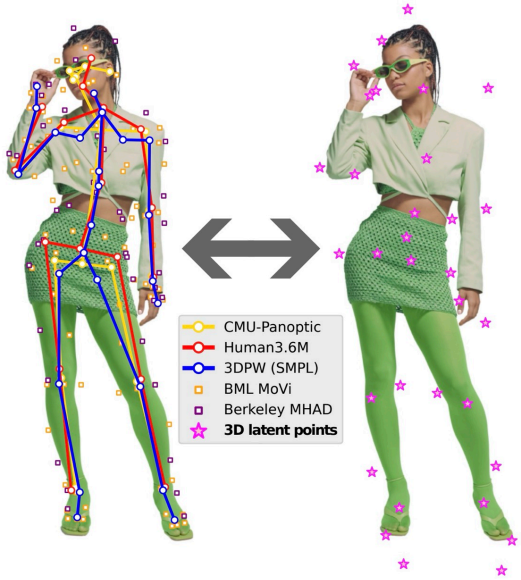


Figure 12: Different 3D human pose datasets (e.g., CMUPanoptic [COCO] and Human3.6M) provide annotations for different sets of body landmarks (From: Sárándi et al. (2022))

5.4 Phase 2: The Generalized Pipeline Architecture

To achieve input flexibility and robust processing, I designed a modular pipeline architecture implemented entirely in JavaScript. Each stage addresses a specific challenge in the path from raw key points to final avatar animation.

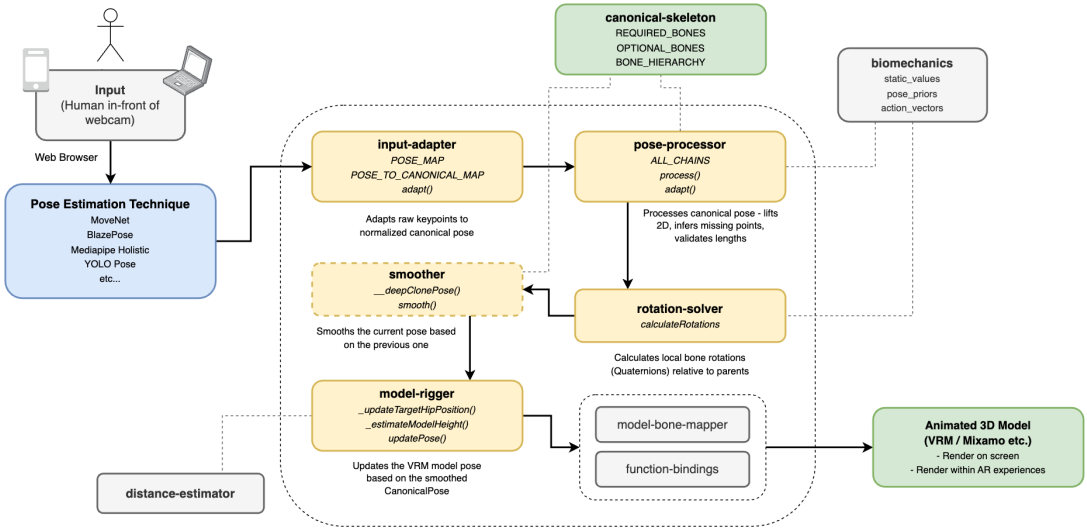


Figure 13: High-level diagram of the overall architecture and detailed pipeline

The key design principles were:

- **Modularity:** Each major processing step (adaptation, processing, solving, smoothing, rigging) is encapsulated in its own class/module, allowing for independent development, testing, and future replacement or enhancement (e.g., adding an IK module later).
- **Standardized Intermediate Representation:** The *CanonicalPose* structure serves as the common data format passed between modules, decoupling them.
- **Root-Relative Processing:** All internal processing (lifting, inference, rotation solving) operates in a consistent normalized, root-relative coordinate space (with the 'hips' bone at the origin), simplifying calculations and making them independent of the user's absolute position or scale in the input image. The architecture was designed from the ground up with dual generalisation in mind: flexibility in accepting various pose estimation inputs at the front-end, and adaptability in retargeting the processed motion to different standard humanoid avatar formats (primarily VRM, with extensibility demonstrated for others) at the back-end.
- **Efficiency:** Algorithms were chosen and implemented with browser performance constraints in mind, favouring efficient vector math and heuristics over computationally heavy operations where possible.

5.5 Phase 3: Input Adaptation and Standardisation

This module addresses the challenge of input heterogeneity. Its primary role is to translate the diverse outputs from the selected pose estimators into the standardized *CanonicalPose* format, specifically populating the raw landmarks map and providing initial estimates for the bones map.

- **Handling Source Diversity:** The adapt method uses a switch statement based on the sourceType ('MoveNet', 'YOLO', 'BlazePose', 'Holistic').
- **Keypoint Mapping:** Internal maps (*COCO MAP*, *BLAZEPOSE MAP*) translate the source keypoint indices to descriptive names (e.g., 'leftShoulder'). These names are used as keys in the landmarks map.
- **Coordinate Unification and Normalisation:** This was a critical implementation step requiring careful handling:
- **Pixel Inputs (MoveNet, YOLO, potentially BlazePose 2D/3D)**: Calculates a root position (midpoint of high-confidence hips, fallback considered) in pixels. Calculates a reference scale factor (e.g., using videoHeight or pixel shoulder width). Centres all key points relative to the pixel root and divides by the scale factor, inverting the Y-axis (normY $= -\frac{\text{centeredY}}{\text{scale}}$), to map into the internal normalized space where the hip is effectively (0,0,0). The Z coordinate is initialized to 0.
- **Holistic Normalized Input *(poseLandmarks)*:** Coordinates are already [0,1]. Finds the hip midpoint in this space. Centers other landmarks relative to this hip midpoint. Applies Y-axis inversion. Scales the result based on a reference size (e.g., normalized shoulder width) to match the scale of the internal normalized space derived from pixel inputs. The Z coordinate is often unreliable and might be scaled or initially ignored.

- **Holistic World Input *(poseWorldLandmarks)*:** This data is already root-relative and in pseudo-meters. The primary step is axis mapping (e.g., potentially swapping/negating Y and Z) to align with the internal Y-Up, Z-Forward convention. Depending on the chosen internal unit system (purely relative vs. meters-relative), it might require scaling (e.g., dividing all coordinates by the measured shoulder width in meters to get a consistent relative scale).
- **Populating *CanonicalPose*:** For each detected landmark, it stores the raw position (Vector2/Vector3 in source space) and confidence in *canonicalpose -> landmarks*. It then attempts to find the corresponding canonical bone name (e.g., 'leftWrist' landmark maps to 'leftsHand' bone) using *'TO CANONICAL'* maps and populates the *canonicalpose -> bones* entry with the calculated normalized, root-relative position, confidence, and flags (is3D, isEstimated=false).
- **Root Handling:** Explicitly sets the 'hips' bone position to (0,0,0) in the normalized space and stores its confidence.

This adapter ensures that the *PoseProcessor* receives a consistently formatted input, regardless of the upstream detector's keypoint count, coordinate system, or dimensionality (2D/3D). This inherent flexibility allows the user or application to select the most suitable pose estimator based on available resources, a core design goal differentiating this work from single-source pipelines.

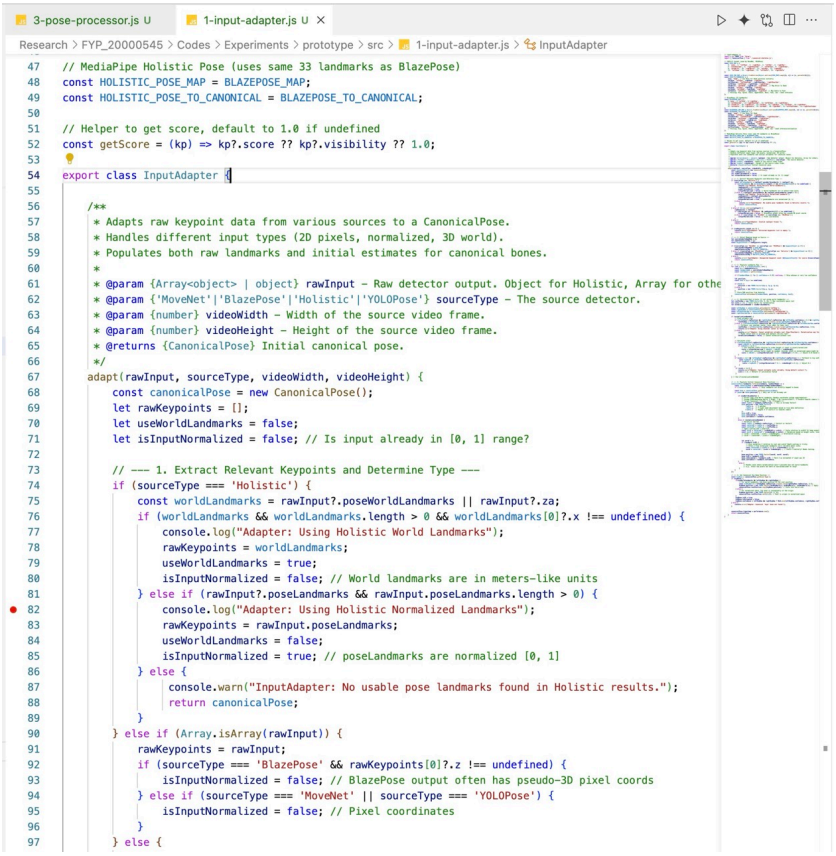


Figure 14: Implementation of input-adapter to work with selected pose estimation techniques

5.6 Phase 4: Pose Processing - Reconstruction and Biomechanical Refinement

This module forms the core intelligence of the pipeline, taking the potentially sparse or 2D normalized pose from the adapter and generating a complete, validated, and plausible 3D pose in the same normalized, root-relative space. A key strategy employed here, particularly for handling sparse (e.g., 17-point COCO) or 2D inputs, is the use of **pose priors**. These priors represent learned statistical patterns and correlations derived from large datasets of human motion (in our case, H36M, detailed in Phase 5), effectively creating a **"statistical skeleton" or "pose prior"**. They encapsulate knowledge about typical bone lengths, proportions, joint relationships, and common postures, providing essential constraints and guidance for reconstructing a plausible 3D pose from incomplete information.

5.6.1 Heuristic 2D-to-3D Lifting

When the input lacks depth (`!isPose3D(pose)`), this algorithm estimates the normalized Z coordinate.

- **Method:** Uses a Breadth-First Search (BFS) traversal starting from the *hips* (guaranteed to be 3D with $z = 0$). For each child bone with only 2D data, it calculates the target normalized bone length (from loaded priors). It solves for:

  $$\Delta Z^2_{\text{norm}} = \|\mathbf{b}_{\text{norm}}\|^2 - \|\mathbf{b}_{2D}\|^2$$

  where $\|\mathbf{b}_{\text{norm}}\|$ is the prior normalized 3D bone length, and $\|\mathbf{b}_{2D}\|$ is the current 2D distance between the parent and child joints.

- **Disambiguation and Robustness:** If $\Delta Z^2_{\text{norm}} < 0$ (i.e., 2D distance exceeds prior length), then the Z difference is clamped to 0, treating the bone as parallel to the viewing plane. Otherwise, the ambiguity in sign from $\pm\sqrt{\Delta Z^2_{\text{norm}}}$ is resolved primarily using temporal consistency: the solution closer to the previous pose's Z coordinate is chosen. This approach proved significantly more stable than using static heuristics alone. The chosen ΔZ_{norm} is added to the parent's Z to estimate the child joint's Z.

  Limitation: This remains an approximation, its accuracy depends on the quality of the prior data and temporal stability. Unusual poses or rapid depth changes can lead to errors.

5.6.2 Missing 3D Joint Inference

This addresses missing data points after the initial 3D structure (lifted or direct) is established.

When the input pose lacks certain joints (due to detector limitations or occlusion), this crucial step estimates their 3D positions within the normalized, root-relative space. It uses an iterative BFS approach (traversing the skeleton graph multiple times up to maxInferenceIterations) to allow inferred positions to inform subsequent inferences. For a missing bone position, it attempts the following strategies in a prioritized order, leveraging the pre-calculated priors (see Section 5.7).

- **Kinematic Chain Estimation using Priors:** If the parent joint is known, this combines kinematic constraints with priors. It uses the known ParentPosition, the prior *AverageBoneLength* for the current segment, and potentially angular priors (like average bend angles or relative vectors from Phase 5) to estimate the missing joint's position. For instance, estimating a missing LWrist given LElbow involves using LowerArmLength prior and potentially an average elbow bend angle prior or the Average Relative Vector (*V_avg_Elbow_Wrist)* rotated according to the upper arm's current orientation.

- **Average Relative Vector Prior:** Retrieves the pre-calculated average normalized 3D vector from the parent to the current bone (using *getActionAverageRelativeVector* or *getGlobalAverageRelativeVector*). This vector, potentially rotated by the parent's estimated orientation, is added to the parent's position. This provides a strong, data-driven initial guess, especially useful for joints like the neck relative to the spine.

- **Temporal Information (Previous Frame):** If the joint was visible in the previous frame (*previousPose*), its last known position is used as a fallback. A simple velocity estimate ($P_{\text{current}} = P_{\text{last}} + \text{Velocity} \times \Delta t$) could also be incorporated to predict the position, potentially projecting it onto the valid sphere/circle defined by the parent and bone length prior. However I could not complete this fully according to the plan with the constraints.

- **Symmetry:** If the contralateral joint (e.g., RWrist if LWrist is missing) is available and confident, and the pose appears relatively symmetrical (a simple check based on hip/shoulder alignment), its position is mirrored across the sagittal plane (X=0 in our root-relative space) to estimate the missing joint's position. This relies on the powerful prior of bilateral symmetry.

Inferred positions are assigned a moderate confidence score (*inferredConfidence*), influencing how they might blend with existing low-confidence detections or be used in subsequent inferences. This multi-stage, prior-informed inference allows the pipeline to reconstruct a reasonably complete skeleton even from sparse inputs like COCO-17, enabling its generalisation capabilities.

5.6.3 Biomechanical Validation

- **Bone Length Enforcement:** After inference, it iterates through connections again via BFS. It calculates the current distance between connected joints in the normalized space and compares it to the average normalized bone length prior using *validateNormalizedBoneLength*. If the deviation exceeds tolerance (e.g., 20%), it corrects the child's position by moving it along the parent-child vector to match the target normalized length. This prevents unnatural stretching/compression artifacts. However this process has to be optimized to reduce the extra traversal after inference.

- Joint Limits: While Euler limits based on NASA-STD (*Human Spaceflight and Aviation Standards - NASA* (2023))/Winter's (Thomas et al. (2023)) data were defined in *biomechanics.js*, robust enforcement requires operating on local rotations. Therefore, direct clamping was deferred to the *RotationSolver* or *VRMRigger* stage as a conceptual step, acknowledging the complexities involved.

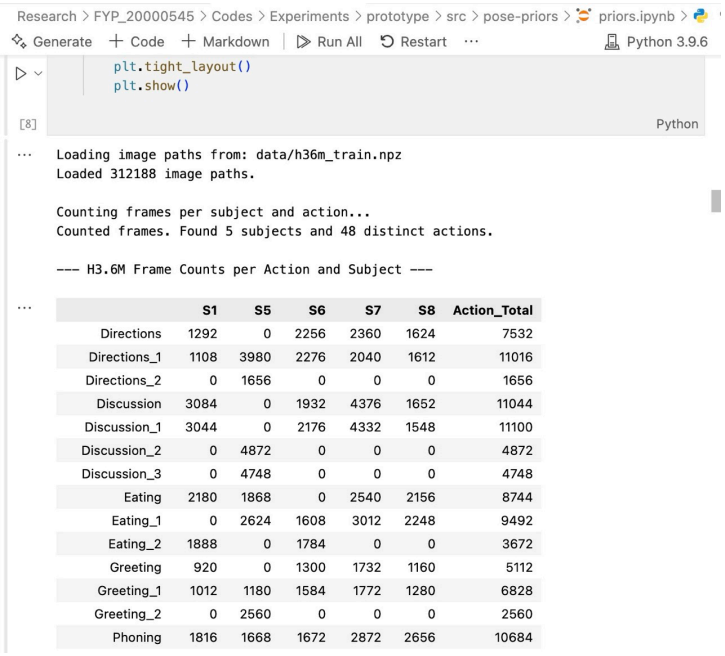
5.7 Phase 5: Offline Prior Generation and Analysis

The effectiveness of the real-time inference and validation steps relies heavily on data-driven biomechanical priors. In addition to studying about already defined static values, I tried to do a exploration on my own by analyzing a dataset of actual motion-capture data. Extensive offline analysis was performed on the Human3.6M (H3.6M) dataset (Ionescu et al. (2014)) using Python (NumPy, SciPy, Matplotlib) within Jupyter Notebooks. H36M was chosen over datasets like COCO ("in the wild") because it provides accurate 3D ground truth motion capture data from controlled scenarios, making it suitable for extracting reliable biomechanical statistics.

Data Used: Pre-processed H36M .npz files containing root-relative 3D keypoints (keypoints3d) and metadata including subject IDs and action labels (e.g., 'Walking', 'Sitting', 'Phoning'). The standard 17-joint H36M skeleton mapping was primarily used, which shares similarities with COCO but includes distinct hip and spine points.

Coordinate System Verification: Confirmed the 3D coordinate system (e.g., Y-Up or Z-Up depending on the specific data processing) and ensured consistent mapping to our internal canonical bone names. All calculations were performed in a root-relative frame, treating the hip midpoint as the origin (0,0,0). Preliminary analysis included estimating approximate subject heights based on standing/walking poses to understand scale variations, although the primary priors used internally were scale-normalized.

- **Normalized Bone Lengths:** Poses were scale-normalized (e.g., scaling such that average torso length = 1.0 across the dataset). The mean and standard deviation of the Euclidean distance between connected joints were calculated for each bone segment (e.g., 'leftUpperArm', 'rightLowerLeg'). These are fundamental for heuristic 3D lifting and kinematic chain inference.
- **Body Proportions:** Ratios between key average bone lengths (e.g., arm-to-leg, torso-to-height) were computed as a sanity check and potential scaling reference.
- **Average Relative Joint Vectors:** The mean 3D vector from each parent joint to its child joint was calculated in the normalized, root-relative space. This captures both average length and typical resting orientation. Priors were computed globally (averaged over all actions) and also **per-action** (e.g., the average 'Hip' to 'Neck' vector specifically for 'Sitting' poses). These action-specific priors capture characteristic postural differences.



Research > FYP_20000545 > Codes > Experiments > prototype > src > pose-priors > priors.ipynb >

Generate | Code | Markdown | Run All | Restart | ... | Python 3.9.6

```
    plt.tight_layout()
    plt.show()
```

[8] Python

```
Loading image paths from: data/h36m_train.npz
Loaded 312188 image paths.

Counting frames per subject and action...
Counted frames. Found 5 subjects and 48 distinct actions.

--- H3.6M Frame Counts per Action and Subject ---
```

	S1	S5	S6	S7	S8	Action_Total
Directions	1292	0	2256	2360	1624	7532
Directions_1	1108	3980	2276	2040	1612	11016
Directions_2	0	1656	0	0	0	1656
Discussion	3084	0	1932	4376	1652	11044
Discussion_1	3044	0	2176	4332	1548	11100
Discussion_2	0	4872	0	0	0	4872
Discussion_3	0	4748	0	0	0	4748
Eating	2180	1868	0	2540	2156	8744
Eating_1	0	2624	1608	3012	2248	9492
Eating_2	1888	0	1784	0	0	3672
Greeting	920	0	1300	1732	1160	5112
Greeting_1	1012	1180	1584	1772	1280	6828
Greeting_2	0	2560	0	0	0	2560
Phoning	1816	1668	1672	2872	2656	10684

Figure 15: Stats of data in H36M train data

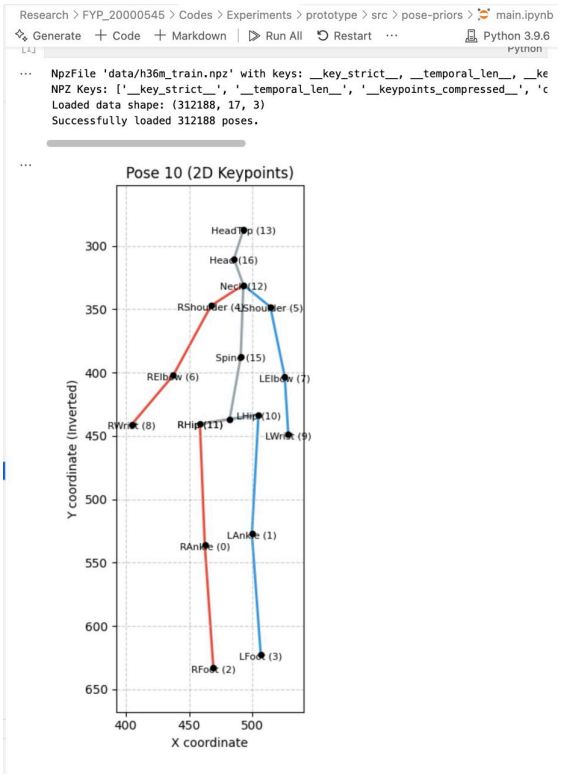


Figure 16: Visualisation of 2D pose data in a sample image (Pre-processed)

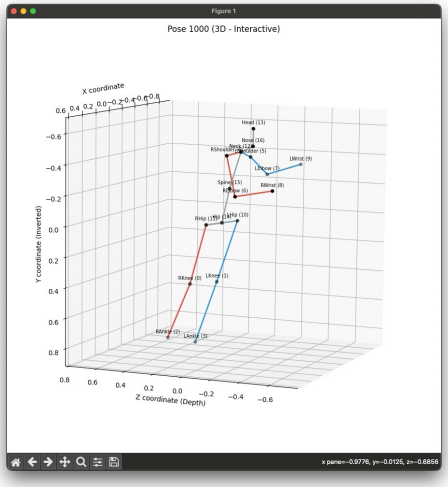


Figure 17: Visualisation of 3D pose data in a sample image (Pre-processed)

5.7.1 Positional Priors Calculated:

- **Rotational Priors and Coordination Patterns (Exploratory Analysis and Heuristic Usage):** While direct application of complex rotational constraints and making use of all this information (specially action-specific) proved challenging for real-time JS, understanding typical coordination patterns was crucial for guiding heuristic development and identifying future work. Analysis included:

  - **Yaw, Pitch, Roll Definition:** Joint orientation components were analyzed relative to parent segments or the world frame. Yaw (turn) was typically calculated via yaw $= \arctan 2(x, z)$ around the Y-axis, Pitch (nod) via pitch $= \arctan 2(-y, \sqrt{x^2 + z^2})$ relative to the XZ plane, and Roll (tilt) derived from full orientation matrices or quaternions where available.
  - **Spine-Torso Coordination:** Analyzed the correlation between torso twist (e.g., hip-shoulder yaw difference) and spine joint rotations, aiming to prevent rigid "block" torsos. This informed the need for potential spine interpolation logic.
  - **Neck-Head Coordination:** Examined typical head yaw/pitch relative to the neck orientation, important for gaze direction.
  - **Shoulder Girdle / Clavicle Behavior:** Investigated subtle shoulder joint adjustments correlated with arm elevation and forward/backward movement (protraction/retraction).
  - **Elbow/Wrist Natural Angles:** Observed typical forearm roll accompanying elbow flexion and relaxed wrist flexion/deviation.
  - **Action-Specific Rotational Patterns:** Noted distinct joint angle distributions for actions like Sitting (e.g., typical knee flexion around 90°), Walking (arm swing correlated with leg stride), and Phoning (characteristic arm posture). While not implemented as dynamically switching priors in the final version (see Section **??**), this analysis highlighted the potential benefits.

- **Output:** All calculated priors (mean/std dev of normalized lengths, global and action-specific average relative vectors, key joint angle statistics) were saved into structured JSON files:
  (e.g.: *h36m_positional_priors.json*, *h36m_action_avg_vectors.json*). These files are loaded by the biomechanics.js module in the JavaScript pipeline for efficient access during real-time processing.

  This extensive offline analysis forms the data-driven foundation enabling the Pose Processor (Phase 4) to realistically infer missing joints and validate pose geometry even when provided with sparse or 2D input.

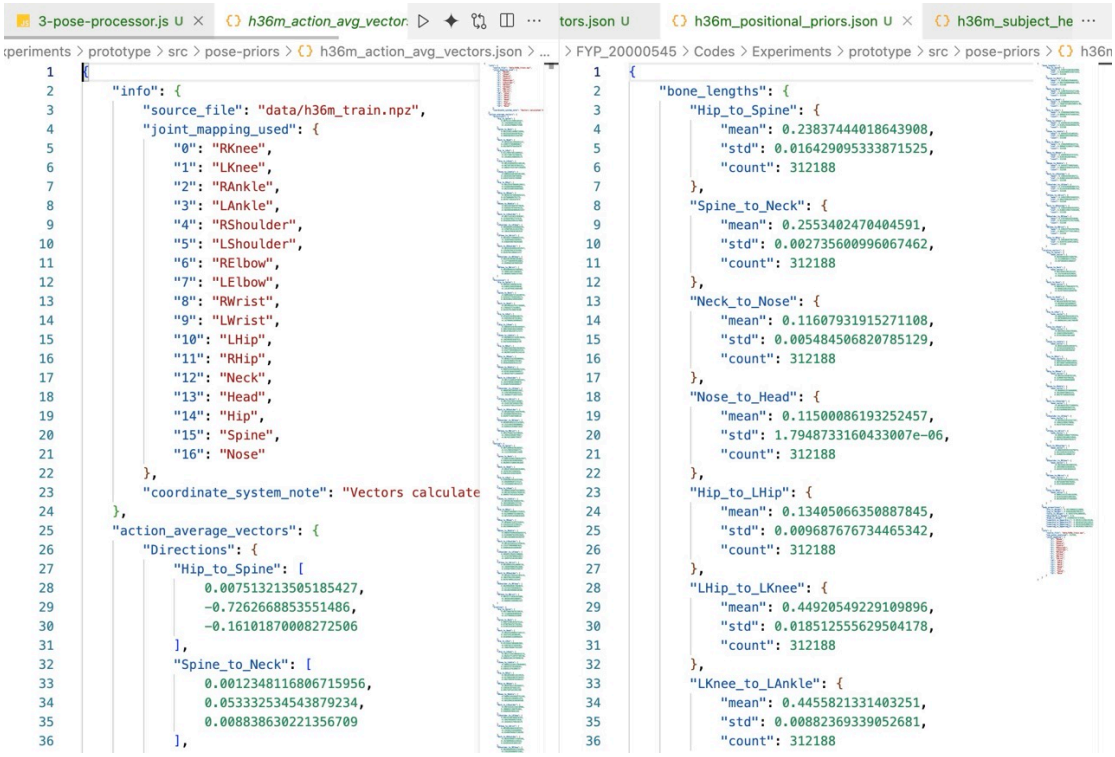


Figure 18: JSON files with calculated prior information

- Body Proportions: Ratios between key average bone lengths were computed.
- Rotational Priors (Exploratory): Analyzed correlations like Spine Yaw vs. Torso Twist using linear regression.

This offline analysis forms the data-driven foundation for the real-time inference and validation components.

5.8 Phase 6: Rotation Solving, Smoothing, and VRM Retargeting

5.8.1 Rotation Calculation

This module converts the final processed normalized 3D positions into the necessary joint rotations for animation.

- Input: Validated, normalized *CanonicalPose* with 3D positions relative to the hip origin.
- Output: Populates the rotation property (a THREE.Quaternion) for each *CanonicalBone* with its local rotation relative to its parent.
- Method: Employs a two-stage BFS approach. First, it estimates the world orientation of each bone based on the vector to its child using *lookAt* (with careful handling of the 'up' vector and axis correction). Second, it converts these world

quaternions into local quaternions using the parent's inverse world rotation. This ensures correct hierarchical transformations. Leaf nodes inherit orientation appropriately.

5.8.2 Temporal Smoothing (Smoother)

Applies an Exponential Moving Average (EMA) filter to introduce temporal coherence and reduce visual jitter.

- Input: *CanonicalPose* with calculated local rotations and positions.
- Output: Smoothed *CanonicalPose*.
- Method: Uses *Quaternion.slerp* for rotations and *Vector3.lerp* for the hip position between the current calculated pose and the previous smoothed pose, controlled by adjustable alpha factors. However this step required more optimisation to run reliably.

5.8.3 VRM Retargeting Rendering (ModelRigger)

This final stage maps the processed canonical pose onto the loaded THREE.js VRM avatar and handles world placement.

- Input: Smoothed, normalized *CanonicalPose* (local rotations, root at 0,0,0), plus the separately estimated cameraDistance, however the camera distance implementation could not be completed successfully during initial prototype implementation.
- Distance-Based Positioning: Calculates the target absolute world position for the VRM 'hips' bone based on the estimatedCameraDistance (placing it along the Z-axis) and an estimated floor height (derived from the loaded vrm.modelHeight).
- Scaling (Deferred): Calculates a potential absoluteScale factor based on comparing apparent pixel size to priors/calibration (as detailed previously), but currently applies a fixed scale (1.0) to the VRM root node (vrm.scene.scale), letting camera perspective handle apparent size changes. Dynamic scaling was not attempted currently due to complexity and potential visual artifacts.
- Rotation Application: Maps canonical bone names to the cached THREE.js Bone nodes from the VRM humanoid structure. Applies the final smoothed local rotations using Quaternion.slerp (allowing for optional additional smoothing here or even skip previous smoothing step entirely).
- Joint Limit Enforcement: Further robust joint limit clamping (e.g., swing-twist or carefully applied Euler clamping via clampRotation) can be applied to the local quaternions before they are set on the VRM bones, however this should be done carefully by checking with the output.
- Rendering: The main loop calls *THREE.WebGLRenderer.render()* and importantly *vrm.update(deltaTime)* to update VRM-specific features like spring bones after the rigger has updated the core skeleton transforms.

- Format generalisation: The pipeline's output (normalized pose with standard VRM bone names and local rotations and some extra joints as well) is inherently adaptable. While VRM is the primary target, adding support for other formats like Mixamo-rigged FBX or standard glTF humanoids would primarily involve creating a new output mapping configuration within the *VRMRigger* to match different bone names and potentially adjust for different rest poses or axis orientations, demonstrating the flexibility of the canonical skeleton approach.

- **Format generalisation and Novelty:** While primarily targeting the well-defined VRM 1.0 standard due to its humanoid focus and growing adoption, a key goal was output format flexibility. The pipeline's output – a normalized*CanonicalPose* (with several extra keypoints as well) with standard humanoid bone names and local rotations – is inherently adaptable. To demonstrate this, I successfully developed and tested a proof-of-concept mapping configuration for humanoid rigs originating from Adobe's Mixamo service. Mixamo rigs often present challenges due to their different bone naming conventions (e.g: mixamorig:Hips, mixamorig:Spine1) and sometimes slightly different hierarchy or rest pose compared to VRM.

Mixamo *Mixamo* (n.d.) is a widely used online platform offering a vast library (thousands) of pre-captured, professional full-body character animations and a significant collection of downloadable 3D character models, often provided in FBX or glTF formats. Crucially, Mixamo also provides an **automatic rigging service ('Auto-Rigger')** where users can upload their own custom 3D character models (e.g., in '.obj', '.fbx' format) and have Mixamo automatically generate a standardized humanoid skeleton rig fitted to their mesh. This auto-generated rig, while consistent across Mixamo, uses its own specific bone naming convention (e.g., 'mixamorig:Hips', 'mixamorig:Spine1', 'mixamorig:LeftArm') and typically includes around 65 bones.

My Mixamo mapping proof-of-concept involved:

- Manually analyzing the typical Mixamo skeleton structure and identifying the correspondence between the canonical VRM bone names used internally by my pipeline (like 'hips', 'spine', 'leftUpperArm') and the common Mixamo bone names (like 'mixamorig:Hips', 'mixamorig:Spine', 'mixamorig:LeftArm').

- Creating a specific mapping function within the *'VRMRigger'* to retrieve the correct THREE.js 'Bone' object from the loaded Mixamo model based on its name.

- Applying the calculated local rotations from the 'CanonicalPose' to these mapped Mixamo bones. For this proof-of-concept, I focused on mapping the core body and limb rotations. Mapping to the more detailed spine segments or individual finger bones present in the Mixamo rig was identified as a straightforward extension but not fully implemented due to time constraints.

- Addressing potential minor discrepancies in rest poses (T-pose vs. A-pose) or coordinate axis orientations between the VRM-centric canonical pose and the Mixamo rig through potential small, corrective rotation offsets during application (though often the core rotations transfer reasonably well).

This successful demonstration of mapping the same processed canonical pose to both standard VRM models and common Mixamo-rigged models shows the **novelty and**

utility of the developed middleware pipeline. It validates the canonical skeleton as an effective intermediate representation for achieving output format generalisation.

Limitation: Fully automating the mapping to arbitrary, non-standard glTF/FBX rigs remains a complex challenge requiring sophisticated analysis of bone names, hierarchy, and potentially user intervention, which was beyond the scope of this implementation. The focus remained on demonstrating adaptability across common standards like VRM and Mixamo.

```
{
    "name": "Vanguard",
    "type": "fbx",
    "path": "./Vanguard.fbx",
    "binding": {
        "Hips": {
            "name": "mixamorigHips",
            "order": "XYZ",
            "func": { "fx": "-x", "fy": "y", "fz": "-z" }
        },
        "Neck": {
            "name": "mixamorigNeck",
            "order": "XYZ",
            "func": { "fx": "-x", "fy": "y", "fz": "-z" }
        },
        "Chest": {
            "name": "mixamorigSpine2",
            "order": "XYZ",
            "func": { "fx": "-x", "fy": "y", "fz": "-z" }
        },
        "Spine": {
            "name": "mixamorigSpine",
            "order": "XYZ",
            "func": { "fx": "-x", "fy": "y", "fz": "-z" }
        },
        "RightUpperArm": {
            "name": "mixamorigRightArm",
            "order": "ZXY",
            "func": { "fx": "-z", "fy": "x", "fz": "-y" }
        },
```

Figure 19: Mapping data between Mixamo model joints to CanonicalPose joints

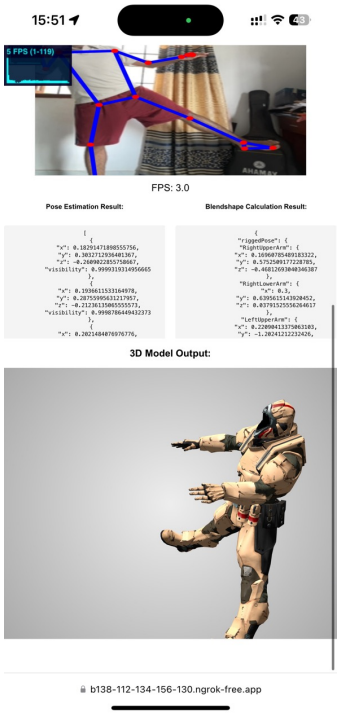


Figure 20: Mixamo model demo in prototype application

5.9 Summary of the Chapter

This chapter detailed the comprehensive methodology employed in this research, tracing its evolution from preliminary explorations to the design and specification of the final generalized architecture for real-time avatar animation. I began by outlining the initial experimental phase, which involved surveying and testing various pose estimation techniques and exploring 3D model formats and web rendering engines. This phase identified the significant challenges posed by input heterogeneity and the limitations of existing direct-mapping solutions, motivating a pivot towards a more flexible, generalized approach.

The core of the methodology focused on the design and specification of a modular JavaScript pipeline. This included establishing the development environment and

detailing each crucial stage (Input Adaptation, Pose Processing, Prior Generation and Usage, Rotation Solving and Smoothing, VRM Retargeting).

The chapter elaborated on the specific algorithms, data structures (like the *CanonicalPose*), handling of key challenges (like 2D->3D ambiguity and missing data), and the rationale behind implementation choices (like prioritizing heuristic methods over complex ML or IK solutions due to real-time web constraints). This detailed methodology provides the blueprint for the system whose implementation and results are presented in the following chapter.

6 Results and Analysis

This section presents the results obtained from the implementation and testing of the generalized real-time avatar animation pipeline developed in this research. The analysis focuses on addressing the core research questions concerning suitable real-time pose estimation techniques (RQ1), the research and development of the canonical pose transformation model and generalized pipeline (RQ2), and the resulting performance, realism, and user experience (RQ3) within the target web browser environment.

Due to significant time constraints and unforeseen technical challenges encountered during the development phase (particularly concerning cross-platform WebXR compatibility and model conversion intricacies), I could not fully execute a comprehensive quantitative and qualitative evaluation, especially within immersive AR scenarios, as planned. However, extensive testing of the core pipeline components within the prototype web application yielded valuable performance metrics and qualitative observations, which are presented and analyzed below.

6.1 RQ1 Analysis: Real-Time Pose Estimation in Constrained Environments

The goal of RQ1 was to identify optimal algorithms for real-time human pose capture in resource-constrained environments. The preliminary experiments (detailed above) involved integrating and evaluating four distinct techniques within the JavaScript prototype: MoveNet (TF.js, 17 2D pts), BlazePose (MediaPipe/TF.js, 33 2D/3D pts), Holistic (MediaPipe, 543 2D/3D landmarks), and YOLOv8-Pose (TF.js via conversion, 17 2D pts).

Performance evaluation focused on Frames Per Second (FPS) achieved on representative hardware. While more extensive benchmarking across various devices was planned, testing was conducted on available hardware: a MacBook running a Chromium based browser and an iPhone running Safari. Average FPS results provide insight into the real-time feasibility of these models within a typical browser tab running the webcam feed, pose estimation and inference pipeline, and the basic THREE.js rendering setup.

Table 3: Approximate Average FPS for Pose Estimation Models in Browser (Single Subject)

Pose Estimation Model	**MacBook Pro M1 Pro (Arc - Chromium)**	**iPhone 14 Pro (Safari iOS)**
MoveNet (Lightning)	40–60 FPS	20–30 FPS
BlazePose (Full)	20–30 FPS	10–25 FPS
MediaPipe Holistic	10–15 FPS	5–20 FPS
YOLOv8n-Pose (Converted TF.js)	15–20 FPS	5–20 FPS

Analysis:

- As expected, lightweight models like **MoveNet (Lightning)** generally offered the highest FPS, particularly demonstrating reasonable performance even on mobile (Table 3). Its limitation lies in providing only 17 sparse 2D keypoints.

- **BlazePose** provided a good balance, offering more keypoints (33) including some depth information, while maintaining FPS comparable to or slightly lower than MoveNet.
- **YOLOv8n-Pose**, despite being efficient in its native environment, showed moderate performance after conversion to TF.js, comparable to BlazePose but providing only 17 2D keypoints. Its primary strength in multi-person detection was not leveraged here. The model conversion process itself was a significant hurdle, indicating the challenges of using non-web-native models. Model sizes also vary significantly, impacting initial load times (e.g., YOLOv8n-Pose: 180KB vs. Holistic full: 6.4MB).
- **MediaPipe Holistic**, while providing the richest data (543 landmarks), exhibited the lowest average FPS, especially on mobile where thermal throttling noticeably impacted sustained performance. This highlights the direct trade-off between input detail and computational cost.
- It is important to note that these performance figures reflect the current prototype implementation. **Significant opportunities exist for optimisation**, including refining JavaScript code for better performance, implementing more rigorous memory management to reduce garbage collection pauses, potentially offloading specific computations to Web Workers, or exploring WebAssembly (Wasm) for critical code paths. Such optimisations could potentially improve FPS and reduce fluctuations across all models, particularly on mobile.

Conclusion for RQ1: There is no single "optimal" algorithm for all scenarios. The choice depends heavily on the target device's capabilities and the required level of detail. Lighter models like MoveNet or BlazePose are better suited for broad compatibility and lower-end devices, while Holistic offers maximum detail at the cost of performance. This finding strongly validated the research pivot towards a **generalized pipeline** capable of handling this spectrum of inputs, rather than relying on a single estimator. The pipeline developed allows users (or applications) to select the most appropriate estimator based on their specific resource constraints and fidelity needs. Future integration of even more efficient pose estimators will be easily supported by the modular *'InputAdapter'*.

6.2 RQ2 Analysis: The Generalized Transformation Model (Pipeline)

The second research question (RQ2) aimed to determine how a generalized transformation pipeline could be designed to robustly map diverse pose estimation outputs to a canonical representation, particularly when handling incomplete data. To evaluate this, the developed pipeline was tested against its core design goals: its ability to adapt to heterogeneous inputs, its effectiveness in reconstructing a complete 3D pose from sparse data, and its capacity to produce a standardized output suitable for animation.

Demonstration of Input generalisation The pipeline's primary functional requirement was its ability to adapt to multiple input sources. Testing confirmed that the InputAdapter module successfully processed data streams originating from all four integrated estimators: the sparse 17-point 2D skeletons from MoveNet and YOLO-Pose,

the 33-point 3D skeleton from BlazePose, and the dense 543-landmark data from MediaPipe Holistic. The system demonstrated its ability to unify these varied keypoint counts, coordinate systems, and topologies into the consistent, internal CanonicalPose structure, thereby validating its input generalisation capability.

Effectiveness of Pose Reconstruction and Inference A key measure of the pipeline's success is its ability to create a complete and plausible 3D pose even from incomplete data. This was evaluated through qualitative observation.

2D-to-3D Heuristic Lifting: When provided with 2D-only input from MoveNet or YOLO-Pose, the heuristic lifting algorithm in the PoseProcessor successfully generated poses with apparent depth. Visual inspection confirmed that by using bone-length priors and temporal consistency, the system avoided the "flat" appearance typical of naive 2D mappings and produced a reasonable 3D reconstruction. The primary limitation observed was in poses with significant limb foreshortening, where the estimated depth could be inaccurate, as illustrated in Figure **??**.

Missing Joint Inference: The system's ability to "fill in the blanks" was most evident with sparse 17-point inputs. As shown in Figure **??**, even when the input skeleton from YOLO-Pose lacked a spine, neck, or chest, the inference logic successfully estimated their positions using a combination of kinematic extrapolation from known joints (like the shoulders and hips) and the pre-calculated average vector priors. This allowed the system to drive a complete VRM skeleton, preventing the "broken model" or "stiff torso" effects that would otherwise occur. The main observed limitation is that this rule-based inference struggles with highly unusual poses or large-scale occlusions (e.g., a user sitting down completely out of frame from the waist down), where more contextual understanding is required. Biomechanical Validation: During testing with noisy input, the bone length validation step proved crucial. It successfully prevented anatomical artifacts, such as limbs unnaturally stretching or compressing, by correcting joint positions to adhere to the biomechanical priors (as shown in Figure **??**).

Demonstration of Output Adaptability Finally, the pipeline's output was evaluated for its utility and flexibility. The RotationSolver and VRMRigger modules consistently converted the final CanonicalPose into local joint rotations and applied them to a standard VRM avatar. To further test the generalisation of the canonical output, a proof-of-concept mapping was created for a non-VRM model rigged using Adobe's Mixamo service. As demonstrated in Figure 20, the pipeline was able to successfully drive the Mixamo model, validating that the standardized canonical pose serves as an effective intermediate representation for retargeting to different humanoid rigs.

Conclusion for RQ2: The experiments confirm that the developed pipeline successfully functions as a generalized transformation model. Its key design features—the input adapter, the prior-driven inference, and the canonical output representation—effectively solve the challenge of mapping diverse and imperfect pose inputs to a standard animatable avatar.

6.3 RQ3 Analysis: Realism, Performance, and User Experience

The third research question (RQ3) assessed the overall quality of the end-to-end system, focusing on the achievable realism, real-time performance, and user experience within the

target web browser environment. The evaluation was conducted through performance benchmarks and qualitative analysis of the final animated output.

Quantitative Performance Analysis The real-time performance of the full system (pose estimation + pipeline execution + rendering) was benchmarked. The primary bottleneck identified was the pose estimation model itself, as detailed in the RQ1 analysis (Table 3). To measure the overhead of the custom pipeline, additional profiling was conducted. The results showed that on the desktop test machine, the JavaScript pipeline (Adapter, Processor, Solver, and Rigger) added a manageable overhead, typically processing a frame in under 20ms. This indicates that on capable hardware, the pipeline itself does not prevent real-time operation. However, the combined computational load is significant, especially for mobile devices. While the system remains interactive on mobile when using a lightweight estimator like MoveNet, performance drops noticeably when using a heavyweight model like Holistic, confirming the critical importance of allowing users to choose an appropriate performance-quality trade-off.

Qualitative Analysis of Realism and Plausibility The visual quality of the animation was assessed under different conditions: Animation from Dense Input: When using detailed input from BlazePose or Holistic, the resulting avatar animation for common upper-body motions (waving, gesturing) was observed to be smooth and visually plausible. The quality was comparable to direct-mapping solutions, with the added benefit of the pipeline's validation steps preventing occasional anatomical errors. Animation from Sparse Input: The true value of the pipeline was most apparent when driven by sparse 2D input from MoveNet or YOLO. In these cases, the system produced a coherent, full-body animation. While lacking the fine detail of the dense input, the core posture and limb movements were correctly represented, successfully preventing the "frozen lower body" or "gliding torso" effect common with naive mappings. This represents a significant improvement in perceived realism for sparse-input scenarios. Observed Artifacts: The primary visual artifacts noted were related to the inherent limitations of monocular vision. As seen in Figure **??**, the heuristic depth estimation can sometimes result in incorrect limb placement in 3D space (e.g., a hand appearing in front of the head when it should be touching it). Additionally, the temporal smoothing, while effective at reducing jitter, could introduce a minor "lag" during very sudden, fast movements.

WebXR Integration and Identified Challenges I initiated experiments to integrate the avatar animation into a WebXR AR session using THREE.js. The goal was to overlay the animated avatar onto the real-world camera feed, potentially anchored to the floor.

Challenges Encountered: Significant difficulties arose with WebXR API inconsistencies and limitations across platforms. Specifically, obtaining reliable plane detection or using the 'local-floor' reference space (essential for placing the avatar correctly on the real floor) failed consistently on desktop browsers (using WebXR API Emulator) and iOS Safari during testing, throwing 'NotSupportedError' exceptions (local-floor reference space is not supported). While basic AR session startup was achieved, reliable anchoring and interaction proved highly problematic within the available time. Debugging hit-testing and reticle placement also consumed considerable effort without reaching

a stable solution.

Conclusion: While the core avatar animation pipeline functions independently, robustly integrating it into a cross-platform WebXR AR experience requires overcoming significant, rapidly evolving browser API limitations and warrants further dedicated development effort. The pipeline itself is compatible, but the WebXR integration layer needs more work.

Conclusion for RQ3: The developed system demonstrates that plausible real-time avatar animation is achievable in a web browser using a single webcam. The pipeline's performance is viable on modern hardware, and its inference capabilities provide a significant realism boost for sparse-input scenarios. The main limitations identified are tied to the fundamental challenges of monocular 3D reconstruction and the current maturity of WebXR browser APIs.

6.4 Summary of the Chapter

The results demonstrate the successful development of a generalized JavaScript pipeline for real-time avatar animation. RQ1 was addressed by evaluating multiple pose estimators and confirming the need for flexibility, with MoveNet/BlazePose offering performance advantages and Holistic providing maximum detail. RQ2 was addressed by the design and implementation of the modular pipeline featuring input adaptation, heuristic 3D lifting, prior-based inference, and rotation solving, proving its capability to process diverse inputs into a standard canonical representation. RQ3 was partially addressed, showing plausible real-time animation is achievable, especially showcasing the system's ability to generate full-body motion from sparse input. Performance benchmarks indicate viability, particularly on desktop, though mobile optimisation is needed. Limitations in the heuristic lifting accuracy, inference robustness for complex occlusion, lack of dynamic action context, and challenges in WebXR integration were identified through analysis and testing. Overall, the pipeline represents a novel, functional prototype demonstrating a viable approach to generalized, biomechanically-informed avatar animation in constrained web environments.

7 Discussion

This chapter moves beyond the presentation of results to an interpretation of their significance. The findings from the previous chapter are analysed here in the context of the initial research questions and the broader field of web-based immersive technologies. We will discuss the core contributions of this work, examine its implications, and frankly acknowledge its limitations, which in turn illuminate promising directions for future research.

7.1 Interpretation of Key Findings

The successful implementation and testing of the generalized pipeline provide answers to the core research questions posed at the outset of this thesis. The central hypothesis—that a generalized middleware could successfully drive real-time avatar animation from diverse inputs in a browser—was validated. In relation to RQ1, the performance benchmarks (Table 3) clearly demonstrated that there is no single "optimal" pose estimation algorithm for the web. Instead, a distinct trade-off exists between the detail of the input data and the computational performance. This finding confirms that a flexible system, capable of adapting to different input estimators, is not just a desirable feature but a necessary one for creating accessible applications that can run on a wide range of devices. In response to RQ2, this research presented a novel and effective design for a generalized transformation model. The modular pipeline, centered around the CanonicalPose representation, proved capable of handling the heterogeneity of modern pose estimators. The most significant finding was the success of the heuristic-based PoseProcessor. Its ability to perform 2D-to-3D lifting and infer missing joints using data-driven priors from the H36M dataset allowed the system to generate a complete, plausible 3D representation even from sparse 2D data. This demonstrates a viable, lightweight alternative to computationally expensive machine learning models for data completion, which is a key contribution of this work. The successful retargeting of the output to both standard VRM and proof-of-concept Mixamo models further validates the flexibility of this canonical approach. Finally, regarding RQ3, the results indicate that plausible, real-time avatar animation in a browser environment is indeed achievable. The visual quality of the animation was directly correlated with the quality of the input data, but the pipeline's inference mechanisms provided a significant and observable improvement in realism for sparse-input scenarios. While performance remains a challenge, particularly on mobile devices, the system's ability to support lightweight estimators like MoveNet makes real-time animation possible on hardware where more demanding, direct-mapping solutions would fail.

7.2 Key Contributions

This research makes several distinct contributions to the field of real-time, web-based graphics and human-computer interaction:

1. **A Novel Architecture for Input generalisation:** This work presents the design and implementation of what is, to my knowledge, the first purely browser-based middleware explicitly designed to handle multiple, heterogeneous real-time pose estimation inputs for avatar animation. This provides a blueprint for more flexible and accessible systems.

2. **A Lightweight Heuristic Inference System:** The development of a pose processing system that uses biomechanical priors and geometric heuristics for 2D-to-3D lifting and missing joint inference is a key contribution. It demonstrates a practical approach to achieving data robustness that is optimized for the resource-constrained execution environment of a web browser.

3. **A Practical End-to-End Web Pipeline:** This thesis delivers a complete, functional prototype that integrates all necessary components—from pose estimation to final rendering—using standard web technologies. It serves as a practical demonstration and a foundation upon which others can build.

7.3 Implications and Applications

The successful development of this generalized pipeline has several implications:

- **Accessibility:** By supporting lightweight pose estimators (MoveNet, YOLO) and running entirely in the browser, it lowers the barrier to entry for real-time 3D avatar experiences. Users do not require specialized hardware or high-end devices.
- **Flexibility:** Application developers can offer users a choice of pose estimation models, allowing them to balance performance and animation fidelity based on their device.
- **Web Platform Empowerment:** Demonstrates the increasing capability of the web platform (WebGL, WebAssembly, JavaScript engines) to handle computationally intensive tasks like real-time motion capture and 3D animation.
- **Foundation for Web Metaverse/Social VR:** Provides a core component necessary for enabling expressive, embodied interactions in web-based virtual worlds or telepresence applications without requiring app installations.
- **AR/VR Prototyping:** Offers a readily accessible tool for researchers and developers prototyping AR/VR interactions involving avatar representation.

Potential applications include virtual conferencing, online gaming, virtual try-on experiences, remote collaboration tools, educational simulations, and accessible VTubing setups.

7.4 Limitations of the Current Approach

A critical part of design science research is acknowledging the boundaries and limitations of the developed artifact. The choices made to prioritize real-time performance and flexibility in a web environment necessarily introduce specific trade-offs.

- **Accuracy of Heuristic Inference:** The core design of the PoseProcessor relies on heuristics and statistical priors rather than a deep learning model for pose completion. While this choice was essential for achieving real-time performance in JavaScript, it has limitations. The 2D-to-3D lifting is an approximation and can produce inaccurate depth estimates for complex or heavily foreshortened poses (as seen in Figure **??**). Similarly, the inference of missing joints is based on average human proportions and may not produce natural-looking results for highly unusual body types or unconventional poses.

- **Lack of Dynamic Action Context:** A key simplification in the current design is the use of global biomechanical priors. The system does not perform real-time action classification (e.g., detecting if a user is "sitting" vs. "standing"). As a result, its ability to infer the lower body is limited when the user is seated, as it lacks the context to assume the legs should be bent. This was a conscious design trade-off to keep the pipeline lightweight, but it limits the plausibility of inference in certain common scenarios.
- **Reliance on Forward Kinematics:** The pipeline currently uses Forward Kinematics (FK) to animate the avatar by setting each bone's rotation. While efficient, this means there is no mechanism to enforce certain constraints, such as ensuring a hand touches a specific point in space. An Inverse Kinematics (IK) solver was considered during the design phase but was deferred due to the significant performance and complexity challenges of implementing it efficiently in the browser. This limits the system's potential for more precise physical interactions.
- **Performance on Low-End Devices:** While the system is designed for resource-constrained environments, the combined load of running a pose estimator and the full JavaScript pipeline can still be too demanding for older or lower-end mobile devices, leading to low frame rates. The prototype has not undergone exhaustive performance optimization (e.g., using Web Workers or Wasm), which remains a crucial step for production-level deployment.
- **Scope of Evaluation:** Due to significant time constraints, the planned comprehensive evaluation could not be fully executed. The current results are based on performance benchmarks and qualitative developer observation. A formal user study to quantitatively measure perceived realism and user satisfaction is necessary to fully validate the user experience aspects of the system.

7.5 Summary of the Chapter

The discussion contextualized the research results, confirming the successful development of a generalized pipeline addressing the core research questions, albeit with certain limitations identified during implementation and testing. The key contributions lie in the system's input flexibility, its novel integration of biomechanical priors and heuristics for robust 3D pose reconstruction and inference from potentially sparse/2D data in real-time JavaScript, and its modular architecture centered around a canonical pose representation compatible with the VRM standard. While acknowledging limitations related to monocular depth ambiguity, distance/scale estimation, real-time action context, performance optimisation, and comprehensive evaluation, the research provides significant implications for making real-time avatar animation more accessible on the web platform. The identified limitations directly inform promising avenues for future research outlined in the concluding chapter.

8 Conclusion and Future Work

This thesis presented the research, design, implementation, and preliminary evaluation of a novel, generalized pipeline for real-time 3D humanoid avatar animation using standard webcams within resource-constrained browser environments. Motivated by the limitations of existing solutions often tied to specific high-fidelity inputs or specialized hardware, this work focused on creating an adaptive middleware capable of handling diverse pose estimation inputs, including sparse 2D keypoints, and generating plausible full-body motion.

8.1 Summary of Research

This thesis set out to address a critical gap in the accessibility of real-time 3D avatar animation. It identified the challenge that high-quality academic solutions are too resource-intensive for the web, while accessible web-based tools lack flexibility and robustness.

To solve this, this research successfully designed, developed, and evaluated a novel, generalized middleware pipeline that runs entirely in a web browser. The system's modular architecture is capable of ingesting and normalizing pose data from a variety of sources, from sparse 2D skeletons to dense 3D landmarks. Its core innovation lies in a lightweight processing module that uses data-driven biomechanical priors and heuristics to reconstruct a complete and plausible 3D pose, even from incomplete data.

The results demonstrate that this approach is viable, achieving plausible real-time animation on standard hardware. The pipeline effectively serves as a "universal adapter," making a wide range of pose estimation technologies more useful and robust for the purpose of avatar animation. In doing so, this work provides both a practical foundation and a conceptual blueprint for the future of accessible, web-based embodied interaction.

The core research questions were addressed through a multi-stage methodology. RQ1, concerning optimal real-time pose estimation techniques, was answered by demonstrating that no single method is universally optimal; the best choice involves a trade-off between detail and performance, validating the need for a generalized input pipeline. Several web-compatible estimators (MoveNet, BlazePose, Holistic, YOLO-Pose) were successfully integrated.

RQ2, focusing on the transformation model, was addressed by the design and implementation of the modular pipeline architecture centered around a *'CanonicalPose'* based on the VRM standard. Key components developed include:

- An *'InputAdapter'* for normalizing heterogeneous inputs into a consistent root-relative space.
- A *'PoseProcessor'* employing data-driven priors (derived from extensive offline H36M analysis) and biomechanical rules for heuristic 2D-to-3D lifting, missing joint inference (using average vectors, heuristics, kinematics), and bone length validation.
- A *'RotationSolver'* calculating standard local quaternion rotations.
- A *'VRMRigger'* applying the final processed pose to VRM avatars, incorporating relative distance estimation for world placement.

This pipeline successfully demonstrated its ability to transform diverse inputs into a standardized representation suitable for animation.

RQ3, concerning realism, performance, and user experience, was evaluated through prototype testing and performance measurements. The pipeline achieves plausible real-time animation, notably enabling full-body motion even from sparse 2D inputs, albeit with less detail than richer sources. Performance is viable on desktop/laptop hardware and usable on mobile with lighter estimators, although further optimisation is required. While formal user studies were limited by time, initial qualitative assessments were positive. The research successfully highlighted the trade-offs involved and the challenges remaining, particularly regarding depth accuracy, action context awareness, and seamless WebXR integration.

8.2 Future Work

The limitations identified in the discussion provide a clear and actionable roadmap for future research and development. The modular nature of the designed pipeline makes it well-suited for the following enhancements:

- **Integrate an Inverse Kinematics (IK) Solver:** To address the limitations of a purely FK-based system, an efficient, lightweight IK solver could be integrated. This would allow for more robust enforcement of bone length constraints and enable more complex interactions, such as ensuring the avatar's feet stay planted on the ground.
- **Implement Dynamic Action Classification:** To improve inference accuracy, a lightweight action classification model could be added to the pipeline. By detecting if a user is 'sitting' or 'walking', the system could dynamically switch to action-specific biomechanical priors, leading to much more plausible lower-body animations.
- **Explore Hybrid Inference Models:** The accuracy of the heuristic methods could be enhanced by exploring a hybrid approach. The current system could provide a fast initial guess, which is then refined by a very lightweight neural network specifically trained for tasks like depth estimation, if it can be run within the performance budget.
- **Full Performance Optimisation:** A dedicated phase of performance profiling could be undertaken to identify bottlenecks. Key components of the pipeline could then be rewritten in WebAssembly (Wasm) to achieve near-native execution speed, significantly improving performance on low-end mobile devices.
- **Conduct Formal User Experience (UX) Studies:** To rigorously validate the system, a formal user study should be conducted. This would involve gathering quantitative and qualitative feedback on the perceived realism, responsiveness, and overall satisfaction of the animation produced by the pipeline, especially when compared to a baseline direct-mapping approach.

This research provides a solid foundation, and these future directions offer exciting possibilities for further enhancing the accessibility, realism, and robustness of web-based real-time 3D avatar systems.

References

Al-Ansi, A. M., Jaboob, M., Garad, A. & Al-Ansi, A. (2023), 'Analyzing augmented reality (ar) and virtual reality (vr) recent development in education'.

Alexiadis, D. S., Chatzitofis, A., Zioulis, N., Zoidi, O., Louizis, G., Zarpalas, D. & Daras, P. (2017), 'An integrated platform for live 3d human reconstruction and motion capturing', *IEEE Transactions on Circuits and Systems for Video Technology* **27**, 798–813.

Alldieck, T., Magnor, M., Xu, W., Theobalt, C. & Pons-Moll, G. (2018), 'Video Based Reconstruction of 3D People Models'.
URL: *https://arxiv.org/abs/1803.04758*

Bi, W., Ma, Y., Tian, D., Yang, Q., Zhang, M. & Jing, X. (2023), Demystifying Mobile Extended Reality in Web Browsers: How Far Can We Go?, *in* 'Proceedings of the ACM Web Conference 2023', ACM, Austin TX USA, pp. 2960–2969.
URL: *https://dl.acm.org/doi/10.1145/3543507.3583329*

Bogo, F., Kanazawa, A., Lassner, C., Gehler, P., Romero, J. & Black, M. J. (2016), 'Keep it smpl: Automatic estimation of 3d human pose and shape from a single image'.
URL: *http://arxiv.org/abs/1607.08128*

Cao, Z., Simon, T., Wei, S.-E. & Sheikh, Y. (2016), 'Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields'.
URL: *https://arxiv.org/abs/1611.08050*

Caserman, P., Garcia-Agundez, A. & Gobel, S. (2020), 'A survey of full-body motion reconstruction in immersive virtual reality applications'.

Charles, S. (2016), Real-time human movement mapping to a virtual environment, *in* '2016 IEEE Region 10 Symposium (TENSYMP)', IEEE, Bali, Indonesia, pp. 150–154.
URL: *http://ieeexplore.ieee.org/document/7519395/*

Chen, Y., Tian, Y. & He, M. (2020), 'Monocular human pose estimation: A survey of deep learning-based methods', *Computer Vision and Image Understanding* **192**, 102897.
URL: *https://linkinghub.elsevier.com/retrieve/pii/S1077314219301778*

Choi, H., Moon, G., Chang, J. Y. & Lee, K. M. (2020), 'Beyond Static Features for Temporally Consistent 3D Human Pose and Shape from a Video'.
URL: *https://arxiv.org/abs/2011.08627*

Deng, H., Zhang, Q., Jin, H. & Kim, C. H. (2023), 'Real-time interaction for 3d pixel human in virtual environment', *Applied Sciences (Switzerland)* **13**.

Desmarais, Y., Mottet, D., Slangen, P. & Montesinos, P. (2020), 'A review of 3d human pose estimation algorithms for markerless motion capture'.
URL: *http://arxiv.org/abs/2010.06449*

Fang, H.-S., Li, J., Tang, H., Xu, C., Zhu, H., Xiu, Y., Li, Y.-L. & Lu, C. (2022), 'AlphaPose: Whole-Body Regional Multi-Person Pose Estimation and Tracking in Real-Time'.
URL: *https://arxiv.org/abs/2211.03375*

Güler, R. A., Neverova, N. & Kokkinos, I. (2018), 'DensePose: Dense Human Pose Estimation In The Wild'.
URL: *https://arxiv.org/abs/1802.00434*

HTC-Corporation (n.d.), 'VRM Avatar Maker, Download Free VRM Models | VIVERSE'.
URL: *https://www.viverse.com/avatar/what-is-vrm*

Huang, X., Twycross, J. & Wild, F. (n.d.), 'A process for the semi-automated generation of life-sized, interactive 3d character models for holographic projection'.

Human Spaceflight and Aviation Standards - NASA (2023).
URL: *https://www.nasa.gov/directorates/esdmd/hhp/human-spaceflight-and-aviation-standards/*

Husein, A. M. & Ciawi, J. (2021), 'Motion capture in humanoid model with unity engine using kinect v2', *Architecture and High Performance Computing* **3**.
URL: *https://doi.org/10.47709/cnahpc.v3i2.1067*

Ionescu, C., Papava, D., Olaru, V. & Sminchisescu, C. (2014), 'Human3.6M: Large Scale Datasets and Predictive Methods for 3D Human Sensing in Natural Environments', *IEEE Transactions on Pattern Analysis and Machine Intelligence* **36**(7), 1325–1339.
URL: *http://ieeexplore.ieee.org/document/6682899/*

Ji, X., Fang, Q., Dong, J., Shuai, Q., Jiang, W. & Zhou, X. (2020*a*), 'A survey on monocular 3d human pose estimation'.

Ji, X., Fang, Q., Dong, J., Shuai, Q., Jiang, W. & Zhou, X. (2020*b*), 'A survey on monocular 3D human pose estimation', *Virtual Reality & Intelligent Hardware* **2**(6), 471–500.
URL: *https://linkinghub.elsevier.com/retrieve/pii/S2096579620300887*

Jo, B. & Kim, S. (2022), 'Comparative Analysis of OpenPose, PoseNet, and MoveNet Models for Pose Estimation in Mobile Devices', *Traitement du Signal* **39**(1), 119–124.
URL: *https://www.iieta.org/journals/ts/paper/10.18280/ts.390111*

Kanazawa, A., Black, M. J., Jacobs, D. W. & Malik, J. (2017), 'End-to-end recovery of human shape and pose'.
URL: *http://arxiv.org/abs/1712.06584*

Kim, J. W., Choi, J. Y., Ha, E. J. & Choi, J. H. (2023), 'Human pose estimation using mediapipe pose and optimization method based on a humanoid model', *Applied Sciences (Switzerland)* **13**.

Kocabas, M., Athanasiou, N. & Black, M. J. (2019), 'VIBE: Video Inference for Human Body Pose and Shape Estimation'.
URL: *https://arxiv.org/abs/1912.05656*

Koleini, F., Saleem, M. U., Wang, P., Xue, H., Helmy, A. & Fenwick, A. (2025), 'BioPose: Biomechanically-accurate 3D Pose Estimation from Monocular Videos'.
URL: *https://arxiv.org/abs/2501.07800*

Li, J., Xu, C., Chen, Z., Bian, S., Yang, L. & Lu, C. (2020), 'HybrIK: A Hybrid Analytical-Neural Inverse Kinematics Solution for 3D Human Pose and Shape Estimation'.
URL: *https://arxiv.org/abs/2011.14672*

Li, Z., Chen, L., Liu, C., Gao, Y., Ha, Y., Xu, C., Quan, S. & Xu, Y. (2019*a*), 3d human avatar digitization from a single image, Association for Computing Machinery, Inc.

Li, Z., Chen, L., Liu, C., Gao, Y., Ha, Y., Xu, C., Quan, S. & Xu, Y. (2019*b*), 3d human avatar digitization from a single image, Association for Computing Machinery, Inc.

Li, Z., Hao, J. & Gao, C. (2021), Overview of research on virtual intelligent human modeling technology, Institute of Electrical and Electronics Engineers Inc., pp. 268–273.

Lin, K., Wang, L. & Liu, Z. (2020*a*), 'End-to-end human pose and mesh reconstruction with transformers'.
URL: *http://arxiv.org/abs/2012.09760*

Lin, K., Wang, L. & Liu, Z. (2020*b*), 'End-to-end human pose and mesh reconstruction with transformers'.
URL: *http://arxiv.org/abs/2012.09760*

Lin, Y., Jiao, X. & Zhao, L. (2023), 'Detection of 3d human posture based on improved mediapipe', *Journal of Computer and Communications* **11**, 102–121.

Liu, L., Xu, W., Habermann, M., Zollhofer, M., Bernard, F., Kim, H., Wang, W. & Theobalt, C. (2021), 'Learning dynamic textures for neural rendering of human actors', *IEEE Transactions on Visualization and Computer Graphics* **27**, 4009–4022.

Liu, R., Shen, J., Wang, H., Chen, C., ching Cheung, S. & Asari, V. K. (2021), 'Enhanced 3d human pose estimation from videos by using attention-based neural network with dilated convolutions', *International Journal of Computer Vision* **129**, 1596–1615.

Liu, W., Bao, Q., Sun, Y. & Mei, T. (2021), 'Recent Advances in Monocular 2D and 3D Human Pose Estimation: A Deep Learning Perspective'.
URL: *https://arxiv.org/abs/2104.11536*

Loper, M., Mahmood, N., Romero, J., Pons-Moll, G. & Black, M. J. (n.d.), 'Smpl: A skinned multi-person linear model'.

Lugaresi, C., Tang, J., Nash, H., McClanahan, C., Uboweja, E., Hays, M., Zhang, F., Chang, C.-L., Yong, M. G., Lee, J., Chang, W.-T., Hua, W., Georg, M. & Grundmann, M. (2019), 'Mediapipe: A framework for building perception pipelines'.
URL: *http://arxiv.org/abs/1906.08172*

Maji, D., Nagori, S., Mathew, M. & Poddar, D. (2022), 'YOLO-Pose: Enhancing YOLO for Multi Person Pose Estimation Using Object Keypoint Similarity Loss'.
URL: *https://arxiv.org/abs/2204.06806*

Mazuryk, T. & Gervautz, M. (n.d.), 'Virtual reality history, applications, technology and future'.
URL: *http://www.cg.tuwien.ac.at/*

Mehta, D., Sridhar, S., Sotnychenko, O., Rhodin, H., Shafiei, M., Seidel, H.-P., Xu, W., Casas, D. & Theobalt, C. (2017), 'Vnect: Real-time 3d human pose estimation with a single rgb camera'.
URL: *http://arxiv.org/abs/1705.01583 http://dx.doi.org/10.1145/3072959.3073596*

Mixamo (n.d.).
URL: *https://www.mixamo.com/*

MMHuman3D: OpenMMLab 3D Human Parametric Model Toolbox and Benchmark (2021). original-date: 2021-11-29T02:10:31Z.
URL: *https://github.com/open-mmlab/mmhuman3d*

MoveNet: Ultra fast and accurate pose detection model. | TensorFlow Hub (n.d.).
URL: *https://www.tensorflow.org/hub/tutorials/movenet*

Murugan, A., Vanukuru, R. & Pillai, J. (2021), Towards Avatars for Remote Communication using Mobile Augmented Reality, *in* '2021 IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops (VRW)', IEEE, Lisbon, Portugal, pp. 135–139.
URL: *https://ieeexplore.ieee.org/document/9419211/*

Orts-Escolano, S., Rhemann, C., Fanello, S., Chang, W., Kowdle, A., Degtyarev, Y., Kim, D., Davidson, P. L., Khamis, S., Dou, M., Tankovich, V., Loop, C., Cai, Q., Chou, P. A., Mennicken, S., Valentin, J., Pradeep, V., Wang, S., Kang, S. B., Kohli, P., Lutchyn, Y., Keskin, C. & Izadi, S. (2016), Holoportation: Virtual 3D Teleportation in Real-time, *in* 'Proceedings of the 29th Annual Symposium on User Interface Software and Technology', ACM, Tokyo Japan, pp. 741–754.
URL: *https://dl.acm.org/doi/10.1145/2984511.2984517*

Piumsomboon, T., Lee, G. A., Hart, J. D., Ens, B., Lindeman, R. W., Thomas, B. H. & Billinghurst, M. (2018), Mini-Me: An Adaptive Avatar for Mixed Reality Remote Collaboration, *in* 'Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems', ACM, Montreal QC Canada, pp. 1–13.
URL: *https://dl.acm.org/doi/10.1145/3173574.3173620*

qhanson (2022), 'Answer to "How to convert Mediapipe Face Mesh to Blendshape weight"'.
URL: *https://stackoverflow.com/a/73481375*

Rapczyński, M., Werner, P., Handrich, S. & Al-Hamadi, A. (2021), 'A Baseline for Cross-Database 3D Human Pose Estimation', *Sensors* **21**(11), 3769.
URL: *https://www.mdpi.com/1424-8220/21/11/3769*

Research, G. (n.d.), 'On-device, Real-time Body Pose Tracking with MediaPipe BlazePose'.
URL: *http://research.google/blog/on-device-real-time-body-pose-tracking-with-mediapipe-blazepose/*

Rong, Y., Shiratori, T. & Joo, H. (2021), FrankMocap: A Monocular 3D Whole-Body Pose Estimation System via Regression and Integration, *in* '2021 IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)', IEEE, Montreal, BC, Canada, pp. 1749–1759.
URL: *https://ieeexplore.ieee.org/document/9607543/*

Saito, S., Huang, Z., Natsume, R., Morishima, S., Kanazawa, A. & Li, H. (2019), 'PIFu: Pixel-Aligned Implicit Function for High-Resolution Clothed Human Digitization'.
URL: *https://arxiv.org/abs/1905.05172*

Saito, S., Simon, T., Saragih, J. & Joo, H. (2020), PIFuHD: Multi-Level Pixel-Aligned Implicit Function for High-Resolution 3D Human Digitization, *in* '2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)', IEEE, Seattle, WA, USA, pp. 81–90.
URL: *https://ieeexplore.ieee.org/document/9157468/*

Sengupta, A., Jin, F., Zhang, R. & Cao, S. (2019), 'mm-Pose: Real-Time Human Skeletal Posture Estimation using mmWave Radars and CNNs'.
URL: *https://arxiv.org/abs/1911.09592*

Shen, Z., Pi, H., Xia, Y., Cen, Z., Peng, S., Hu, Z., Bao, H., Hu, R. & Zhou, X. (2024), 'World-Grounded Human Motion Recovery via Gravity-View Coordinates'.
URL: *https://arxiv.org/abs/2409.06662*

Silva, L. J. S., da Silva, D. L. S., Raposo, A., Velho, L. & Lopes, H. (2019), 'Tensorpose: Real-time pose estimation for interactive applications'.
URL: *www.elsevier.com/locate/cag*

Sárándi, I., Hermans, A. & Leibe, B. (2022), 'Learning 3D Human Pose Estimation from Dozens of Datasets using a Geometry-Aware Autoencoder to Bridge Between Skeleton Formats'.
URL: *https://arxiv.org/abs/2212.14474*

Tejo, R. (2024), 'ricardotejo/openpose-web-demo'. original-date: 2019-11-06T14:51:29Z.
URL: *https://github.com/ricardotejo/openpose-web-demo*

Thomas, S. J., Zeni, J. A. & Winter, D. A. (2023), *Winter's biomechanics and motor control of human movement*, fifth edition edn, Wiley, Hoboken, NJ.

yeemachine on GitHub (n.d.), 'yeemachine/kalidokit: Blendshape and kinematics calculator for Mediapipe/Tensorflow.js Face, Eyes, Pose, and Finger tracking models.'.
URL: *https://github.com/yeemachine/kalidokit*

Zhan, Y., Li, F., Weng, R. & Choi, W. (2022), 'Ray3d: ray-based 3d human pose estimation for monocular absolute 3d localization'.
URL: *http://arxiv.org/abs/2203.11471*

Zhang, H., Tian, Y., Zhou, X., Ouyang, W., Liu, Y., Wang, L. & Sun, Z. (2021), PyMAF: 3D Human Pose and Shape Regression with Pyramidal Mesh Alignment Feedback Loop, *in* '2021 IEEE/CVF International Conference on Computer Vision (ICCV)', IEEE, Montreal, QC, Canada, pp. 11426–11436.
URL: *https://ieeexplore.ieee.org/document/9711286/*

Zhang, J., Jiang, Z., Yang, D., Xu, H., Shi, Y., Song, G., Xu, Z., Wang, X. & Feng, J. (2022), 'Avatargen: a 3d generative model for animatable human avatars'.
URL: *http://arxiv.org/abs/2208.00561*