UNIVERSITY OF CRETE
DEPARTMENT OF COMPUTER SCIENCE
FACULTY OF SCIENCES AND ENGINEERING

Achieving Total 3D Human Capture with MocapNETs

by

Ammar Qammaz

PhD Dissertation

Presented

in Partial Fulfillment

of the Requirements

for the Degree of

Doctor of Philosophy

Heraklion, February 2024

UNIVERSITY OF CRETE

DEPARTMENT OF COMPUTER SCIENCE

Achieving Total 3D Human Capture with MocapNETs

PhD Dissertation Presented

by **Ammar Qammaz**

in Partial Fulfillment of the Requirements

for the Degree of Doctor of Philosophy

APPROVED BY:

Author: Ammar Qammaz

Supervisor: Antonis Argyros, Professor, University of Crete

Committee Member: Constantine Stephanidis, Professor, University of Crete

---

Committee Member: Xenophon Zabulis, Research Director, ICS-FORTH

Committee Member: Panagiotis Trahanias, Professor, University of Crete

---

Committee Member: Nikolaos Komodakis, Professor, University of Crete

---

Committee Member: Kostas Daniilidis, Ruth Yalom Stone Professor, University of Pennsylvania

---

Committee Member: Markus Vincze, Professor, Technical University of Vienna

---

Department Chairman: Antonis A. Argyros, Professor, University of Crete

Heraklion, February 2024

Πάντα κατ’αριθμόν γίγνονται.

Πυθαγόρας

Acknowledgments

I would like to express my heartfelt gratitude to all those who contributed to the completion of this thesis:

- First and foremost I would like to thank Professor Antonis Argyros for giving me the opportunity to join the FORTH-ICS-CVRL group. His unwavering belief in me, his invaluable supervision, constructive feedback, support and guidance in difficult times as well as all the time he generously committed are what made this thesis possible.
- To my wife, Elina Paflioti, for her companionship in this journey, emotional support, positivity and practical assistance contributing with recorded videos of her as datasets and willingness to be involved in method user-testing.
- The Foundation for Research and Technology Greece (FORTH) for its financial support and the generous supply of computing and networking resources.
- The Hellenic Foundation for Research and Innovation (HFRI) under the HFRI PhD Fellowship grant (Fellowship Number: 1592) and by HFRI under the "1st Call for HFRI Research Projects to support Faculty members and Researchers and the procurement of high-cost research equipment", project I.C.Humans, number 91.
- The "Archimedes" Center for Research in Artificial Intelligence of Athena Research Center and especially Christos Papadimitriou, Constantinos Daskalakis, Timos Sellis, Kostas Daniilidis, Katerina Fragkiadaki, Alexandros Dimakis, Sergios Theodoridis, Dimitris Papailiopoulos, for organizing, hosting, accepting me in their summer school program, their valuable lectures and feedback to my many inquiries.
- Georgios Pavlakos for his pioneering and inspiring "Expressive Body Capture" work, his time and valuable feedback regarding this thesis during ICCV 2023.
- Constantine Stephanidis for his lifetime work on AmI, that inspired the HCI aspect of this work and his constructive feedback as part of the PhD committee.
- Xenophon Zaboulis for his trust, constructive feedback and giving me the opportunity to test and improve the method in the context of Project Mingei.
- Tassos Roussos for literature and guidance in regards to the facial capture problem.
- Nikos Komodakis for his exceptional neural networks class during the first PhD semester, that provided significant theoretical knowledge and insight in the subject.
- Vasilis Nicodemou, Filippos Gouidis, Kostas Bacharidis and Victoria Manousaki for countless hours of troubleshooting, sanity checks and stimulating lab discussions.
- Aggeliki Tsoli for reviewing and helping with various MocapNET related proposals.

- Stylianos Piperakis for our explorations in the context of controlling humanoid robots, suggesting the SeLU activation function and Butterworth filter smoothing.
- Kostis Tzevelekakis, with whom we developed the generative score-based 3D pose estimator during the final year of the PhD. For the countless conversations about our progress on our daily meals at the university dining area and short sailing trips.
- Iasonas Oikonomidis for exploring applications of the method in the context of the SignGuide project, his suggestion of Sobol sequences as a sampling technique and donation of the NVIDIA Quadro card he was awarded during 2021-2022.
- Kostas Papoutsakis for being an awesome colleague, co-traveler during the BMVC19 trip to present MocapNET, exploring applications of the method in the context of the SustAGE program and provision of datasets that helped improve the method.
- Pashalis Panteleris for providing the DeepJoint 2D joint estimation engine bolstering the "popular" MocapNET Github demos. For volunteering to be recorded on the first datasets that could be handled by the method. His positive remarks towards very early unsuccessful tracking output when the method was at its infancy, boosted my morale making me pursue it instead of abandoning it for a less ambitious goal.
- Nikolaos Kyriazis for his pioneering MBV work that influenced the design of the HCD algorithm, the great discussions during our trip in Bologna and advice and guidance in regards to the BonsApps program.
- Damien Michel for being an amazing coworker and person. Our friendly competition between our "rival" body pose estimation methods gave me motivation, his method's high accuracy and performance focus, his insightful and constructive comments helpful feedback and his experiments with Ceres/Levmar IK practical help.
- The authors would like to gratefully acknowledge support for this research from the VMware University Research Fund (VMURF).
- The support of NVIDIA Corporation with the donation of a Quadro P6000 GPU used for the execution of this research.
- BonsApps (EU H2020 Grant no.101015848) and Matteo Sorci for the award of the AI Talent grant (Winner No. Bons_1OC_20).
- This work was partially supported by EU H2020 projects Mingei (Grant No 822336) and Co4Robots (Grant No 731869).
- In memory of Maria Michail Manassaki whose bequest supported this research.

To all the individuals and organizations mentioned above, I am deeply grateful for your support, encouragement, contributions. Your assistance has been instrumental in the successful completion of this PhD thesis, and I will cherish your valuable contributions throughout my career. Thank you all from the bottom of my heart.

In addition, I wish to acknowledge the following individuals for their invaluable contributions and camaraderie during this research journey:

- George Papadopoulos, George Karvounas and Dennis Bautembach for being awesome coworkers but also, great neighbors. Our shared wireless internet connection was essential for the completion of this work, while, the long talks, barbeques, daily walks with the dogs on the neighboring field, movie and take-out nights made some difficult days manageable.
- My brother and friends that despite the geographical distance and their busy lives where there after a long day to share a chat and a game of DoTA with me.

Abstract

The goal of the presented thesis was to investigate and develop a novel, fast, portable, robust and accurate plug and play 3D Human Capture module that receives RGB images captured in-the-wild and regresses the 3D body configuration of any depicted person in the scene. The proposed architecture was built from scratch using first principles and taking advantage of recent advancements in Neural Networks, taking its final form as an ensemble of neural networks. We identified and bridged gaps between state-of-art deep learning methods and well-established model-based vision methodologies predating CNNs. Its name, "MocapNET" was coined to concisely describe it as it became the first neural network-based method in the literature to directly regress Motion Capture (Mocap) output in an end-to-end fashion. To improve accuracy and address personalization aspects, a novel real-time generative optimization algorithm was also developed named "Hierarchical Coordinate Descent" and tailored to the conditionally independent encoders of the MocapNET ensemble complementing their output. The ambition and scope of the retrieved 3D output gradually broadened as the method successfully generalized to more articulated structures during the course of its development. The total 3D capture solution presented includes upper body, lower body, hands, face and gaze. With the term 3D Human Capture we refer not only to positions in a 3D space but rather, the full kinematic solution of the skeleton. The method performs in real-time and its output is natively compatible with 3D editing software due to its BVH container. This makes it globally unique and among a select very few methods that can successfully tackle all these sub-problems that traditionally were sub-fields of the broader computer vision research. The 3D human pose estimation solution developed can be used in devices such as mobile phones, AR/VR headsets, self-driving cars, smart devices, home and factory robots etc, endowing them with capabilities to perceive, compare and enumerate human body poses, which would ultimately facilitate understanding of human behavior. The thesis attempts to carefully document all the aspects of the method including 2D shape descriptors, NN design, PCA compression to allow usage on mobile devices and the various attempts that shaped the method to its final version.

Keywords: MocapNET, 3D human pose estimation, holistic motion capture, hierarchical coordinate descent (HCD), inverse kinematics (IK), neural network ensemble.

Supervisor: Antonis Argyros, Professor, Computer Science Department, Univ. of Crete

Περίληψη

Ο στόχος αυτής της διδακτορικής εργασίας ήταν να διερευνήσει και να αναπτύξει μία νέα, γρήγορη, φορητή, αξιόπιστη μέθοδο για τρισδιάστατη εκτίμηση της πόζας του σώματος των ανθρώπων που να λαμβάνει εικόνες από κάμερες χαμηλού κόστους και να εξάγει με ακρίβεια την τρισδιάστατη διαμόρφωση του σώματος ενός επιλεγμένου ατόμου που απεικονίζεται στη σκηνή. Η προτεινόμενη αρχιτεκτονική εκμεταλλεύτηκε τις πρόσφατες εξελίξεις στα Νευρωνικά Δίκτυα παίρνοντας την τελική της μορφή ως μια σύνθεση ενός σύνολου κωδικοποιητών νευρωνικών δικτύων. Εντοπίσαμε και γεφυρώσαμε κενά μεταξύ των μεθόδων βαθιάς μάθησης αιχμής και των παλαιότερων και πιο παγιωμένων μεθοδολογιών όρασης που βασίζονται σε μοντέλα που προηγήθηκαν των CNN. Το όνομα της, «MocapNET» επινοήθηκε για να την περιγράψει συνοπτικά, καθώς έγινε το πρώτο νευρωνικό δίκτυο στη βιβλιογραφία που επιτυγχάνει άμεσα τη σύλληψη κίνησης (motion capture - Mocap) χρησιμοποιώντας ένα νευρωνικό δίκτυο. Για να βελτιωθεί η ακρίβεια της μεθόδου και να αντιμετωπιστούν ζητήματα εξατομίκευσης, αναπτύχθηκε επίσης ένας νέος αλγόριθμος γενετικής βελτιστοποίησης σε πραγματικό χρόνο με το όνομα «Ιεραρχική Κάθοδος Συντεταγμένων», ο οποίος εφαρμόζεται στην έξοδο των υπό συνθήκες ανεξάρτητων κωδικοποιητών του MocapNET. Η φιλοδοξία σχετικά με το εύρος της ανακτημένης 3Δ εξόδου σταδιακά διευρύνθηκε καθώς η μέθοδος γενικεύτηκε με επιτυχία σε πιο πολλές αρθρωτές δομές του ανθρώπινου σώματος. Η συνολική λύση τρισδιάστατης εκτίμησης της πόζας περιλαμβάνει το πάνω και κάτω μέρος του κορμού του σώματος, τα χέρια, το πρόσωπο και το βλέμμα. Το MocapNET είναι μία από τις πολύ λίγες μεθόδους που μπορούν να αντιμετωπίσουν με επιτυχία όλα αυτά τα υποπροβλήματα που παραδοσιακά αποτελούν υποπεδία της ευρύτερης έρευνας στην υπολογιστική όραση. Με τον όρο τρισδιάστατη λήψη δεν αναφερόμαστε μόνο σε θέσεις σημείων σε έναν τρισδιάστατο χώρο αλλά στην πλήρη κινηματική λύση του σκελετού. Η μέθοδος λειτουργεί σε πραγματικό χρόνο και η έξοδος της είναι άμεσα και εγγενώς συμβατή με λογισμικά 3Δ επεξεργασίας, λόγω της κωδικοποίησης BVH. Αυτό καθιστά το MocapNET παγκοσμίως μοναδικό. Επίσης, η εκτίμησης της 3Δ ανθρώπινης πόζας που αναπτύχθηκε μπορεί να χρησιμοποιηθεί σε συσκευές όπως κινητά τηλέφωνα, γυαλιά εικονικής πραγματικότητας, αυτοοδηγούμενα αυτοκίνητα, έξυπνες συσκευές, οικιακά και εργοστασιακά ρομπότ κ.λπ., προσδίδοντάς τους δυνατότητες αντίληψης, σύγκρισης και απαρίθμησης στάσεων του ανθρώπινου σώματος, κάτι που θα διευκολύνει τελικά την υπολογιστική κατανόηση και ερμηνεία των ανθρώπινων δράσεων. Η διατριβή επιχειρεί να τεκμηριώσει προσεκτικά όλες τις πτυχές της μεθόδου, συμπεριλαμβανομένων των 2Δ περιγραφέων σχήματος, της συμπίεσης PCA για να επιτρέπεται η χρήση σε κινητές συσκευές και τις διάφορες προσπάθειες που διαμόρφωσαν τη μέθοδο μέχρι την τελική της έκδοση.

Λέξεις κλειδιά: **MocapNET**, **εκτίμηση 3Δ πόζας ανθρώπινου σώματος**, **ολιστική σύλληψη κίνησης**, **ιεραρχική κάθοδος συντεταγμένων**, **αντίστροφη κινηματική**, **σύνολα νευρωνικών δικτύων.**

Επόπτης: Αντώνης Αργυρός, Καθηγητής, Τμήμα Επιστήμης Υπολογιστών, Πανεπιστήμιο Κρήτης.

Contents

List of Figures

List of Tables

Preface

The document you are reading represents the culmination of four years of intensive PhD research, which came after half a lifetime of devoted Computer Science studies. My journey in this field began at the age of 16 when I learned programming which by the age of 18 allowed me to suceed on the national programming contest of Greece, aspiring to represent my country in the Olympiad of Informatics. Witnessing the remarkable accelerating advancements in modern computer capabilities and the transformative power of Machine Learning, especially Deep Neural Networks, everything seemed to align perfectly for me to embark on this research journey.

Throughout the development of this thesis, progress in the field has been nothing short of astonishing, with "Foundation Model" breakthroughs like Large Language Models (ChatGPT, LLama [55], Bard), Generative Score Based Methods (DALL-e, Stable Diffusion [56] and CoDi) and a host of other techniques like CLIP [57], DINO [58] and NeRFs [59] pushing the boundaries of what was previously considered possible in Computer Vision. Yet, amidst these great scientific achievements towards Artificial General Intelligence, I write these lines during one of the worst summer heatwaves on record, a very challenging global economic situation and especially for Europe and Greece due to the war in Ukraine, and with a considerable amount of the work for this thesis carried out under COVID-19 lockdowns and at a state of uncertainty about the future of humanity.

In this bittersweet time, reflecting on the work accomplished, I find great satisfaction that the effort put into this thesis was well worth it. Not only does it contribute to the academic landscape with theoretical novelty in a field that is blossoming and at a time of great advancements, but it also holds practical value, with numerous applications for ordinary people that I had the privilege to explore and document. My hope is that the love, time, passion, and dedication poured into developing and writing this thesis are reflected and preserved in these pages. I would like to sincerely thank you for taking the time to go through it, hoping that you find it a valuable source of knowledge to build upon.

This thesis stands as my humble contribution, to the best of my abilities, to the intricate computations of our universe.

Chapter 1
Introduction

Moving away from traditional static computing to a mobile first / cloud first ubiquitous generation of computing devices that sense the world around them in the same way humans do, they will need to be well-equipped for direct natural scene understanding to achieve operational parity to their user high expectations. We live in a human-centric world where the development of holistic methods and systems that will be able to accurately and efficiently perceive humans will play an integral part in the success of efforts to further revolutionize the way we live.

By leveraging recent advancements in deep neural networks (DNNs) and NN ensembles and by crafting novel algorithms that build on them, we propose a method for real-time 3D human motion capture from monocular visual data. We foresee that vision-enabled smart devices and robots that are able to capture humans in 3D will be able to understand and anticipate the needs of the users more effectively than current solutions, thus offering smarter and more immersive user experiences, facilitating effective human-machine collaboration, as well as human-human interaction when using VR as a mediator platform.

1.1 General Objective

Our goal and thesis objective, was to investigate and develop a **fast**, **portable**, **robust** and **accurate** plug and play module, that **receives RGB images** captured in-the-wild and estimates the 3D **body**, **hand**, **gaze** and **facial configuration** of the depicted person in that scene **outputting BVH motion capture data**. We aimed at providing 3D human pose estimation solutions that rely on **well defined open standards** that are already used to represent motion capture and facilitate motion playback on devices such as personal computers,

gaming consoles, Augmented Reality (AR)/Virtual Reality (VR) headsets, mobile phones and aiming to extend them to smart devices, home and factory robots etc. We developed and implemented our solution from scratch using first principles resulting in a fast and lean code-base without many dependencies.

The significance of the presented research is showcased by the fact that prior to our first publication during this PhD there were no total capture solutions available due to their complexity and extensive surface area spanning through previously different subdomains of computer vision research. However during the development of this PhD and at a time of incredible progress in the field there are now a handful [36, 60–64] of state-of-the-art architectures that can perform holistic 3D total human pose capture. Each of them has its unique pros and cons that we will examine in our literature review, however none of them exhibit the unique characteristics and performance advantages of our formulation. We aim for our method to be a drop-in replacement motion capture source endowing connected applications to its **real-time** output with capabilities to perceive and compare human body poses, and thus facilitate human computer interfaces and understanding of human behavior. We pose that human behavior understanding is best done holistically and, thus, the scope of the retrieved 3D body includes not only 3D positions of **body** keypoints, but the full kinematic solution of the skeleton as well as of the human **hands**, **gaze** and **face**. The significance of our work to the objective is not only theoretical with our published research [6, 8, 9, 20, 31] but also practical with our open implementation on Github [65]. We also believe that it has already inspired change to the field. A series of commercial webservices (Table 1.2) for motion capture from monocular RGB like DeepMotion [66] Radical Motion [67], Plask [68], RushMotion [69] and Mixamo [70] started to appear shortly after our first publications. Our method is disruptive since it allows a "democratization" of motion capture. Using our research, it can now be performed in real-time and with decent accuracy without expensive sensors and suits at a fraction of the cost normally expected from existing commercial MOCAP systems. The cost of setup and hardware has been replaced by our formulation that can perform the task using a single monocular camera source and solving the complex and high-dimensional problem using software.

Figure 1.1: We envision the research presented powering a multitude of applications ranging from entertainment and commerce to robotics and security.

1.2 Motivation and Vision

Smart assistants such as Google Home, Amazon Alexa, Apple Siri and Microsoft Cortana, have gained great popularity recently enabling users to interact with both in-house and third-party services via voice commands. Leveraging Computer Vision technologies in a smart environment is expected to revolutionize the functionality of smart devices as well as their interaction with humans. For instance, in addition to voice commands that are now commonly used, users could interact with devices via gestures inferred from capturing the motion of 3D human body from visual data. Moreover, having cameras observe humans in the scene performing their daily activities can provide useful insights about the type of activity being performed as well as the habits of a person in a seamless and unobtrusive way. This information in turn can be used to provide personalized services to a user e.g. adjust the lighting of the room based on the user's activity or detect unusual behavior that may be indicative of an emergency situation e.g. an elder that has fallen and is in need of assistance.

The capability to observe humans in 3D can also be very beneficial to robotic agents in both home and industrial environments. Latest efforts in Google Deep Mind Robotics in the context of the Everyday Robot project with recent RT2 publications [71] have focused on manipulation of objects adding navigation in crowded scenes and human robot interactions will require a solution like the one we propose. When human and embodied robotic assistants coexist in the same environment, knowledge about the motion and activity of the human is of paramount importance for both tasks. Proper perception coupled with a Large Language Model [72] cognitive engine can uplift a robot from a mere household appliance to a valuable and valued household "presence" that its user can even form emotional bonds with [13]. Although there are quite a few technical robotics problems that still need to be resolved, many of the typical tasks that a robotic assistant will need to perform will rely on proper user detection and tracking. These tasks could include identifying the people in the scene, maintaining a proper distance from the user, gracefully handing him objects, inferring the emotional or physical state of the human, being able to track and anticipate human intent, movement and gaze as well as cooperating with humans in tasks ranging from receiving commands via gestures to collaborative transportation of objects in the scene.

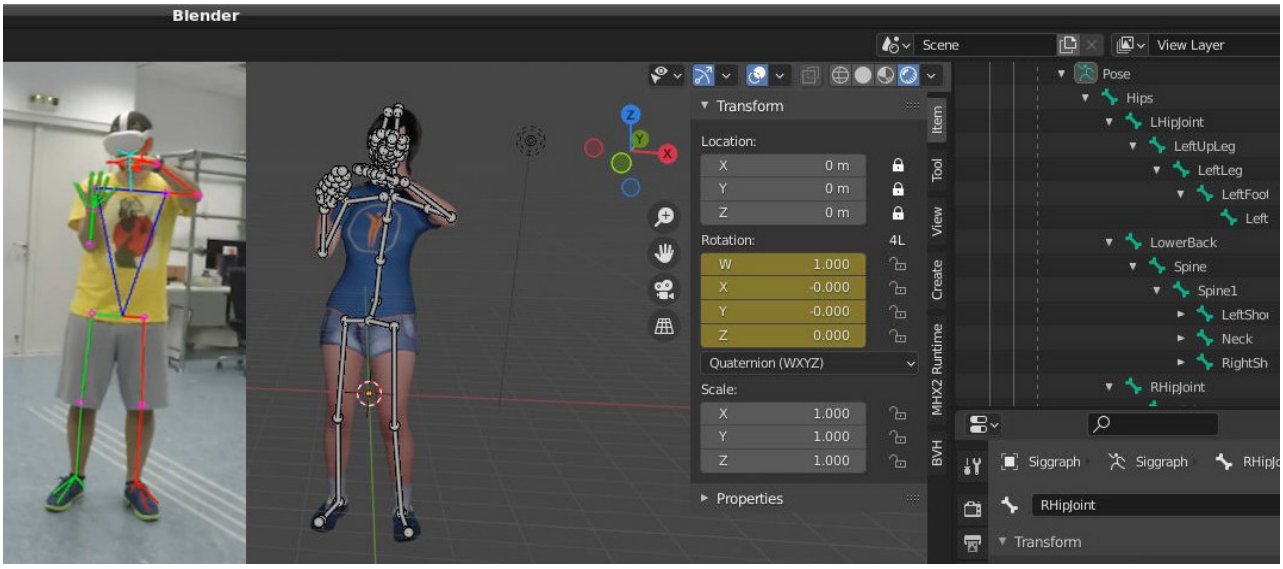


Figure 1.2: The presented method acts as a real-time bridge between off-the-shelf RGB cameras and 3D applications. Since our method's output includes Inverse Kinematics and uses the open standard BVH file format it is directly compatible with a wide range of software including the Blender 3D editor that is shown in this illustration.

Capturing the motion of humans in the real world can additionally benefit systems related to AR/VR such as Oculus/Meta Quest, Google Glass or Microsoft Hololens. High accuracy 3D human estimation from onboard cameras enables realistic 3D rendering of avatars for replaying observed human motions, potentially from different viewpoints, or even interacting with the avatars in real-time. Having the ability to capture multiple people simultaneously additionally allows studying and simulating social interactions. That promotes immersive experiences in a series of applications such as multiplayer video games, navigation using AR, demonstration of tasks in industrial settings and virtual social networks. An interesting design characteristic of our work is that we directly output a BVH animation file with the captured human motion bringing our technology one step closer to existing software for human animation in virtual worlds (Figure 1.2).

We envision a world where AI-powered robots will serve humanity and provide all of the fundamental hard and menial everyday labor allowing humanity to flourish and focus on an existence with less suffering, a higher standard of living, and more time to pursue higher goals than spending much of their lives on routine problems that can be automated away. Perceiving humans with computer vision, is bound to play a critical role in this vision and these thoughts summarize our motivation behind this work.

1.3 Research Questions

With this work we attempt to shed light into some important questions we encountered in our effort to achieve our stated objective. Its worth noting that many of them formed gradually and while trying to answer other more naïve questions with the primary one being: *"How can we develop a fast, portable, robust and accurate plug and play module that receives RGB images captured in-the-wild and estimates the 3D body,*

hand, gaze and facial configuration of the depicted person in that scene in the form of motion capture data?".

Throughout this thesis we will come back to these questions and gradually offer solutions to them while connecting pieces of the motion capture from RGB puzzle and describing our answers to them which formed our approach and our final solution.

The core research questions we attempt to address in this thesis are:

1. Given an articulated 3D structure that we want to approximate using 2D projections of some of its landmarks recorded from a conventional camera sensor:
   (a) How can we divide and conquer the very high dimensional configuration space of the problem to effectively treat it?
   (b) How can we split our solution in transparent parts where we can have intermediate control and monitor their internal state for explainable Artificial Intelligence (AI) instead of a monolithic black-box solution?
   (c) What is a good/compact 2D descriptor to facilitate the machine learning task?
   (d) How can we quantify the amount of information carried by descriptors and how can we shape them to optimize them in terms of representational capacity?
   (e) How would someone go about designing a compact neural network architecture for the task?
   (f) What are the ethical considerations behind such a system?
2. In terms of the data needed to facilitate training:
   (a) How can we prepare ground truth for a non/semi/fully suppervised method that can target different sub-regions of the human body?
   (b) How can we study a given dataset and what are potential biases encountered?
   (c) How can we have end-to-end regression in our native data format to minimize processing latency?
3. In terms of the Neural Network ensemble proposed:
   (a) How can we facilitate training given an articulated structure?
   (b) Which neural network architecture is the most compact for the task?
   (c) How can we handle occlusions and leverage symmetrical properties?
   (d) How is an encoder internally performing its task?
4. In terms of the Accuracy/Performance of the Resulting method and given an estimation by the proposed neural network:
   (a) How can we adapt our output to personalise and improve the fitting of our output for subjects departing from an average human phenotype?
   (b) Given the hierarchical nature of an articulated structure what is an efficient way to improve the accuracy of results?
   (c) How different parameters affect the resulting pose accuracy?
   (d) What are the 2D to 3D accuracy limits for a monocular system like ours?

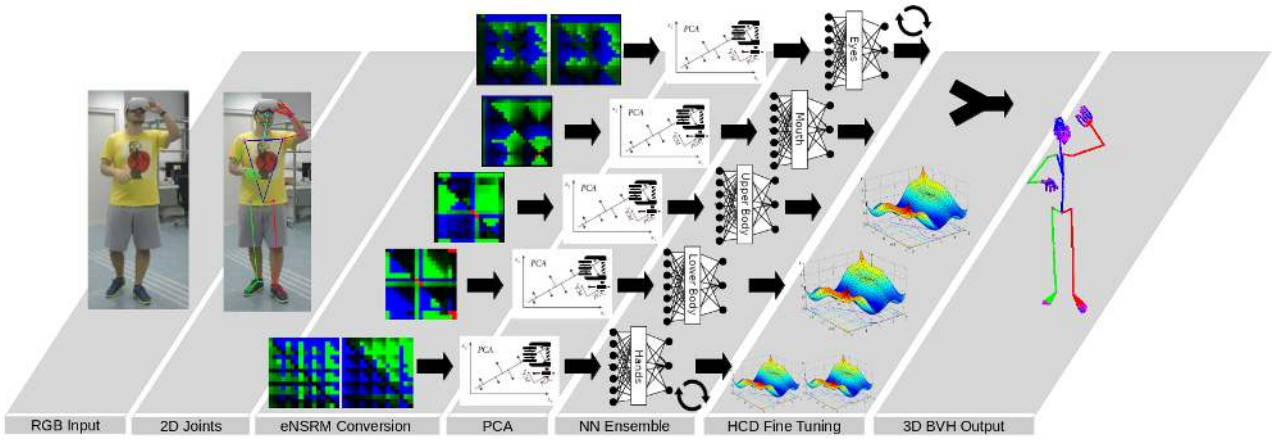


Figure 1.3: Method outline, series, concurrency and discrete steps of our formulation from an RGB image to a 3D BVH pose featuring body + hands + gaze + face. Hands and eyes are regressed via the same ensemble by leveraging their symmetry property.

(e) Why does our Hierarchical Coordinate Descent fine tuning algorithm work?

Many of the questions mentioned will be answered in the methodology Chapter 3 while for some of them we will experimentally attempt to provide an answer in the explored ideas Chapter 4 and experiments Chapter 5.

1.4 The Approach

Depending on the needs of an application there are a lot of ways to facilitate human sensing. The lowest fidelity sensing techniques provide detection in the form of a classification score for the whole image [73], bounding boxes [74] localized in a specific region of the input image or text summarization [75]. While these may be enough for simple presence detection, typically a richer representation for activity tracking involves the retrieval of 2D points for each joint of the person [4]. Making the step from 2D to 3D points typically entails an estimation of a depth measurement for each joint [76]. Our approach however goes a step further by fully regressing the "inverse kinematics" configuration of the skeleton, yielding a single position for the whole armature and encoding pose as joint rotations across a kinematic chain with each of them being independent of the others. Although the bulk of the task is handled by the Neural Network (NN) Ensembles, which immediately and in an end-to-end fashion produce 3D output encoded in the Bio Vision Hierarchy (BVH) [77] format, our full formulation involves a list of discrete steps from RGB image to 3D BVH output that are illustrated in Figure 1.3.

Although Figure 1.3 might appear overwhelming at first, as we will clearly explain in the methodology Chapter 3 some of the steps are optional, with PCA used to compress the network for mobile devices for extra performance and HCD fine-tuning used to improve accuracy depending on the application and the eNSRM conversion code requiring just a few hundred lines of code.

Our formulation is elegant since individual steps from RGB to BVH are well seperated and independent. We receive an RGB image and treat it through a 2D joint estimator like Openpose [4] or Mediapipe Holis-

tic [78] which yield 2D joints. After encoding this 2D data using our descriptor we pass it through our ensemble and the final 3D BVH output is assembled by simply concatenating the parts (degrees of freedom) handled by each ensemble. Occluded sub hierarchies of the body can be thus skipped without affecting the rest of the body, while we can enforce a variety of policies to the occluded parts (remembering their last visible state, extrapolating their previous position based on their velocity, or setting them to sensible defaults).

Our approach offers huge room for parallelism. All of our formulation is designed to be executed in Central Processing Unit (CPU) in real-time with the bottleneck being the RGB to 2D joint regression. Each of the discrete steps seen can happen in isolation of others in a multi-core CPU and multi-threaded capable Operating System kernel. Furthermore we can offload the 2D joint estimation that heavily relies to convolutions to a General Purpose Graphics Processing Unit (GPGPU) and pipeline execution. This way while the Graphics Processing Unit (GPU) is handling the newely acquired RGB frame and producing 2D joints CPU resources can concurrently estimate the previous 3D BVH output. This can further boost performance and improve hardware utilization. Finally taking advantage of the symmetries of the human body we can "horizontally flip" our 2D input and 3D output and thus handle different symmetric parts of the body using the same ensembles conserving memory and further improving performance due to better instruction fetching rates.

In conclusion our flexible architecture which we will explain in detail through this thesis, divides the very difficult problem of Total Human Capture to smaller ones that can be individually addressed very fast or even entirely skipped with minimal consequences to the rest of the results. The runtime of "MocapNET" is very compact and straightforward compared to other methods with our initial implementation achieving 400Hz [6] when considering the 2D Joint to 3D BVH Output, while even our final implementation that addresses the whole body using a relatively slow python interpreter can work at real-time rates that match the $\approx 25Hz$ capture rate of modern web-cameras.

1.5 Contributions of this Dissertation

After studying the literature and identifying its gaps we formulated a novel hybrid method to regress high-dof articulated structures from their 2D landmark projections. The work we present started as a 2D to 3D body pose estimator built from scratch and from first principles with our BMVC'19 publication [6]. As we will see in the next literature study Chapter, there are many methods that deal with the same task, however our unique contributions in our first formulation was regressing Motion Capture (BVH) data instead of just 3D points which was the norm before our work in an end-to-end manner, the novel Normalized Signed Distance Matrix (NSDM) encoding, coupling an orientation classifier with subsequent encoder ensembles for specific views of the subjects to divide and conquer the problem and employing a very shallow and wide fully connected network with only 4 layers that allowed for incredibly high framerates of $\approx 400Hz$ when executed on CPU. Drawbacks of the initial method where the mediocre response to occlusions due to using a single matrix for the whole armature and low accuracy compared to other 2D to 3D methods. Given our initial work we set out to work on alleviating its drawbacks. During ICPR'21 [8] we presented the improved and novel Normalized Signed Rotation Matrix (Normalized Signed Rotation Matrix (NSRM)) descriptor that utilized half the elements of Normalized Signed Distance Matrix (NSDM), we managed to further split pose space in four instead of two orientations, and the body hierarchy to upper and lowerbody providing much better handling of subjects under heavy occlusions. Furthermore we proposed the novel Hierarchical Coordinate Descent (HCD) algorithm and studied the biases of ground-truth motion capture data dramatically

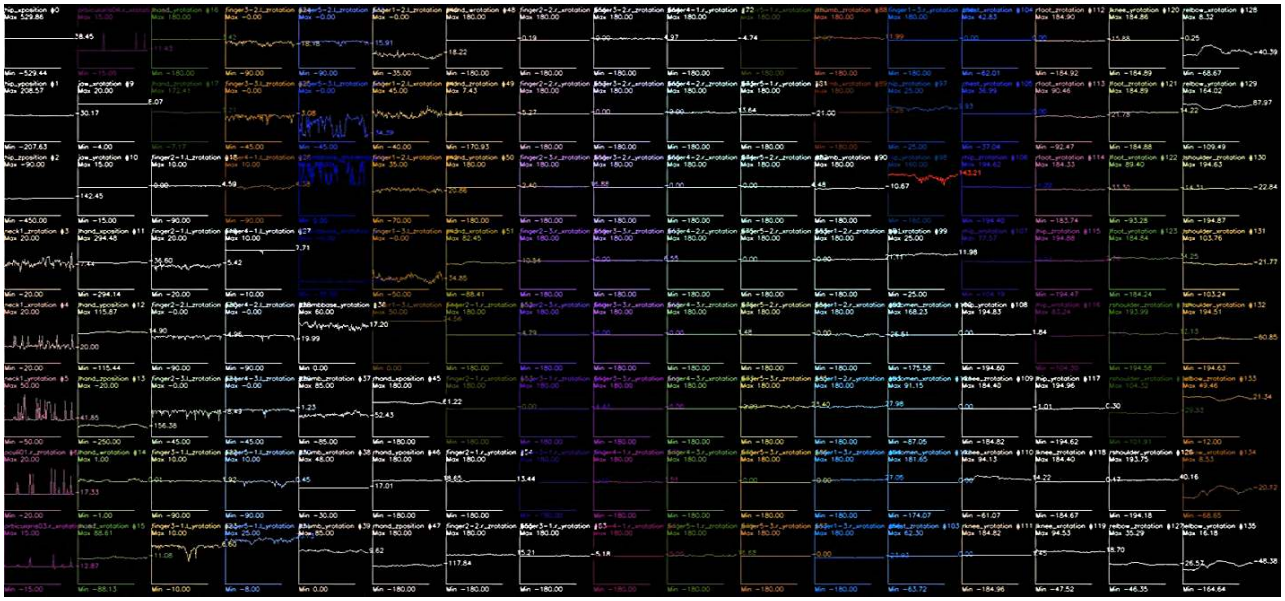


Figure 1.4: MocapNET when tracking a person from an RGB video stream decomposes the observed pose in a series of independent (since each part's rotation is not influenced by the rotations of other parts) signals. Each of the plots represents in the vertical axis the variation of the euler angle value of one degree of freedom across the horizontal axis of time. Our method tackles the whole problem of upper+lower body + hands + face and gaze in real-time offering a rich source of 3D MOCAP data.

increasing the accuracy while maintaining real-time performance. Having a stable prototype for the upper and lower body, continuing our effort we incorporated 3D Hand Pose estimation in BMVC'21 [9] further enhancing our descriptors proposing the "enhanced Normalized Signed Rotation Matrix (enhanced Normalized Signed Rotation Matrix (eNSRM))" that also included scale information, and overcoming the need for orientation classification after reaching its logical conclusion and instead using deeper networks. After this publication the success of our method and the requirements of the BonsAPPs project it was piloted on presented the need of further improving its computational aspects. We managed to achieve this with our PeTRA'23 work [20], that allowed deployment of the body tracking method in Raspberry Pi 4 computers at interactive framerates. During the final year of this PhD to complete the method we dealt with the 3D facial and gaze capture problem managing to incorporate it in the original formulation [31] and yielding the final version of the method we present in this dissertation.

We believe that our work comes full circle from the pioneering 1994 work of Rehg and Kanade "Visual Tracking of High Degrees of Freedom (DOF) Articulated Structures: an Application to Human Hand Tracking" [79] that first described the problem of high d.o.f articulated structures with an implementation exclusively targeting the hands. Similarly we began with a formulation designed for high d.o.f. and expanded and improved it successfully targeting hands and then faces thus providing a solution that handles high d.o.f articulated 3D pose estimation.

In terms of iterative 3D pose refinement, a long standing problem with pioneering work in 1992 with the Iterative Closest Point (ICP) algorithm [80], we contributed with the novel Hierarchical Coordinate De-

Year	Name	Body	Hands	Face	Gaze	2D	3D	Realtime
2019	Monocular total capture [36]	✓	✓	✓	✗	✓	✓	✗
2019	Expressive body capture [60]	✓	✓	✓	✗	✓	✓	✗
2020	Monocular Real-Time Full body capture [61]	✓	✓	✓	✗	✓	✓	✓
2021	Lightweight multi-person total motion capture [83]	✓	✓	✓	✗	✗	✓	✗
2022	ZoomNAS [63]	✓	✓	✓	✗	✓	✗	✓
2023	Reconstructing Signing Avatars [64]	✓	✓	✗	✗	✓	✓	✗
2023	This PhD Proposal / MocapNETs	✓	✓	✓	✓	✗	✓	✓

Table 1.1: **Total 2D/3D capture of the combined human body+hands+face** became feasible shortly after the start of this PhD in 2019 with the seminal Monocular total capture [36] CVPR paper from CMU. With the acceptance and completion of the work described in this PhD proposal, MocapNET will gain its final facial ensemble which will make it one of very few methods globally that tackles the combined problem in 3D, let alone in real-time

scent algorithm which is an algorithm tailored to our neural network ensemble output and its conditionally independent encoders.

We provide a novel unified solution for total human capture aspects of which have been long studied for close to three decades for body [81], hands [79] and face [82].

We provide a well crafted method that offers multiple methodological advantages and showcases how to clearly take advantage of many interesting properties of the human bodies. while experimentally applying the programmatic implementation of our approach on a wide range of applications to study it and record our findings in this dissertation.

Summarizing the contributions of this dissertation, to the best of our knowledge:

1. Our Neural Network ensemble method is the first to regress BVH motion capture data (3D+Inverse Kinematics) in an end-to-end fashion.

2. From the handful of Total capture methods (Table 1.1) appearing after our first publication, our proposed method is currently state of the art in terms of computational complexity.

3. The 2D descriptors (NSDM, NSRM, eNSRM) we propose are novel and exhibit both theoretical and practical significance when coupled to neural networks.

4. The hierarchical coordinate descent (HCD) iterative pose refinement algorithm we propose is novel.

5. "MocapNET" takes advantage of symmetries, gracefully handles occlusions and addresses the Total Capture problem in a direct, elegant and compact fashion.

6. In our effort to train, optimize and deploy our solution, we offer many other smaller contributions like hard sample mining, neural network compression through Principle Component Analysis (PCA), EigenPoses, 3D Pose Diffusion and others that exhibit theoretical novelty with various levels of success.

7. The proposed approach has the potential to impact various applications in fields such as human-computer interaction, virtual reality, gaming, surveillance, and healthcare, and advance the state-of-the-art in human behavior understanding research. This research contributes to the field of computer vision and human behavior understanding by addressing the limitations of existing methods and enabling a more comprehensive and accurate understanding of human behavior from visual data.

Year	Name	Body	Hands	Face	Gaze	Cost
2017	Radical Motion [67]	✓	✓	✓	✗	8$/month
2021	DeepMotion [66]	✓	✓	✓	✗	9-83/month
2021	RushMotion [69]	✓	✓	✗	✗	15$/month
2022	Plask [68]	✓	✓	✗	✗	50$/month
2022	Pixcap [84]	✓	✓	✗	✗	7-30/month
2023	Move.AI [85]	✓	✓	✗	✗	Not announced

Table 1.2: **Commercial** SaaS websites recently appeared providing **offline** motion capture extraction from monocular videos, their closed-source implementations are closely guarded to protect their competitive advantages.

1.6 Ethics

AI research has significant transformative societal impact, as evidenced by the recently proposed European Union (EU) AI Act. In light of this, we recognize the importance of addressing the ethics of our research. Before concluding our introduction Chapter, we assess ethical implications of our work providing the reader with insights into our approach.

(1) **Impact on Citizens and Society**: We have carefully reflected on the effect of our application on fundamental rights, including safety, health, non-discrimination, and freedom of association. Our method relies on anonymous 2D joint input that does not contain any gender, age, ethnicity, or subject identification cues, ensuring safety and non-discrimination. Even aspects such as favoring right-handed versus left-handed individuals through better or worse tracking is taken into account since we use the same mirrored ensembles for the left/right hands and eyes. Furthermore freedom of association concerns do not apply to our case, since the method does not contain any subject identifying information.

(2) **Privacy and Data Protection**: We comply with the General Data Protection Regulation (GDPR). We store no bio-metric data and no video or images of the perceived persons. The skeletal output of our method is also anonymous since Bio Vision Hierarchy (BVH) skeletons are based on a fixed synthetic digital body armature of "average size". So, even in cases of people of different physical measurements, the 3D output of our method cannot be used to identify them. The training of the Neural Network model relies on anonymous Motion Capture Data collected by Carnegie Mellon University using proper procedures, that once again lack any identifying information about the recorded subjects. Moreover, the Self-Supervised synthetic data training approach we explored to cover the vast amount of data needed to train the neural network relies on random generation of physically plausible human poses in the form of skeletons with no correspondence to real humans. Nevertheless, we do consider and can accommodate skeletons of various sizes (with this information provided with the consent of the user) in order to be able to be inclusive and accommodate persons of various body shapes.

(3) **Transparency Rights**: There is no association of the data we process and output with the corresponding human. We envision applications utilizing our work enabling transparency by notifying users about data processing, providing access to information on collected personal data, allowing users to control their data, and offering explanations of the system's results.

(4) **Accessibility**: While our system itself is accessible to computer scientists, researchers and developers of diverse demographics and languages, it is designed to particularly assist people with disabilities, such as tracking sign language for interpretation. One of the core reasons for performing this work

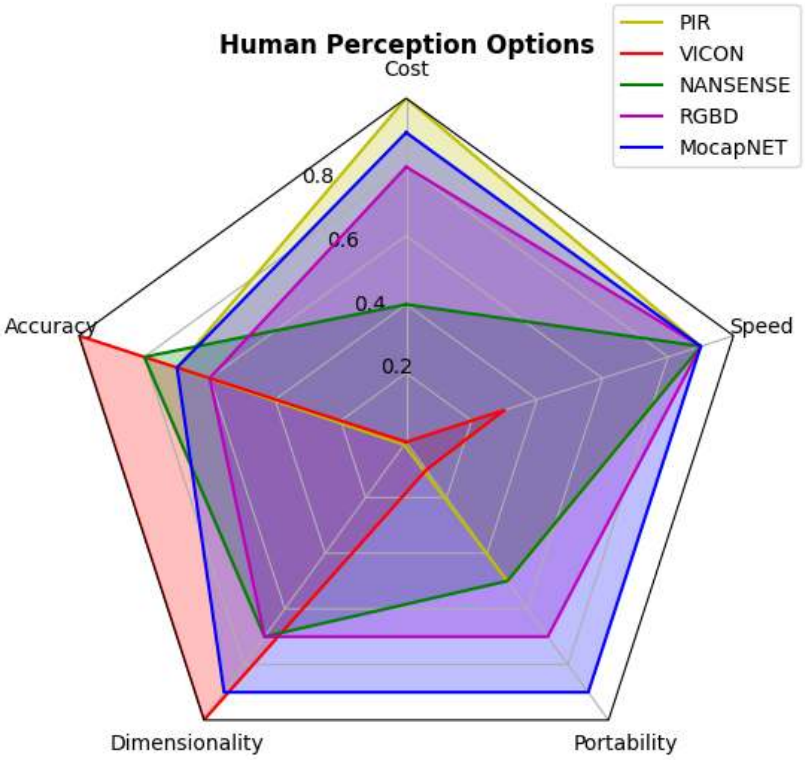


Figure 1.5: Perceiving humans with various technological solutions range from inexpensive 0.6$ Passive Infrared (PIR) motion detectors providing 1 bit of data, to commercial VICON MOCAP systems that cost 100K+ $ providing detailed MOCAP, with dedicated infrastructure and specialized suites with reflective markers. Other solutions include IMU based MOCAP suites (NANSENSE) as well as RGBD based pose estimation solutions. Our work offers unique trade-offs in all aspects making it a worthwhile endeavor.

is providing a system that can facilitate direct human sensing by computers and robots, regardless of demographics, language, disability and digital literacy. In terms of financial accessibility, we believe that the computational efficiency of our method will lower the bar "democratizing" motion capture and allowing people that own computers with limited resources like the Raspberry Pi 4 Computer (RPI4) to use our method.

(5) **Security**: BVH output is anonymous and due to the high performance of our method we envision it being immediately consumed from applications, providing immediate feedback or facilitating human computer interaction with the consent and awareness of the user. Since the BVH output is typically ephemeral and locally consumed there are no security mechanisms as part of the method.

(5) **Data Management**: As stated before, we follow the data-minimization principle, employing local and temporary storage. Our method captures and processes only strictly necessary 2D joint data, that contains no identifying information, ensuring data protection by design.

(7) **Non-AI Alternatives to our method**: An application could choose to use a lower fidelity solution to perceive humans that would, however, carry more risks since a method directly handling RGB or A Red Green and Blue color image accompanied by a Depth map (RGBD) images would pose a bigger privacy threat by potentially leaking sensitive images. Non AI MOCAP systems typically entail bulky suits with Inertial Measurement Units (IMUs) so there is no non-obtrusive equivalent like the one we propose.

(8) **Open-Source Code**: The source code for evaluating our method is provided to the scientific community under the Foundation for Research and Technology, Greece (FORTH) license which allows its use for non-commercial and research applications in our Github repository [65]. We where very careful during development of our neural networks by not using proprietary training datasets, and opted to only using permissive motion capture datasets [16, 18] or self-supervision for hands and the face.

(9) **Governance and Ownership**: Ownership of the provided implementation of our method [65], including code, data, and use, belongs to the Foundation for Research and Technology Greece, as it funded and supported its development.

(10) **Openness over Data Governance**: We maintain intermediate openness of data governance by using publicly available Motion Capture Data from Carnegie Mellon University [16, 18] and synthetic data for self-supervised training. We anonymize all used data to address privacy concerns.

In conclusion, this research was performed with a commitment to ensuring that it aligns with ethical considerations, respects existing laws, user rights, and upholds privacy and data protection principles. We believe that we provided a comprehensive insight into the ethical implications of our work.

1.7 Outline of Dissertation

After being provided with the introductory overview, the reader should now have a broad sense of the scope of our work as well as of its significance and novelty. The rest of this dissertation is organized in a way to provide details in a comprehensible way and giving insight on the various methodological choices and its soundness.

In Chapter 2, we will conduct a thorough literature review to contextualize our research within the state of the art at the time of writing this thesis and highlight the advancements our method brings in comparison.

Building on this foundation, Chapter 3 will offer a detailed analysis of our approach, dedicating individual Sections to each of the design elements employed. This will provide readers with a comprehensive understanding of our architecture and its implementation.

In Chapter 4, we will focus on method variations and experimental endeavors aimed at further refining our approach. While some of these experiments may not have yielded the intended results, they provide valuable insights by guiding researchers towards more promising paths. Even unsuccessful attempts contribute to the advancement of knowledge and lead us closer to innovative solutions.

Next, in Chapter 5, we will summarize method experiment outcomes, presenting both qualitative and quantitative measurements against ground truth. This will allow readers to assess the method's performance and effectiveness in different scenarios.

Finally, Chapter 6 will provide a comprehensive conclusion, drawing together all the details presented

throughout the dissertation. We will reiterate our goals, reflect on the provided results, and offer a holistic assessment of our method. This Chapter will be instrumental in allowing readers to form a well-informed judgment of our work in its entirety.

Chapter 2
Literature Review

Human Pose Estimation (HPE) is a very active topic with a large volume of recent, novel works. Categorization of methods is a hard task since many of them share similar ideas and have been published during the same period albeit all of them are structured and executed in vastly different ways every one of which is interesting and novel in its respective way. We will attempt to categorize works based on their architectural similarities, their underlying techniques and estimation accuracy. Decades of research on the topic of human pose estimation are briefly summarized in the surveys listed on Table 2.1.

The first architectural distinction of methods is their mode of operation. Techniques either use a **Top-Down** approach working from an abstract internal state leading to pixel level information that validates their hypotheses or **Bottom-Up** where techniques start from observed pixel information gradually combining it to deduce higher-level abstractions that gradually reveal the correct output. A related categorization of pose estimation algorithms is whether they are **generative** (i.e. relying on an internal model as seen in Figure 2.1 to generate samples that get matched to observations) or **discriminative**, (i.e methods that are model free and learn to filter input regardless of a given model). Methods can utilize **multiple input** sources or work with **monocular** data. The utilized camera technologies can also greatly vary from highly accurate calibrated RGBD cameras to cheap RGB cameras with unknown intrinsics. Output of methods can be **2D bounding boxes, 2D heatmaps, 2D joints, 3D joints, 3D angles** or **3D meshes**. The pose estimation output can be directly mapped in **one-stage** using regression or can take place in **multiple stages** starting from a basic detection and then moving in intermediate representations before being transformed to its final form. Model based methods utilize models that can be volume based using 3D meshes, contour based or skeleton based. Methods can target a **single person** and extend to multiple person pose estimation via iterative means or organically handle the **multiple persons** that may be present in a scene. Their computational performance can be **offline** for stochastic algorithms that typically produce very high fidelity output albeit very slowly, **online/interactive** for algorithms that are able to handle incoming video streams as they arrive by skipping some input frames and **real-time** for methods that are so fast that can fully process input streams that come in high framerates typically over 30 Frames Per Second (FPS). Finally there are many different optimization algorithms and techniques that can be used in conjunction with the rest of the architectural characteristics of a pose estimation algorithm however as we will see in the following Sections deep learning methods have prevailed as the state of the art and thus we will focus on these novel methods that however again can greatly vary in their internal organization as well as the type of neural network deployed.

Survey Title	Year	Publisher	Citations
Human motion analysis: A review [86]	1999	CVIU	2393
The visual analysis of human movement: A survey [87]	1999	CVIU	2626
A survey of computer vision-based human motion capture [88]	2001	CVIU	2290
A survey on visual surveillance of object motion and behaviors [89]	2004	TSMCS	2799
Advances in vision-based human motion capture and analysis [90]	2006	TSMCS	3017
Vision-based human motion analysis: An overview [90]	2007	CVIU	1012
3D human motion analysis in monocular video [91]	2008	Springer	58
Advances in view-invariant human motion analysis [92]	2009	CVIU	257
Visual analysis of humans [93]	2011	Springer	141
HPE & activity recognition from multi-view videos [94]	2012	JSTSP	124
A survey of human motion analysis using depth imagery [95]	2013	Elsevier	356
A survey of on model based approaches for 2D and 3D HPE [96]	2014	Sensors	48
A survey of HPE: body part parsing methods [96]	2015	JVCIR	52
HPE from monocular images: A comprehensive survey [97]	2016	Sensors	48
3D human pose estimation: Review of the literature [98]	2016	CVIU	121
RGB-D based human motion recognition with deep learning [99]	2018	CVIU	109
Monocular Human Pose Estimation: Deep learning methods [100]	2020	Elsevier	71
The progress of human pose estimation [101]	2020	Access	12
A comprehensive survey on 2D multi-person pose estimation [102]	2021	Elsevier	-

Table 2.1: Summary of published **surveys** on the topic of **human pose estimation (HPE)** and their research impact sorted by incremental year of publishing.

2.1 2D Human Body Pose Estimation

In order to maintain a reasonable scope in this Chapter, we will focus on monocular methods. Multi-view pose estimation methods record and offer more data and thus less ambiguous constraints for pose estimation algorithms and thus provide results that exhibit much better accuracy. At the same time they typically require elaborate calibration steps precise synchronization and since most recorded video is monocular allow much less interesting applications than the ones we target. Before we categorize recent methods that can be used as input for our work we must first begin by identifying 2D pose estimation methods from RGB images. 2D Joint estimation is a prerequisite to our method since it provides us with our input which we in turn regress to our 3D BVH output. Despite this Section specifically targeting 2D joint estimation, we must always keep in mind that 2D estimation accuracy is a deciding factor for the 3D estimation accuracy achieved by multi stage 3D pose estimation methods. Any 2D estimation jitter adversely affects 3D estimation accuracy on methods that rely on 2D joints as input to derive 3D information. Keeping those things in mind we can begin with our examination. Classic 2D human body pose approaches using manually selected features, elaborate body models and a variety of regressors [103] as well as random-forests [104] paved the wave for further research that took the tools provided by state of the art neural network research of the time to offer new methods like [105, 106] and DeepPose [107]. The stacked hourglass networks of Newell et al. [108], Convolutional Pose Machines [109] and OpenPose [4, 110] are among the most popular and influential works. Other cutting-edge research includes DeepCut [111] as well as the ones presented in [112–114].

These works motivated many researchers to incorporate neural networks for the task of 2D pose estimation, moving it away from initial human perception as image classification [73, 115]. This trend continued

Year/Ref	Framework	Input Dims.	Highlights	PCKh
Regression-based methods				
2014 [107]	AlexNet	220x220	End-to-end 2D regression based on AlexNet	-
2016 [113]	GoogleNet	224x224	Iterative error refinement from initial pose	81.3
2017 [117]	ResNet	224x224	Study of neural network efficiency	86.4
2017 [118]	ResNet	224x224	Feature combination learning	91.2
Detection-based methods				
2014 [116]	AlexNet	320x240	ConvNets + Markov random fields = heatmaps	79.6
2016 [119]	VGG	256x256	Deep CNNs + deformable mixture of parts	-
2016 [108]	Hourglass	256x256	Stacked hourglass architecture	90.9
2016 [109]	CPM	368x368	Convolutional pose machines	88.5
2017 [120]	Hourglass	256x256	Multi-scale input for increased receptive fields	91.5
2017 [121]	Hourglass	256x256	Stacked hourglass feature pyramids	92.0
2017 [122]	GAN	256x256	Generative adversarial network for pose	91.9
2018 [123]	Hourglass/GAN	256x256	Data augmentation GAN + Hourglass	91.5
2018 [124]	Hourglass	256x256	Structure aware loss and augmetation	92.1
2018 [125]	Hourglass	256x256	Residual siamese network	92.3
2019 [126]	HRNet	256x256	Maintaining high resolution	92.3
2019 [127]	Hourglass	256x256	Part based branching networks	92.7
2020 [128]	HRNet	256x256	Multi scale HRNet	-
2020 [129]	RSN-50	348x288	Residual Steps Network	93.0
2021 [130]	Lite-HRNet	256x256	Lightweight High-Resolution Network	-
2023 [131]	Transformer	256x192	ViT [132] based multi task network	93.2
2023 [133]	Swin-Base	256x256	Pose as Compositional Tokens	94.3

Table 2.2: Summary of **2D single person** human pose estimation methods. The column PCKh@0.5 reflects the method accuracy on the Max Planck Institute for Informatics (MPII) Human Pose testing set using the Percentage of Correct Keypoints. This will be elaborated later.

with Tompson et al. [116] that offered a technique that coupled convolutional networks with Markov Random Fields to tackle the problem. This, in turn, revealed the potential of this approach resulting in many 2D pose estimation methods from RGB images being powered with deep-learning.

A lot of different methods that were published in a very short period of time during 2016-2017 streamlined the effort towards 2D joint detectors based on neural networks. The stacked hourglass networks of Newell et al. [108], Convolutional Pose Machines [109] and OpenPose [110] are among the most popular works. Other cutting-edge research includes DeepCut [111] as well as the ones presented in [112–114] which provide the computer vision community with tools to robustly handle the 2D human pose estimation task.

2.1.1 Single Person 2D Pose Estimation

Scenes captured "in-the-wild" typically feature multiple persons. It is very common however to approach the human body estimation problem using a method that can only handle a single depicted person. Theoretically a method that can work by targeting a single person in an image can be iteratively applied to multiple cropped and/or resized regions of the image so that they also contain only one person and get individually serially tracked. Early methods that handled this person instantiation problem used face detectors [134] to identify regions of interest, these however where gradually replaced by upper-body detec-

tors [135] and eventually general detectors like You Only Look Once (YOLO) [74] which can identify the human form along with a multitude of other object classification categories. However, due to the possible occlusions and complexity of the task it is very hard to practically achieve good results on multiple person 2D pose estimation using single person 2D pose estimators, while typically performance also scales linearly, with larger numbers of subjects on screen. The work we present in this dissertation is also single person and thus similar and directly compatible with most single person 2D joint estimators.

Regression-based Methods

Regression based methods work by learning to map RGB images to body joint coordinates. As mentioned above, neural networks have dominated this class of methods in a boom inspired by AlexNet [73] leading to Deep Pose [107] the first modern technique to adapt a simple neural network architecture like AlexNet to the different domain of Human Pose Estimation paving the way for efficient end-to-end neural network estimation. All the body representation models (as seen in Figure 2.1 require discrete joint coordinates and thus surrounding heatmap values near a joint maxima may lead to reduced stability if they are ignored. To improve this Livizon et al. [118] proposed a soft-max neural network that can transform a neural network that is built from the ground up to support detection and effectively convert them to a differentiable regression neural network. This output data could be converted to numerical values both using standard interpolation techniques as well as supporting neural network layers that handle numerical coordinate regression via convolutional neural networks [136] to directly calculate joint coordinates from heatmaps. The difficulties arising from the task of direct joint coordinate calculation was also a core consideration of Carreira et.al. [113] that iteratively improved results from an initial mean pose to minimize error. Other works like Sun et al. [137] favor a compositional loss function to enable structure-aware regression that encodes long range interactions in the depicted pose. Table 2.2 shows a list of note-able methods along with their accuracy rating.

Detection-based Methods

Detection based methods began from early methods [138] that utilized sliding windows and image patches to roughly estimate unique patterns that corresponded to body parts. Neural networks enabled inference machines [139] that handled multiple classes using confidence maps, [140] low level feature learing for candidate patches of specific joints. This led to the very influencial [116] paper with separated heatmap channels for each joint that both handles occlusions better and with the 2D gaussian distribution centered around the true joint target enabled robust results an approach that has dominated research on the topic ever since. As seen in Table 2.2 detection based methods initially followed AlexNet and gradually advanced to more elaborate architectures from ResNet derivatives to stacked hourglass networks. Hourglass networks where also the network of choice for the implementation of adversarial networks [122] in the form of two stacked hourglass networks that controlled heatmap generation and discrimination. Recent state of the art methods include High-Resolution networks (HRNets) and the general consensus is that heatmap output is a superior output representation due to the redundancy of neighboring output areas that ensures stable training and output. At the same time heatmap output which is typically 64x64 pixels requires a small but fast dynamic programming post processing step to provide subpixel accuracy in a higher resolution. Although our method is compatible with any source of 2D joints, a particularly fitting detection-based 2D pose estimation method from the literature is BlazePose [34] that converts RGB Images to heatmaps but

also internally tracks and resolves the features in an end-to-end fashion via a Convolutional Neural Network (CNN) to directly yield 2D coordinates and visibility flags.

2.1.2 Multiple Person 2D Pose Estimation

Having reviewed recent state of the art literature on single person estimation we can continue on the more complex subject of multi-person pose estimation. The methods and frameworks that tackle the problem can be found summarized in Table 2.3 along with their respective accuracy which is encoded in the average precision (AP) column. Many of them share common design considerations as single-person methods however all of them are functionally superior since scenes encountered in realistic computer vision applications typically contain an arbitrary number of persons and multi-person methods include the capacity to handle (and sometimes track) all of the observed persons at the same time. Once again we will dissect methods by splitting them in two categories to better comprehend their design characteristics. A core distinguishing factor between multi-person 2D estimators is if they work directly on pixel data building up complex representations (bottom-up architecture) or if they work by first splitting different image regions that correspond to different persons and then localize joint hierarchies by assigning joints to different pixel locations (top-down architecture).

Top-down Methods

Top-down methods typically use an integrated person detection module to handle the problem of body instancing and then use a single-person estimator to derive each of the solutions required for each of the body instances observed, treating them as separate sub-problems. This affects both their computational performance that is inversely proportional to the number of observed humans, as well as their accuracy that depends on the detected bounding box and can be adversely affected by overlapping skeletons and occlusions. As seen in Table 2.3 most methods rely on existing neural network architectures that have been repurposed for the multi-user pose estimation application. Such neural networks include Faster RCNN [159] which is a very influencial work, Mask RCNNs [145], Feature pyramid networks [160] and Convolutional Pose Machines [141]. We observe that single-user methodologies (as seen in Table 2.2) have been directly applied after being modified to focus on different regions of the image.

Bottom-up Methods

In contrast to top-down architectures, bottom-up methods work by performing pattern matching on pixel data to detect unique color constellations that reveal the location of joints. The detected keypoints are then grouped in hierarchies to yield the final human skeletons observed in the scene. This grouping can be handled both as an optimization post-processing problem as well as be organically handled by the neural network with the invention of Part Affinity Fields(PAFs) [110] that use machine learning to encode the direction of joint hierarchies thus making it very easy to connect joints encountered in an RGB image to their respectful skeleton. Note-able methods in this category include Deep-Cut [111] that was one of the first methods to successfully partition observed joints based on their proximity and refined by Deeper-Cut [151] which used a better optimization strategy for joint candidates. OpenPose [110] was a major breakthrough in the field of 2D human pose estimation because of its accuracy, robustness, real-time performance and mature code-base. Part affinity fields [110] enabled natively regressed limb orientations that made combining the peaks of heatmaps for joints straightforward and fast. The improved version of OpenPose [4]

Year/Ref	Framework	Summary	AP(%)
Top-down methods			
2016 [141]	Faster RCNN/CPM	Linear Programming refinements	-
2017 [142]	Faster RCNN/Hourglass	Spatial transformer network	63.3*
2017 [143]	Faster RCNN/ResNet-101	Heatmaps and offset maps for each joint	64.9*
2017 [144]	Faster RCNN/Inception v2	Multi scale coarse-fine feature fusion	72.2*
2017 [145]	Mask RCNN	Simultaneous joint and mask derivation	63.1*
2018 [146]	Faster RCNN/ResNet	Extra deconv layers	73.7
2018 [147]	Feature Pyramid Net	Two-Stages with hard keypoint mining	73.0
2018 [147]	Feature Pyramid Net	Two-Stages with hard keypoint mining	73.0
2018 [147]	Feature Pyramid Net	Two-Stages with hard keypoint mining	73.0
2019 [148]	ResNet	Posefix, model-agnostic pose refinement	-
2019 [126]	Faster RCNN/HRNet	Multi-scale feature fusion	75.5
2020 [149]	Siamese Network	Graphical Representation	-
2022 [150]	HRNet	Structural Group Inference	79.2
Bottom-up methods			
2016 [151]	Fast RCNN	Deeper/faster multi person estimation	-
2016 [110]	VGG-19/CPM	OpenPose, part affinity fields	61.8*
2017 [152]	Hourglass	Joint detection and grouping	65.5
2018 [153]	Hourglass	Pose partitioning networks	-
2018 [154]	ResNet	Mult-Task pose estimation	68.7
2018 [155]	ResNet+RetinaNet	Pose residual network	69.6
2018 [4]	VGG-19/CPM	Increased depth, refined network	64.2*
2019 [156]	ResNet-50	Part intensity fields/part affinity fields	66.7
2019 [157]	FCOS	Fully Convolutional one-stage	44.7
2020 [158]	Hourglass	Body-part connectivity	68.1

Table 2.3: Summary of **2D multiple person** human pose estimation methods. The column AP reflects the Average Precision score on COCO test-dev. Results with asterisk use COCO16 while others COCO17.

refined the network offering real-time 2D body pose estimation while also adding supporting neural networks to accommodate 2D hand and face pose estimation at interactive framerates making it overall a great combination and the method of choice to provide 2D pose data for this PhD. Other methods like Part intensity fields combined with Part affinity fields [156] perform similarly albeit in lower resolution images, while methods like [152] have a different internal mechanism to handle joint associations and [154, 155] achieve slightly better accuracy while solving the association problem through separate processes.

2.2 3D Human Body Pose Estimation

Moving away from 2D body pose estimation that is a task that involves identifying unique regions of pixels in an image, we move to examine 3D pose estimation which is a higher-level task that moves away from the 2D image and also performs the more challenging step of deriving the third dimension that is lost during image capture since all colour intensities regardless of their distance from the observer are condensed or occluded in the 2D surface of the camera sensor. Multi-camera systems can record extra data that can act as mathematical constraints [161] given accurate matching of 2D points between different camera viewpoints that can facilitate 3D calculations. Systems like VICON [162] may also feature pre-calibrated multi camera

systems and motion capture suits featuring markers to further simplify and streamline the task. Active-light sensors like the original Kinect sensor with its infrared projector enabled the collection of 3D information at sensor level. Kinect 2 switched to time of flight sensors improving the 3D depth resolution at the cost of extra processing power required to synthesize the 3D data after capture but ultimately applications remained limited mainly due to the increased cost of sensors, their limited adoption and their technical limitations in outdoor locations. Due to all these reasons monocular RGB without the need for cumbersome motion capture suits that feature physical markers is the desirable target since as we will see with recent advancements can be achievable at low hardware, setup and operational costs. Once again, thus for 3D, we will focus on monocular RGB pose estimation methods.

We should keep in mind that 3D is used as a blanket term that often just denotes "more than two dimensions", however methods that just regress a depth component to 2D joints are functionally inferior to other methods (including ours) that not only regresses depth, but also solve the essential (for many applications) Inverse Kinematics (IK) problem. At the same time, however, the simpler task of just providing a depth measurement often means that purely 3D joint pose estimation without IK are typically more accurate in terms of 3D error. Recent trends, show the importance of high-level inference like the one we attempt. Translation and rotation invariant features [163], representations that handle discontinuities in rotations [164] and works that utilize different data mappings that better encode structural properties such as kinematic chain spaces [165] offer promising results. On the other hand advances and improvements on neural networks like dynamically routed neural networks of capsules [166] focused on equivariance by means of architectural changes.

2.2.1 3D Single Person Pose Estimation

Much like their 2D counterparts 3D body pose estimation methods can be categorized based on their similar design characteristics and commonalities. Table 2.4 shows an extensive list of recent 3D pose estimation methods that have defined the state of the art over the last 5 years. Accuracy of methods is examined by benchmarking them against the publicly available dataset of Human 3.6M [38]. Error is measured using the Mean Per Joint Position Error (MPJPE) in millimeters. There are multiple protocols to enable impartial comparison between different methods, with the most popular comparison protocols being Protocol 1 (seen in Table 2.4) that tests accuracy on test-sets using all available different camera viewpoints. 3D single person estimation methods can work both directly from pixel data or use a discrete single or multi-person 2D estimator (that we described in the previous Section) as a first stage. We will begin by categorizing methods based on the number of discrete stages and afterwards by categorizing them based on their utilized model type.

2.2.2 One-stage 3D Human Body Joints Estimation

Many works that use RGB images as input adopt a holistic approach to the pose estimation problem and infer 3D pose from 2D images in one step. For example, [200] adopts a Bayesian approach, Du et al. [201] defines an intermediate height map generation, Ghezelghieh et al. [202] has a smart camera logic, Rogez et al. [203] uses an image synthesis engine and an end-to-end CNN architecture and LCR-Net [204, 205] combines a pose proposal/classifier with a regressor. Some methods (e.g., DensePose [206], Omran et al. [193] with a combination of a DNN network and the SMPL [207] model) go a step further by also addressing the problem of human body shape estimation by calculating 3D mesh associations. Kanazawa et.al [192] model

Year/Ref	Framework	Summary	D	T	MPJPE
Model-free methods					
2014 [167]	CNN	Body part detection and joint 3D estimation	✗	✗	132.2*
2015 [168]	CNN	Structured output learning	✗	✗	120.2*
2016 [169]	Autoencoder	Autoencoders for 3D pose derivation	✗	✗	116.8*
2017 [170]	Hourglass	Two stage 3D pose regression	✓	✗	69.7
2017 [171]	CPM	3D estimation = 2D estimation + matching	✓	✗	82.7
2017 [172]	CPM	2D + Euclidean Distance Matrices for 3D	✓	✗	87.3
2017 [173]	Hourglass	Coarse-to-fine from 2D	✓	✗	71.9
2017 [174]	Hourglass	Loss from geometric constraints	✓	✗	64.9
2017 [175]	Hourglass	Direct map of 2D to 3D	✓	✗	62.9
2017 [137]	ResNet	Bone based representation	✓	✗	48.3
2018 [176]	Hourglass	Adversarial learning	✓	✗	58.6
2018 [177]	Hourglass	Volumetric representation	✓	✗	56.2
2018 [178]	Mask RCNN	integral pose unifies heatmap/joint regression	✓	✗	40.6
2019 [179]	Hourglass	Multiple hypothesis + mixture density net	✓	✗	52.7
2019 [180]	CNN	Heatmap for third dimension	✗	✗	39.9
2021 [181]	Transformer	Temporal+Spatial transformer mix	✗	✓	44.3
2022 [182]	STMO	Pretraining and finetuning network	✗	✓	42.1
2022 [183]	Transformer	Temporal+Spatial transformer mix	✗	✓	39.8
2023 [184]	Transformer	Two streams encode local relationships	✗	✓	38.4
Model-based methods					
2016 [185]	DeepCut	fit SMPL model to 2D joints	✗	✗	82.3
2016 [186]	ResNet	Kinematic model embedded on network	✗	✗	107.3
2017 [187]	ResNet	Real-time model based skeleton fitting	✓	✓	80.5
2017 [188]	ResNet	Kinematic Model from trained 2D features	✓	✗	74.1
2017 [189]	CNN	Silohouette from SMPL model	✗	✗	-
2017 [190]	LSTM	LSTM for joint depth estimation	✓	✗	79.5
2017 [191]	CPM	3D lifting with probabilistic model	✓	✗	88.4
2018 [192]	ResNet	Adversarial learning for SMPL model	✓	✗	88.0
2018 [177]	Hourglass	Two-stage prediction with SMPL model	✓	✗	75.9
2018 [193]	RefineNet	Volumetric inference using SMPL	✗	✗	59.9
2018 [194]	Hourglass	Two-stage prediction with SMPL model	✗	✗	49.0
2018 [195]	Hourglass	Geometry aware representation	✗	✗	-
2019 [196]	ResNet	Temporal context for 3D pose	✓	✓	77.8
2020 [197]	CNN	Teacher-Student architecture	✓	✓	97.3
2020 [198]	CNN	Neural network with kinematic priors	✓	✓	54.9
2023 [199]	Transformer	Dual stream spatio temporal transformer	✓	✓	37.5

Table 2.4: Summary of **3D single person** human pose estimation methods. The column MPJPE is the Mean per joint position error in millimeters. Entries with asterisk use the whole test dataset, while columns with asterisks only a subset. Column T indicates if temporal information is being leveraged to improve scores. Column D indicates if data outside of the H36M dataset are used for training. The leaderboard for the H36M performance is available online [54]. The recent trend in state of the art accuracy methods is methods using transformers using a history of up to 243 frames and training on extra data outside of H36M.

the human body using SMPL as well as camera information while also coupling a discriminator network. Chen et. al. [171] employ a 3D pose library and sets of virtual cameras. Tan et al. [189] use pose silhouettes to derive 3D human shape and pose. Other works are focused in acquiring 3D points from 2D images using either a single [173], or more cameras [208, 209]. Some methods use a purely convolutional approach to extract 3D pose like [194] that models 3D volume loss, while some others rely on separate algorithms [210] to perform optimization. Tenkin et.al. [170] uses both RGB to 2D as well as 2D to 3D learning. Elepose attempted direct 2D to 3D regression by utilizing PCA compression [76], while very recent versions utilizing transformers [211, 212] achieve state of the art offline accuracy that reaches 16.9mm in the case of [199].

2.2.3 Two-stage 3D Human Body Joints Estimation

Our approach falls to the so called two-stage method category since it separates pose estimation from pattern recognition and operates on 2D points uplifting them to 3D. One-stage methods have the merit of not relying on anything but their own training set. At the same time, this is one of their weaknesses, in the sense that generating an extensive and unbiased dataset is very difficult [174, 213]. Indicatively, there are works specializing in dataset generation using synthetic data coming from 3D models [214] or MOCAP data [203]. High-level data makes data augmentation a much easier task.

Within this method category, Bogo et. al [185] use the SMPL linear model [207] and regress both pose as well as the human shape. Other RGBD based methods [215] use a 3D model acquired using an RGBD camera and use Particle Swarm Optimization [216] as their optimization technique. VNect [217] also uses a two stage RGBD approach but with a generic skeleton model, while Li et al. [168] uses a similar concept albeit using RGB data, only. Certain methods utilize adversarial networks to formulate an inverse graphics problem [218]. Very recent works on RGBD data have reached a point of also accounting for garments [219].

The most similar work to ours is [172] that utilizes 2D EDMs as its input representation, although regressed to 3D EDMs and not to direct output angles like our work. Another recent paper that identifies the need for structure-aware regression is [137] although their joint connection representation is less rich than our NSDM formulation [6]. Our approach shares the compact formulation of [175] although on a much more shallow network without residual connections. Similar approaches include [220] but instead of silhouettes we use the numerical values of 2D points and retrieve results like [221] without physics simulations.

Human pose completion in partial body camera shots A Conditional Generative Adversarial Network is used to obtain a complete and plausible pose, realistic enough to predict a 3D pose with a two-stage 3D human pose estimation method. Research focusing on actual images contain partial body camera shots [222]

Model-free Methods

Model free methods produce 3D pose output without relying on a human body model as an intermediate representation. The advent of convolutional neural networks has enabled methods that can perform direct 3D estimations using simple neural networks like [167] which directly regressed 3D joints. Li et al. [168] designed an embedding latent pose structure information network to guide 3D joint mapping. Tekin et al [169, 223–225] created a variety of methods that treated the problem utilizing autoencoders while also utilizing their prior work on 2D pose estimation. Chen et al. [171] learned 2D to 3D matching and Moreno et al [172] encoded 2D poses in Euclidean Distance Matrices (EDMs).

Model-based Methods

Model based methods on the other hand rely on a parametric body model to use as a means to perform pose comparison. Skinned multi-person linear model [207] is the dominating model used by most model-based works. Bogo et al. [185] fit it to 2D joints to recover SMPL parameters. Tan et al [189] use a neural network that acts as a decoder to SMPL parameters from RGB images. Kanazawa et al. [192] use adversarial networks to both generate SMPL parameters and distinguish correct guesses from the erroneous ones. Other methods use different models however like the kinematic model of [188], the LSTM based method of [190] also used a custom model while Zhou et al. [226] embedded a kinematic model into a neural network to also retreive orienatation and rotational constrains.

Year/Ref	Framework	Summary
2017 [204]	Faster R-CNN + VGG-16	Regress pose based on localization classification
2018 [230]	ResNet	Occlusion robust pose maps for full body pose inference
2018 [227]	DMHS [228]	Feed forward process of body parts using SMPL model
2019 [229]	CNN	Network with skip connections for kinematic skeleton fitting
2020 [231]	Mask-RCNN	Interpenetration loss/Depth ordering loss
2020 [232]	CNN	Part fields from RGBD input
2020 [149]	Siamese Network	Graphical Representation
2021 [233]	HRNet	Spatio-Temporal estimation
2021 [234]	Auto-encoders	Multi-step architecture with depth estimation
2022 [235]	CNN	Multi-view Mode seeking in the space of skeletons
2023 [236]	CNN	Multi-view 2D-bounding boxes and poses to 3D trajectories

Table 2.5: Summary of **3D multiple person** human pose estimation methods.

2.2.4 3D Multiple Person Pose Estimation

An emerging category of pose estimators is the 3D multi-person frameworks. A summary of the few existing methods can be found in Table 2.5. Rogez et al. proposed LCR-Net [204] separated the pose estimation task to localization then classification and finaly regression, using faster R-CNN to detect persons, a 2D classifier to detect joints and finally a network to regress to 3D joints. Zanfir et al. [227] proposed a method based on Deep Multitask architecture for Human Sensing (DMHS) [228] using both feed forward stages to perform semantic segmentation as well as feed backward stages for the 3D pose estimation. Mehta [187,229] proposed a method that uses a convolutional neural network that selectively skips connections in large and small steps by infering 2D pose and refining it after for stability and temporal coherency.

2.3 3D Hand Pose estimation

Notable field milestones include [237] that utilized GPGPU acceleration and the Kinect RGBD sensor to tackle 3D hand tracking in real-time using Particle Swarm Optimization (PSO) [238]. RGBD methods dominated the field and algorithmic strategies where mainly divided to model-based generative approaches and data-driven discriminative ones. Our method uses both concepts thus falling in the “hybrid” category. Noteable generative optimization algorithms are PSO [238], Iterative Closest Point (ICP) [80], Levenberg Marquardt [239] and Gauss-Newton [240] among others, while discriminative algorithms include Support Vector Machines [241] and Random Forests [242]. Depth data allowed many 3D hand tracking methods to utilize nearest object or blob segmentation [243–245] while others utilized RGB skin detection [246, 247] or markers [248] to a similar end. The advent of NNs caused a transition to RGB that for hands started taking place in 2014 after the influential work of Tompson et al. [249] that presented a convolutional neural network (CNN) for 2D joint localization leading to methods like Zimmermann et al. [41] that tackled 3D hand pose estimation from RGB paving the way for recent NN methods [250–253] that continuously improve the state of the art.

“Two Heads Are Better than One” [254] utilizes a 2D CNN to extract visual representations in 2D image space and performs iterative corrections in 3D point cloud space to exploit the 3D geometry information of depth data, while [255] is a semi-supervised method based on Multi-Task and Multi-View Consistency (MTMVC) metric for hand pose estimation. HandMIM [256] uses a Vision Transformer (ViT) as the back-

bone and attach a Py-MAF head after tokens for regression, similarly Deformer [257] uses a dynamic fusion transformer achieving robust hand pose estimation.

2.4 3D Head Pose estimation

A recent survey [258] summarizes 3D head pose estimation research. The problem is long standing with early approaches using RGBD cameras and stochastic optimization [259]. The advent of neural networks yielded seminal works like Liu et al. [260] that regress roll, pitch and yaw in 1 stage using convolutional neural networks (CNNs), while methods like [261] using 2D landmark-based face alignment to improve accuracy. Our method has conceptual similarities to ASMNET [262] which is a 2-stage method that uses a statistical shape model to match 2D observations to facial structure and thus regress its pose.

2.5 3D Gaze estimation

Gaze estimation methods focus primarily on the eyes and typically employ head mounted RGB cameras. A recent survey [263] provides a detailed study of the topic. Influential 1-stage methods include ITracker [264] pairing a CNN with an RNN for temporal refinement, MPIIGaze [265] using a CNN to regress 3D gaze from a single eye image, achieving high accuracy and robustness to variations in head poses and illumination conditions. Our method, however, falls in the 2-stage category with a notable method following the same approach being EyeNet [266] that operates on infrared images of the eye from a head mounted camera. Methods that also handle head pose include [267] for full views of the body far from the camera. RT-GENE [268] achieves real-time performance and accurate gaze estimation in natural environments. The method features eyetracking glasses for recording ground-truth and a smart inpainting GAN solution to provide realistic training samples. A new class of heavier offline algorithms use Generative Adversarial Networks (GANs) like GazeDirector [269], head2head [270, 271] and implicitly deal with gaze detection by modifying it at will while also maintaining high fidelity video output.

2.6 3D Facial Capture

A recent monocular facial capture survey is offered by Zollhofer et.al. [272]. After a period where RGBD cameras were frequently used for 3D facial capture [273], there is now a recent push with methods using Generative Adversarial Network (GAN)s [274] and/or Diffusion Models [275], that exploit their ability for texture completion and extract a 3D morphable facial model along with texture, UV maps and BRDF maps [276]. Methods like EMOCA [277] employ very elaborate architectures with differentiable renderers and provisions like “Emotional Recognition” and “Emotional Consistency Losses”. Other methods employ the same high quality model while also integrating a large number of frames from a video source to further improve visual fidelity [278]. Neural Radiance Fields have recently been coupled with morphable face models [279]. These methods perform very dense and high quality facial capture which is not at all possible in our case since our method just animates a sparse BVH skeleton which by design offers a more limited expressional output. Our method is much more light-weight in a manner similar to [280], however instead of regressing a facial tessellated mesh we employ purely rotational BVH configurations for the face. This is a much higher level representation that does not encode appearance at all, however it makes calculations (about e.g. the gaze, how open are the eyes, or the mouth, eyebrow tilt etc.) very accessible to potential applications since they can be directly accessed as Euler angles floating point numbers. Finally borrowing the design of HRNets an

adapted version targeted on the facial problem was published during 2023 [281].

2.7 3D Human Modeling

For model based human pose estimation algorithms, the computer model that is used to represent the human body plays a crucial role in method accuracy. Generative methods especially, rely heavily on the richness of the model and how close it resembles the human anatomy and actual degrees of freedom of the human body. Even the most elaborate optimization method cannot overcome model limitations that typically arise from a difference in body dimensions, as well as the loss of degrees of freedom of the body due to its over-simplification to make it computationally feasible for usage in real applications. Figure 2.2 shows the human body with all its bones and muscles. Of course the actual dimensions of every detail of the body are different from person to person. Computer representations typically use models that follow the general concepts illustrated in Figure 2.1. As we can see they are much simpler and fall in three main categories, skeleton models, contour models and volumetric models.

2.7.1 3D Human Body Models

Skeleton models also referred to as stick-figure or kinematic models consist of a joint hierarchy graph with nodes that typically cover the most important topologies of the human body, i.e. arms, legs and head. This representation is compact, flexible, easy and fast to compute and powerful enough to facilitate advanced tracking. Commonly used joint hierarchies include the COCO [110] or BODY25 [4] topologies. Drawbacks of this body model is the lack of contour and volume information about the body, as well as the lack of texture and external appearance of the depicted human. Joints located at places where the body is thin and unique patterns are formed like the areas around eyes fingers, elbows, hands, knees and feet are robustly encoded using this pose representation. However for joints that are in thick parts of the body or have no visible landmarks especially given the variety of clothes and external appearance due to the aforementioned shortcomings lead to poor results representing the human spine that is completely ignored in methods like [4, 110].

Contour-based models are used in older techniques and better encode the shape width and height of bodies. Contour models coincide with features generated from digital color images using very simple first-derivative Sobel edge or corner detectors algorithms [282] that have been proposed over 3 decades ago and are widely available in open source computer vision libraries like OpenCV. Popular contour-based models can include cardboard models [283] and active shape models [284], while contemporary 3D GPUs allow rendering of more complex underlying 3D models and extraction of the contours without loss of performance.

Volumetric models began as a combination of simple geometric 3D primitives like cylinders, cones, spheres and cubes stitched together to form a rough human body [285]. Gradually more advanced models where proposed like Shape Completion and Animation of People (SCAPE) [286] which combined with RGBD sensors led to methods that can perform automatic 3D model capture [287, 288]. Other prevalent models include Skinned Multi-Person Linear model (SMPL) [207], unified deformation models [289], GHUM/GHUML [290] as well as open-source projects like Makehuman [23] that aim at realistic configurable and accurate 3D armatures that can be used both for rendering photo-realistic scenes as well as a tool to generate images for deep-learning.

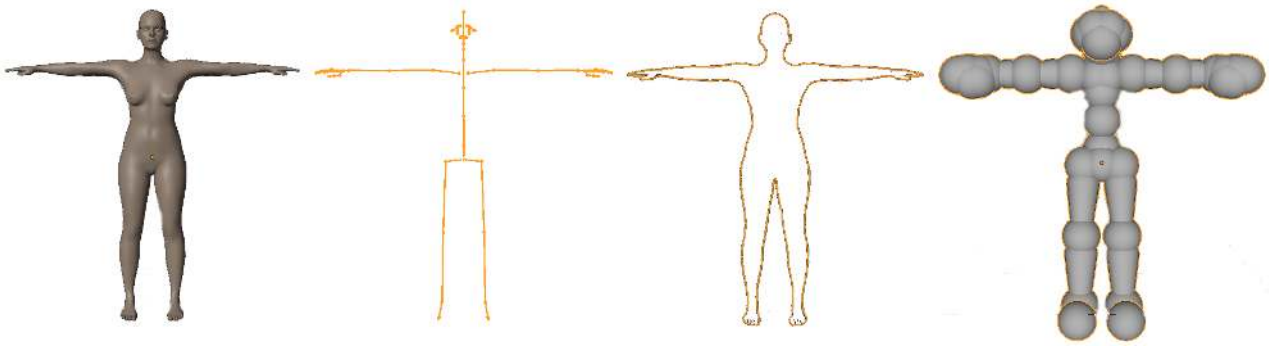


Figure 2.1: Model types that are typically used in model-based body pose estimators. Starting from left, (1) a skinned 3D mesh body model, (2) a skeleton based joint hierarchy, (3) a contour based model and (4) a volume-based model. Our method uses a BVH skeleton based joint hierarchy for the body and hands, a skinned 3D mesh facial model for the face, and a volume based model for occlusion handling during dataset generation.

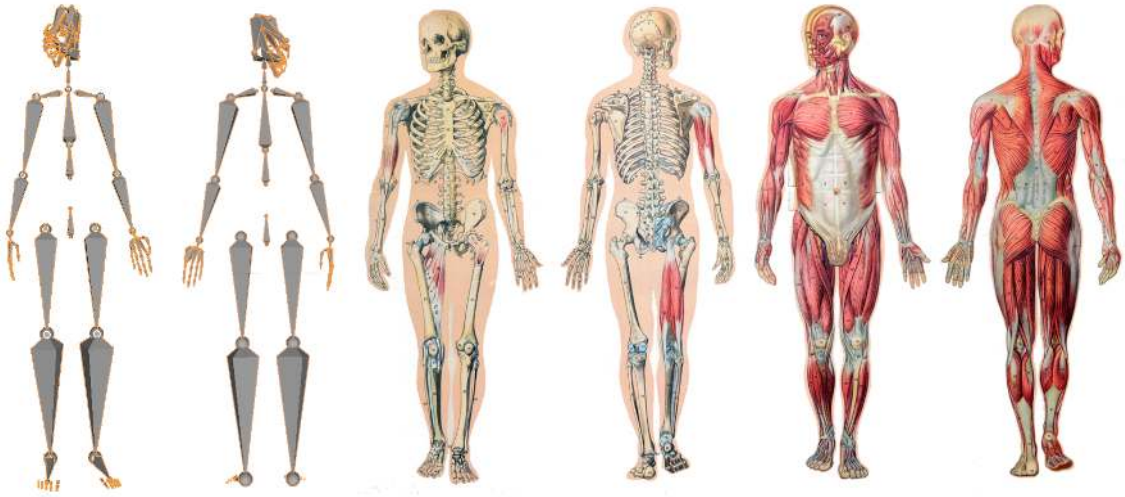


Figure 2.2: Illustration of our BVH armature vs the human body anatomy with all its bones and muscles. An ideal computer human body model should be able to reflect all of this complexity, however even the most elaborate 3D human models used today (Figure 2.1) still operate on a much more simplified representation that only covers the basic degrees of freedom of motion. Illustration of human anatomy from Encyclopedia Britannica [1].

2.7.2 3D Human Hand Models

3D output models employed by methods exhibit incredible variety, since most use their own internal models. Popular common choices are MANO [291] for hands and SMPL [207] for bodies. Some use combinations like [36] with SMPL + frankenstein hand model [289]. Our method uses the BVH [77] open standard

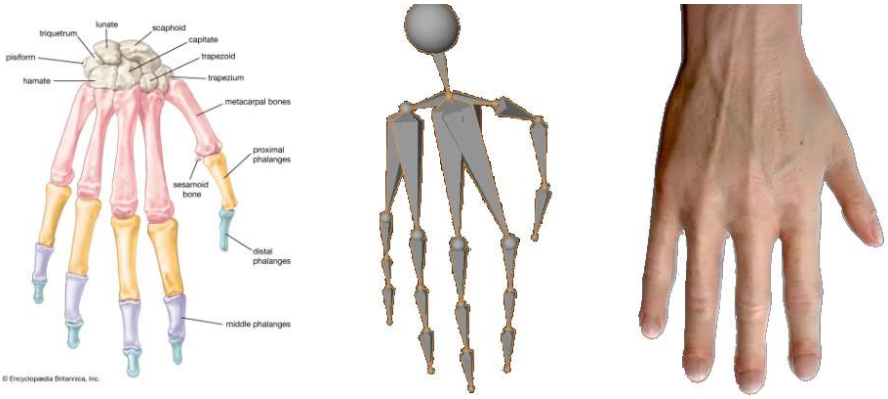


Figure 2.3: Left: Illustration of human hand bone anatomy from Encyclopedia Britannica [1]. Middle: Our BVH hand armature resembles the dimensions and range of motion of real hands. Right: Realistic rendering of a human hand using a high-resolution mode featuring tendons, veins and skin.

with a motionbuilder [292] armature. We can thus render our results using popular open-source tools like Blender [21] and MakeHuman [23].

Figure 2.4: Illustration of our sparse BVH facial model, next to a human face and its underlying anatomy. Even the most elaborate 3D human models currently operate on a much simpler version that only covers the basic degrees of freedom of motion and do not reflect all of the facial complexity. Illustration from Encyclopedia Britannica [1].

Figure 2.5: From left to right. We use a BVH skeleton (1) that is a sparse representation of the human bone and muscle structure. Combining however this BVH file with a skinned 3D model, like Makehuman (2,3,4) we can render photorealistic humans thus recovering higher fidelity output.

2.7.3 3D Human Face Models

Several 2D and 3D facial models have been proposed in the literature, offering various degrees of detail and complexity. A Facial 3D model survey [293] offers a comprehensive list of such models. Out of them two stand out and have been used in many subsequent works, the 3DMM [294] and the FLAME model [277]. The FLAME model combined with the SMPL model makes the SMPL-X [60] full body model which is commonly used in most Total Capture Methods in the literature that we will examine in the next Section. 2D Facial models are also important and typically dictated by the popularity of facial databases and their ground truth annotation error. Very sparse facial models make annotation more accurate and inference during runtime faster while more complex facial models have the opposite effect. Influential 2D models include XM2VTS [295] and Multi-Pie/IBUG [5] model which is compatible with OpenPose [296], very commonly used in the literature. For that reason we adopt IBUG and go into details regarding its keypoints in Chap-

Year	Name	Samples	Actions	Subjects	Cameras	RGB	RGBD	3D	Papers
2014	**Human3.6M** [39]	3.6M	17	11	4	✓	✓	✓	623
2018	3DPW [297]	51000	60	18	1	✓	✗	✓	261
2017	MPI-INF-3DHP [188]	1.3M	8	8	14	✓	✗	✓	211
2018	DensePose-COCO [298]	50000	-	-	-	✓	✗	✓	195
2019	AMASS [299]	$\approx 8M$	-	344	-	✓	✗	✓	193
2010	**Leeds Sports Pose** [44]	2000	-	-	-	✓	✗	✗	186
2015	Panoptic [300]	1.5M	65	-	520	✓	✓	✓	89
2020	LaFAN1 [301]	0.49M	15	5	BVH	✗	✗	✓	4
2003	**CMU Mocap** [18]	3.9M	473	144	BVH	✗	✗	✓	-

Table 2.6: Summary of the most relevant 3D human pose estimation datasets for our work. Fields marked with a dash mean a mixture that makes categorization difficult. For example in Leeds Sports Pose [44] every sample features a different subject, different cameras with different intrinsics and different actions, although all in sport contexts. We mark BVH datasets with the corresponding label in the Camera column to denote that given the BVH armature we can simulate any camera we want. We mark datasets we utilize in this thesis for body pose estimation with a bold font.

ter 3. Our 3D BVH output results can be used to re-animate a virtual skinned avatar of arbitrary appearance. We inherit the BVH armature of the baseline body pose estimation method [9] and use MakeHuman [23] making our work integrateable with open-source community developed tools.

2.8 3D Human Datasets

Datasets play a crucial role in developing and benchmarking 3D Human Pose estimation methods. The quality, variety and biases of recorded datasets and annotations make or break methods that rely on fully-supervised data to be trained. This importance is reflected by the variety of datasets proposed and their individual contributions focusing on specific views and motions of the problem.

2.8.1 Body Pose Datasets

There are many datasets like PosePrior [302] that focuses on extreme poses at the physical angle limits of the body of actors. Datasets like AIST++ [303] feature dancing motions paired with music, EgoCap contains egocentric images [304], IniniteForm [305] is a synthetic dataset for fitness and physical therapy applications, LAAS Parkour [306] that focuses on Parkour, DeepFashion [307] that focuses on the garments worn by subjects and Geometric Pose Affordance [308] which features people interacting with 3D environments.

The bulk of human pose estimation works however typically target less exotic activities and the current need of NN training for large sample sizes makes them a more compelling choice. We present a list of the most relevant datasets for 3D human pose estimation in the context of our research in Table 2.6.

Out of them, the most influencial dataset in the literature is by far the Human 3.6M [39] which is currently the de facto benchmark for 3D pose estimation methods. Its popularity is the reason we adopt it for quantitative experiments in order to be able to compare MocapNETs to other algorithms. The 3D Poses in the Wild dataset (3DPW) [297] was the first dataset in the wild with 3D poses targeting outdoor activities and large recording volumes while also was the first to include video footage taken from a moving phone camera which is a particularly ubiquitous format due to the popularity of smartphones and social

media. MPI-INF-3DHP [188] consists of both indoor and outdoor scenes recorded from 14 camera views. DensePose-COCO [298] was created by running DensePose on 50K COCO images and recording UV 3D coordinates within every human region of each image. AMASS [299] is currently the large database of 3D human motion unifying different optical marker-based motion capture datasets and representing them under a common framework. Although it appeared after our initial publication and thus we did not use it in the course of this PhD we predict AMASS being a very influential source of data in the future due to its extent. The Leeds Sports Pose (LSP) [44] dataset is widely used as a human pose estimation benchmark in athletic activities. We used it for qualitative results as a means to test our method on exotic poses outside of the distribution of our training dataset. CMU Panoptic [300] is a dataset recorded using the Panoptic sphere of Carnegie Mellon University. The impressive setup features a spherical room with 480 cameras, 30 HD cameras, 10RGBD cameras and 5 DLP projectors all of which are synchronized and calibrated. This is currently one of the most hi-end setups to capture 3D pose data in a massively multi-view way.

Recording and hosting professional grade datasets using high precision hardware is difficult and expensive. This is the reason why each of the datasets mentioned above has its own proprietary licenses that can be problematic for applications. An important consideration building our method was to make it commercially viable by not compromising our methods potential applications by relying on proprietary training data. We thus avoided using proprietary datasets for training but rather only to test and compare our method to others for research purposes.

We found the perfect training set for our MocapNET Human Pose Estimation method using the permissive BVH conversion [16] of the also permissive CMU Mocap [18] dataset. Since our method is 2-staged it does not require RGB images and thus BVH files rendered through a virtual camera can provide us with a very rich source of training data that is very easy to augment and modify for good training results. Shortly after our first publication [6], Ubisoft released the LaFAN1 [301] BVH dataset, which features more refined actions and that is already being used as a training/test set by recent methods like [309–311].

2.8.2 Hand Pose Datasets

3D Hand Pose Estimation is a difficult problem that has been long approached using bio-mechanimal kinematic constraints and generative methods that did not require extensive training data. The advent of neural networks however changed this highlighting the need for datasets that standardized training and testing for 3D hand pose estimation methods. Unique requirements in comparison to body pose estimation datasets are higher capture resolutions with lower noise, since hands occupy a much smaller area of an image sensor and thus clear observations are needed. Another consideration is that due to the hands lying on the end effectors of hands they typically exhibit much higher velocity that combined with a low capture framerate can create motion-blur that makes 3D Pose estimation very difficult. Thus typically a static camera is used with hands that move relatively slow in order to yield clear ground truth. Table 2.7 provides a reference to popular 3D Hand Datasets.

Big Hand [313] is a large-scale 21-joint hand pose dataset, partly generated synthetically and partly collected using depth sensors and a tracking system with six 6D magnetic sensors. SynthHand [314] is a synthetic dataset that provides 3D samples virtually rendered using Unity after tracking a real hand. In this way it combines ground truth accuracy with authentic hand poses that exhibit realistic patterns. Rendered Hand Pose (RHD) [315] is a 3D rendered synthetic dataset. It provides 41K samples featuring segmentation maps and 21 keypoints per hand. One Hand 10K [317] real hand dataset was created to facilitate the training of the Neural Network presented in [317] focusing on accurate annotation and realistic appearance. Stereo

Year	Name	Number Of Samples	Bounding Box	Keypoints
2016	**STB** [312]	18K	✓	✓
2017	Big Hand [313]	2.2M	✗	✓
2017	Synth Hand [314]	63K	✓	✓
2017	**RHD** [315]	41K	✓	✓
2017	Panoptic [316]	15K	✓	✓
2018	One Hand 10K [317]	10K	✓	✓
2018	GANerated [318]	330K	✗	✓
2019	FreiHand [319]	130K	✗	✓
2019	MHP [320]	80K	✓	✓
2020	InterHand [250]	2.6M	✓	✓

Table 2.7: Summary of popular 3D hand point estimation datasets. We mark with bold the datasets we employed to quantitatively test our method.

Hand Pose Benchmark (STB) [312] recorded real hands using a stereo/depth camera resulting in 18K R/L image pairs with 18K depth images. GANerated [318] is a cleverly constructed synthetic dataset that uses 3D rendered ground truth that is converted to a realistic image using GANs

Due to a lack of BVH based groudtruth since to the best of our knowledge, our method was the first to attempt BVH hand regression as well as the comparatively straightforward limitations of finger movements that could be used for semi-supervised learning and for licensing reasons already mentioned in the previous Section we selected the RHD and STB datasets for quantitative experiments to study our method's accuracy on the 3D hand pose estimation task. InterHand2.6M [250] dataset is currently the largest dataset with real single and interacting hands.

2.8.3 Head and Facial Pose Datasets

Table 2.8 provides a list of popular datasets used in conjunction with head pose, 2D/3D landmark estimation and 3D gaze estimation. The annotated Facial Landmarks in the Wild (AFLW) [321] dataset offers a collection of 25K annotated face images with 21 landmarks from Flickr that exhibiting variety in appearance, lighting and environmental conditions. The Caltech Occluded Faces in the Wild (COFW) [322] dataset is focused to capturing large variations in shape, occlusions, pose, expression, use of accessories such as sunglasses and hats and interactions with objects (e.g. food, hands, microphones). The Columbia Imaging and Vision Laboratory Gaze Dataset (CAVE) [323] features accurate gaze ground truth for 5880 RGB images depicting 56 people of ethnically diverse background, 21 out of which wear glass with 5 head poses, 21 gaze directions per pose. The 300W [45] dataset consists of 300 Indoor and 300 Outdoor in-the-wild images. It covers a large variation of identity, expression, illumination conditions, pose, occlusion and face size. The images were downloaded from google.com by making queries such as "party", "conference", "protests", "football" and "celebrities". The Wider Facial Landmarks in the Wild (WFLW) [324] database contains 10K faces with 2500 reserved for testing and the rest for training and 98 annotated landmarks. The database features rich attribute annotations in terms of occlusion, head pose, make-up, illumination, blur and expressions. AFLW2000-3D [325] is a dataset of 2000 diverse and often hard to be detected by CNN images that have been annotated with image registered 68-point 3D facial landmarks. Due to the very demanding nature of facial capture, since slight variations in expression are very perceptible by humans, sparse keypoint anno-

Year	Name	Number Of Samples	Number Of Keypoints	Has Iris
2003	**XM2VTS** [329]	2360	68	✗
2010	**FRGC** [330]	950	68	✗
2011	BIWI [326]	15K	3D	✗
2011	AFLW [321]	25K	19	✗
2012	**AFW** [331]	337	68	✗
2013	**LPFW** [332]	1035	68	✗
2013	**IBUG** [333]	135	68	✗
2013	**HELEN** [334]	2330	68	✗
2013	COFW [322]	1852	29	✗
2013	**CAVE** [323]	5880	-	✓
2016	**300W** [45]	3837	68	✗
2016	AFLW2000-3D [325]	2000	68	✗
2016	3DFAW [335]	23K	66	✗
2017	ICT-3DHP [328]	14K	3D	✗
2018	WFLW [324]	10K	98	✓
2021	HPD [327]	23.5K	3D	✓

Table 2.8: Summary of **facial/gaze** datasets. Datasets marked as 3D provide a dense depth map and thus 3D ground truth for every image point. Datasets we use for experimental evaluation of our method are marked with bold.

tations are often not enough for methods that target high fidelity output. Instead the recent trend seems to be using depth cameras for a dense 3D reconstruction of the face and its expression. An early dataset on this direction was the BIWI [326] Kinect dataset. It contained 15K images of 20 people of RGBD frames with head poses covering +-75 degrees yaw and +-60 degrees pitch and ground truth provided in the form of the 3D location of the head and its rotation. HPD (Head-Pose Detection) [327] was generated using Blender, while ICT-3DHP [328] is collected using the Microsoft Kinect sensor and contains 14K RGBD frames.

Due to our method regressing a BVH skeleton that only features a sparse 3D armature, picking datasets to quantitatively assess our method was hard. Using an intermediate Makehuman [23] 3D skinned model renderd using Blender [21] we where able to create realistic human images that allowed us to have comparable 2D facial landmarks with common methods. We use a combination of AFW [331], FRGC [330], LPFW [332], HELEN [334], IBUG [333] and XM2VTS [329] dataset 68 point conversion re-annotation by [336] for landmark accuracy assesments and CAVE [43] for 3D gaze estimation benchmarking.

2.9 3D Total Capture Methods

Our brief literature review thus far provided a review on the topics of Human Body, Hand and Facial 3D estimation. All of the hierarchies of the human body share commonalities, namely the human body is a very elaborate 3D articulated structure with a multitude of bones, muscles, tendons and connective tissues. Thus describing all of the 3D transformations that take place for a specific pose instance at once should mathematically be feasible albeit very complicated due to the massive dimensionality of the problem. Very early Computer Vision research like the 1994 work of Rehg and Kanade "Visual Tracking of High DOF Articulated Structures: an Application to Human Hand Tracking" [79] described the problem of high d.o.f articulated structures recognizing this, however only focusing on hand pose estimation [79] due to tech-

nological limitations of the time. Similar limitations persisted for decades forcing researchers to focus on singular sub-hierarchies of the body to constrain complexity and in order to rigorously study and address them. This however gradually lead to a relatively fractured research landscape that did not consider the totality of the body pose estimation problem. Our research started with the goal of providing a unified solution to all these problems since at the start of this PhD there where no methods that successfully tackled Total 3D capture from RGB. In line with the research goals set in this PhD and the remarkable advancements powered by NNs, such as the release of Image/Video/Speech Generative AI, Large Language Models, the previously distinct areas of study in 3D human perception are converging and yielding multiple Total Capture methods [36,60–64,83]. The field of computer vision research is witnessing a trend of integrating diverse concepts under a unified subsymbolic knowledge base [337]. We are also very proud since our method formulation takes advantage of properties like symmetries, addresses many important aspects of the problem like occlusions and dimensionality and thus appears to be computationally superior to the currently available literature (Table 1.1) due to its sparsity and flexibility.

An early work in the context of Total Capture was [338] focusing on Nonverbal Communication Computing and reviewing earlier motion analysis methods employed in this area. It included a heterogeneous suite of standalone face tracker, expression recognition, body reconstruction, gathering their results to facilitate Human Computer Interaction (HCI).

OpenPose [296] was the first real-time multi-person system to jointly detect human body, hand, facial, and foot 2D keypoints (in total 135 keypoints) on single images. This very influential work has been cited over 10000 times showing its monumental impact.

Inspired by it, we set out to build a complementary 2D to 3D uplifting method. Our first proof of concept baseline was formed during the first semester of PhD studies in 2019 targeting the body [6] featuring very low accuracy but very high computational performance and promise due to its formulation. During the same year, the first total capture methods appeared [36, 60]. They were offline and regressed a coarse 3D facial expression without explicit 3D gaze estimation. These first methods did not offer quantitative results on the task of facial capture due to their main focus being on the torso and limbs.

During 2020 we improved our formulation greatly improving its accuracy [8] and occlusion tolerance. Due to the COVID-19 pandemic lock-downs research conferences where affected globally leading to our publications taking place in 2021 as well as our next milestone that also incorporated 3D hand pose estimation [9]. These coincided with the first real-time method for 3D monocular total capture appeared [61], once again using a combination of the SMPLH body model with the 3DMM [294] head model. In 2021 two more Total Capture methods appeared [62, 83]. Attempting to maintain our method's computational advantage while also expanding it to accommodate more hierarchies meant turning our efforts during 2022 into compressing ensembles [20]. Notable total capture methods released that year included ZoomNAS [63]. The final piece of the puzzle to our Total Capture Method part targeted 3D head pose, gaze and facial capture [31]. A recent work focusing on sign language Reconstructing Signing Avatars From Video Using Linguistic Priors [64].

Chapter 3
Our method and contributions

3.1 Introduction and Overview

Developing methods to tackle Computer Vision tasks can be approached from either a **top-down** or **bottom-up** perspective. The advent of NN methods, particularly after the groundbreaking AlexNet paper [73], in conjunction with the increasingly robust and mature Deep Learning frameworks such as PyTorch and Tensorflow and the appearance of General Purpose Graphics Processors (GPGPUs), marked a revolution. Older state-of-the-art methods that relied on intricate mathematical formulations and complex implementations were outperformed by a few hundred lines of code combined with well-curated training data. At the same time these advancements also amplified the trend of new methods that inherited a newly proposed state of the art NN method, (often from an open-source github repository) and then deepening and tweaking it to further climb accuracy records. We consider this the "top-down" approach to research, starting from a very high basis and then trying to identify and correct problems to refine it. The antithesis of this development paradigm are methods built from scratch using theory and first principles. "Bottom-up" development carries a great risk (since when starting from scratch, there are no guarantees of success and thus a very high probability of failure) but also great rewards when the research does yield results. Our method falls in the "bottom-up" category as it methodological is very succinct, at the same time the network itself was gradually refined using a "top-down" approach. It is indicative that our initial NNs published in 2019 [6] featured was a simple multi-layer, fully connected feed-forward network just 4 hidden layers (the original perceptron [339] featured 2).

Another categorization (initially proposed by George R.R. Martin, the famous writer), that I believe, is applicable to PhD research is the distinction between **architects** and **gardeners**. Quoting Martin: *" The architects plan everything ahead of time, like an architect building a house. They know how many rooms are going to be in the house, what kind of roof they're going to have, where the wires are going to run, what kind of plumbing there's going to be. They have the whole thing designed and blueprinted out before they even nail the first board up. The gardeners dig a hole, drop in a seed and water it. They kind of know what seed it is ... But as the plant comes up and they water it, they don't know how many branches it's going to have, they find out as it grows."* The presented method went through many discrete versions until it reached the form presented in this thesis thus adhering to the "Gardener" development paradigm. At the same time, the goal of the project was always Total 3D Capture, while the formulation (2-stages, BVH container, Ensemble Architecture, divide and conquer approach) has been "Architecturally" constant since the first conception of the method. The main issue with a purely architectural PhD research process is that peer-reviewed publications impose range limits on the extent of the presented research. This happens both in terms of the available space

on conference papers that are typically 8 or 9 pages, but also in terms of the time required to rigorously experimentally validate the presented work. Another complication is that performing the presented work during a booming era for NN and Computer Vision research meant fierce world-wide competition from incredible research teams from all over the world that often have an advantage of decades of research in the particular topic. Furthermore training and evaluating Neural Networks is very time-consuming, costly and requires expensive hardware thus presenting additional difficulties when trying to tackle a very large problem at once. Although since early 2019 it was clear that MocapNET could be applied for total capture the extent of work required to formally make this claim and provide sufficient experiments was 5 publications and [6, 8, 9, 20, 31] 4 years, to be properly materialized peer-reviewed and presented. At the same time multiple other methods appeared [36, 60, 61, 63, 64, 83] so following a different publication paradigm would only cause the performed research to lose novelty due to delays in its publication cycle and not due to lack of actual research merit. It is very interesting that the Gardener VS Architect dilemma is also applicable and permeates the way we can narrate the various design choices of the presented work. Doing so chronologically is easier to the reader since the reasons for the various additions becomes conceptually apparent as we "pour water on the seed of the initial idea", however this inadvertently causes various concepts to lose clarity, something that is counter to the goal of a thesis. Presenting details provided in clearly separated thematic Sections and on a per-contribution basis on the other hand has its own pros and cons. In order to strike a good balance between these two forms of exposition we will first address each methodological contribution on its own using the "Architect" paradigm and then we will try to provide a historical "Gardener" account of how these choices came to be on a per contribution basis.

Coming from a computer vision background predating NNs the attitude towards them was always using them in moderation and with caution. Ideally, (and if they where not so unreasonably effective), it would even be preferable not to consider them at all due to their black-box aspects. Thankfully during the studies for this PhD and due to their relative decoupled use due to our considerations we where partially able to illuminate the black-box something that we will try to communicate in the following Sections of this Chapter.

3.2 Creating a 2D Descriptor for 3D Articulated Shapes

The idea for this thesis (direct BVH MOCAP data regression from RGB videos) came after the WACV 2018 publication [215] during the end of my MSc studies and shortly before starting this PhD. Captivated by the, then recent, very impressive 3D pose estimation results of Neural Network powered methods and at the same time studying how they work, one of the first attempts was building a simple classifier that could estimate if a list of body 2D joint points was front or back facing. The rationalization for this task was that it should be feasible even with a very small network since, due to the left and right points essentially swapping their magnitudes a linear (or even better in the case of ReLU activation non-linear) combination of the values for the R/L Hip and R/L shoulder should produce a product magnitude that would somehow reflect the orientation of the tracked subject. In a similar fashion even a simple network like a multi-layer feed-forward network should be a sufficiently expressive architecture given that they are proven to be universal function approximators [340]. This first attempt was trained against a very small BVH training corpus. It was trivial to implement (since specific BVH motion fields can be parsed even with a Comma Seperated Values (CSV) parser) and was relatively successful.

Using it as a stepping stone we set out to define a different problem of regressing the X,Y and Z 3D coordinates of the hip of the BVH armature (BVH fields #1,#2 and #3 instead of #5), this time with the network

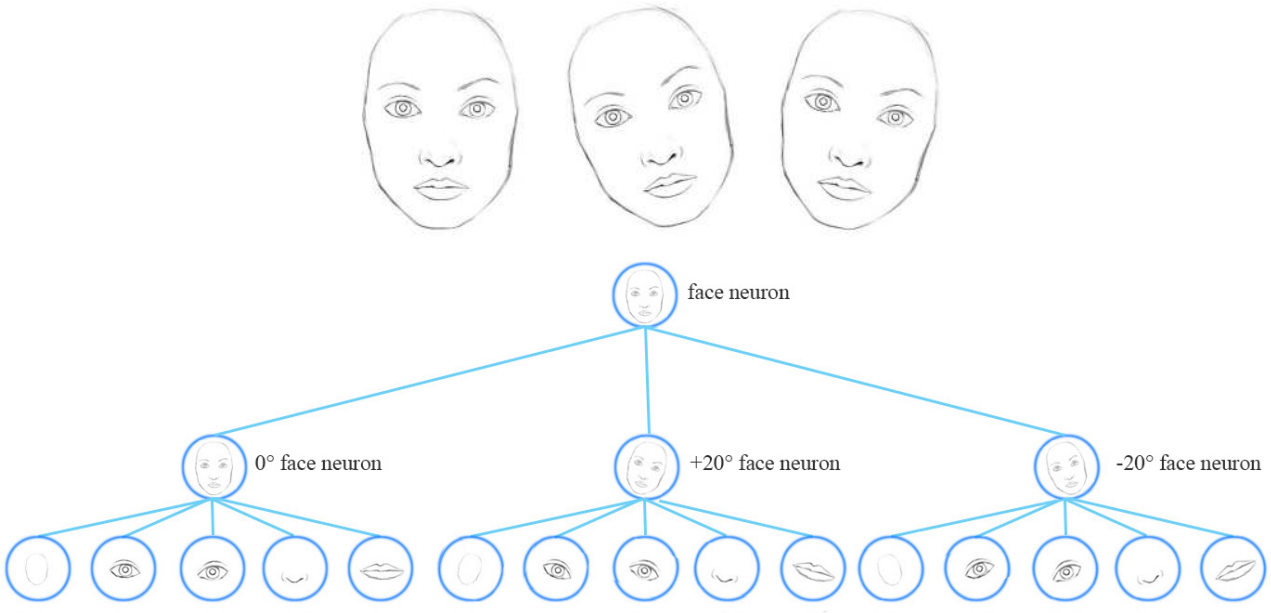


Figure 3.1: CNNs are very good at discerning patterns, however the convolutional paradigm by design tends to memorize training samples instead of smart generalization towards a global solution. Networks have to rely on a very large number of balanced training samples to avoid overfitting. This theoretical illustrated example from "Understanding Matrix capsules with EM Routing" [2] showcases the effect of trying to discern the same face under 3 different affine rotations (0°, −20°, +20°), and the substantial number of CNN "neurons" required to accomodate this task compared to Figure 3.2.

regressing a scalar value in millimeters instead of a soft-max classification score. Once again these experiments successfully regressed output just from 2D points giving very impressive results with respect to the simplicity of the approach. Surveying the literature we found a very similar method to our initial 3D regression attempts "A simple yet effective baseline for 3d human pose estimation" by Martinez et.al. [175]. This method converted a series of 2D joints to 3D joints with at 30% less error rates than the state of the art! Emboldened by these results and also aware of the research gap in direct Inverse Kinematics regression we extended the method to try and regress rotational IK initially focusing on the root DOFs. However, although the positional components could be successfully regressed, to our dismay the rotational degrees of freedom on these very naíve first attempts where highly unsuccessful, prompting the need to examine why.

Convolutional Neural Networks (CNNs) from RGB image sources (the first stage of our 2-stage formulation) use small convolution kernels that gradually convolve larger and larger parts of the input image until the output layer that is essentially "connected" and is thus calculated taking into account the whole input image. An important concept for CNNs is the **receptive field** of a neuron that denotes all of its ancestors that are somehow contributing to the calculation of its value. The success of Deep CNNs comes from their ability to gradually correlate all of the input pixels from all possible inputs from the training set and due to the robust weights derived by the back propagation algorithm [341]. In this way the very high dimensional input with incredible variances in appearance can be gradually handled and decomposed to simpler fea-

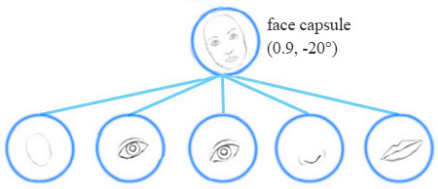


Figure 3.2: In contrast to the operation of a CNN pictured in Figure 3.1, capsule networks were proposed to detect equivariant transformations of features on a per capsule level being effective with a smaller network and generalizing without a massive number of samples.

tures that when combined lead to the correct output. An influencial work in 2017 by the team of Geoffrey Hinton that prompted the idea for the 2D descriptor we will elaborate on further was the "Dynamic Routing Between Capsules" paper [166] that introduced **Capsule Networks**. The idea behind capsules is to group areas of the neural network whose activity vectors represent a specific attribute or part of the observed item we wish to identify and make predictions on instantiation parameters of higher-level capsules. Figure 3.2 juxtaposes the concept of capsule networks compared to Figures 3.1 that shows the organization of a typical CNN.

For our 2D to 3D regression task, having readily available high-quality 2D joint estimations that free us from accounting for appearance variability on our input and allow us to work with abstract geometrical points. Keeping in mind the concepts of receptive fields, capsules, and the various 3D transformations that human bodies go under we set out to combine these to our advantage.

Perceptually, assuming a model like the skeleton based joint hierarchy of Figure 2.1 it is immediately apparent that in order to sufficiently define a 3D pose from its 2D projection, all of the the joints involved are needed. There is simply no way to regress the 3D pose of e.g. the right arm without having its 2D projections. Furthermore in the absence of other visual cues and frames of reference on the scene all of the joints take part and are defined in the reference frame of each other. It should be noted however that these simplified observations pale into comparison with state of the art research on the way our brain identifies and mentally rotates 3D objects which is an open research question [342–344] that has been long studied [345].

During 2017 (same year as Capsule networks [166]) a paper from Moreno-Noguer [172] utilized Euclidean Distance Matrices (EDMs) to encode input 2D images and formulated a 2D EDM to 3D EDM regression architecture with a post processing step that used the constraints rising from the 3D EDM and 2D input joints to resolve their 3D depths. This work showcased that EDMs could serve as a 2D descriptor for a Convolutional Neural Network (using convolutions with a local receptive field as mentioned before), regressing a similar dimensionality output with good accuracy. Although [172] successfully tackled depth regression for 2D points and the use of convolutions was a design choice due to the output having the same dimensionality as the input, the immediate correlation of all 2D input fields at the descriptor level was an idea that was not explored. Since matrices exhibited no locality on neighboring elements of the descriptor, this meant that no convolutions where not necessary and a different paradigm could be followed. Furthermore having features immediately available on the input layer meant that they could be possibly regressed with much a leaner and smaller NN. In the effort to provide ample input features that a NN could easily utilize a studied approach entailed using a Euclidean Distance Matrix (EDM). However EDMs exhibit symmetries, and for example could not accommodate the "easiest" task we attempted since a pose has the same

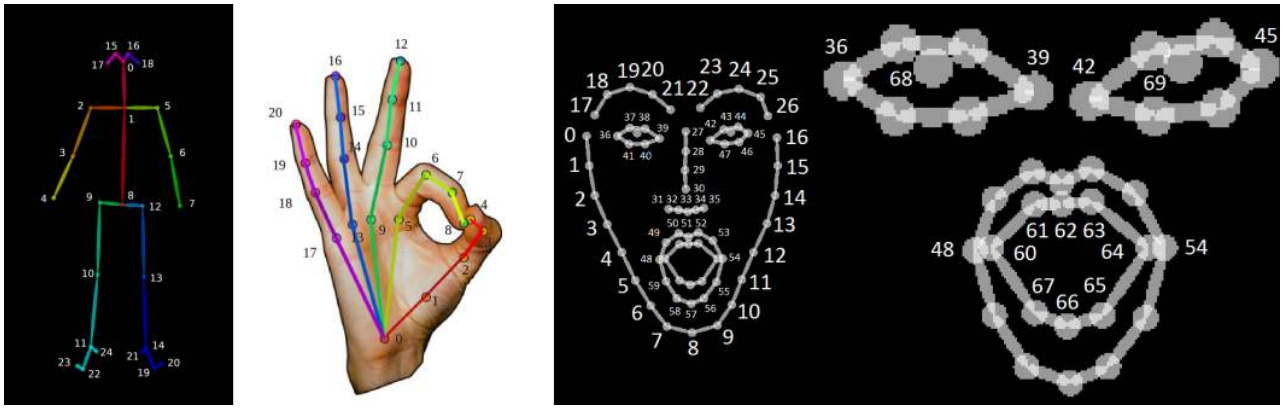


Figure 3.3: Joint orderings for different body hierarchies. From left to right, 1: The BODY25 [3] for upper and lower body. 2: The 21 2D points describing the right hand [4] with the left hand points having their mirrored locations. 3,4: Facial landmarks using the IBUG/Multi-Pie [5] standard.

Euclidean distances regardless of the 3D orientation of a subject, since euclidean distances just encode distance and not order. Looking at Figure 3.1 the 3 faces illustrated would have the exact same descriptors thus making it impossible for a NN to distinguish between the two. This limitation of EDMs was the Raison d'être behind the development of our first 2D descriptor we named Normalized Signed Distance Matrix (NSDM). Using it we successfully managed to regress BVH IK output and provide a first proof of concept of the method with our BMVC19 publication [6]. NSDMs had an important drawback, namely decoupling horizontal from vertical dimensions which led to a very big number of matrix elements and thus NN to handle them.

During 2020 and while trying to improve the method our research led to the creation of NSRM matrices [8]. It is worth noting that during the same year the very influencial to us work of Tancik et.al. [346] showed that **passing input points through a simple Fourier feature mapping** enables a multilayer perceptron (MLP) to learn high-frequency functions in low-dimensional problem domains. In a similar manner and with a geometric alignment (that however once again created sine and cosine features for the NN, through the use of the arctangent function) NSRM matrices proved very effective and more efficient for the problem. Once again however in the effort to further improve the method an enhanced version of the descriptor named eNSRM was provided providing 2D invariance while also encoding absolute rotation (much like the capsules in Figures 3.1 and 3.2) while at the same time also providing a secondary source of scaling data. The very recent work of Marchetti et.al. [347] formally proved that, under certain conditions, if a neural network is invariant to a finite group then its weights recover the Fourier transform on that group.

We will revisit the "information-theory" aspects [346–348] of the provided input descriptors on Section 3.8 that deals with NN/descriptor compression. However what we already managed to formulate using them is in essence the first stage of our inverse 3D renderer from 2D joints to the 3D graph of transformations. Having a closed form function to generate an arbitrary number of 3D articulated bodies and their 2D projections on any virtual camera, the pairs of 3D to 2D points we can begin to facilitate the inverse problem creating as many training samples as we want. Having a suitable 2D input descriptor meant that the NN has sufficient features to learn from how to regress we needed a way to represent our input.

Listing 1 Python implementation of our coordinate transformation function to normalize raw input points, while maintaining the same aspect ratio as our training set. Given $p_i = (x_i, y_i)$, normalizePoint(x=x_i,y=y_i,width,height) yields the normalized coordinates $p'_i = (x'_i, y'_i)$, where $0 \leq x'_i \leq 1$ and $0 \leq y'_i \leq 1$

```python
def normalizePoint(x,y,width,height,targetWidth=1920,targetHeight=1080):
    aspect_ratio = width / height

    if aspect_ratio > targetWidth / targetHeight:
        # The image is wider than the frame
        scaled_width = targetWidth
        scaled_height = int(targetWidth / aspect_ratio)
        offset_x = 0
        offset_y = (targetHeight - scaled_height) // 2
    else:
        # The image is taller than the frame
        scaled_width = int(targetHeight * aspect_ratio)
        scaled_height = targetHeight
        offset_x = (targetWidth - scaled_width) // 2
        offset_y = 0

    new_x = (x / width) * scaled_width + offset_x
    new_y = (y / height) * scaled_height + offset_y
    return new_x, new_y
```

The employed RGB image to 2D joints estimators typically use a serialization standard for their output, with COCO [110] and BODY25 [3] being examples of 2D joint input for the upper and lower body hierarchies. Input consists of a set of N points $J_{2D} = \{p_1, \ldots, p_N\}$. BODY25 joints have $N = 25$ and the respective 2D coordinates have the following order: Nose, Neck, RShoulder, RElbow, RWrist, LShoulder, LElbow, LWrist, MidHip, RHip, RKnee, RAnkle, LHip, LKnee, LAnkle, REye, LEye, REar, LEar, LBigToe, LSmallToe, LHeel, RBigToe, RSmallToe, RHeel. Out of these joints we typically select a subset of M points depending on their relevance to a regression task. Figure 3.3 illustrates the different hierarchies for the body, hands and facial areas along with their order. Coordinates $p_i = (x_i, y_i)$ typically correspond to individual input image pixels, for our method however and due to the multitude of available image sensors we use normalized coordinates with $0 \leq x'_i \leq 1$ and $0 \leq y'_i \leq 1$. An important detail, however, is that this normalization does not fully resolve sensor independence since another important issue is the image aspect ratio as we will see in later Sections. We handle normalization using the routine in Listing 1.

3.2.1 Euclidean Distance Matrix (EDM)

The Euclidean Distance Matrix (EDM) is an important concept used to describe structured points in various fields. It captures the geometric information of the points abstracting it from their actual locations thus providing an easier way to compare and deal with them. Furthermore, EDMs provide insights into the structure and arrangement of the data points in the Euclidean space. As already mentioned they had been used in conjunction with NNs prior to our work both in 2D and 3D [172] spaces.

EDMs encode pairwise distances between a given set of points in a Euclidean space. Their definition is

Notation	Description
$SO(2)$	Rotations in 2D Euclidean space (not involving reflections or inversions) that preserve distances (orthogonality) and angles between vectors. The set of 2×2 real orthogonal matrices with determinant 1. $SO(2) = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$
$SO(3)$	Rotations in 3D Euclidean space $\mathbb{R}^3$ under the operation of composition that preserve distances, angles, and the orientation of objects in 3D space without involving any reflections or inversions. $R_X(a) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos a & -\sin a \\ 0 & \sin a & \cos a \end{bmatrix} R_Y(\beta) = \begin{bmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{bmatrix}$ $R_Z(\gamma) = \begin{bmatrix} \cos\gamma & -\sin\gamma & 0 \\ \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix} Rotation_{XYZ} = R_X * R_Y * R_Z$ where a, β, and γ are the angles of rotation about the x, y, and z axes, respectively.
$SE(3)$	Rigid Body transformations in 3D Euclidean space $\mathbb{R}^3$ under the operation of composition. $\begin{bmatrix} R & \mathbf{t} \\ 0^T & 1 \end{bmatrix}$ Where: R is a 3x3 rotation matrix representing the rotation part of the transformation. $\mathbf{t}$ is a 3-dimensional translation vector representing the translation part of the transformation. $\mathbf{0}^T$ is a row vector of zeros. 1 is the scalar element.
$o(n)$	Distance-preserving transformations of an n-dimensional Euclidean space that preserve a fixed point.
$SU(2)$	Special unitary group, the group of 2×2 unitary matrices with determinant equal to 1. Related to $SO(3)$ with applications in quantum mechanics and angular momentum theory.
$\mathbb{H}$	Heisenberg group, the group of 3×3 upper triangular matrices of the following form: $\mathbb{H} = \begin{bmatrix} 1 & a & c \\ 0 & 1 & b \\ 0 & 0 & 1 \end{bmatrix}$
$SL(2,\mathbb{R})$	Special linear group $SL(2,\mathbb{R})$, often appearing in physics, especially in the study of symmetries and conformal transformations.
$Sp(2n,\mathbb{R})$	Symplectic group $Sp(2n,\mathbb{R})$, describing symplectic transformations in even-dimensional phase spaces.

Table 3.1: Lie Algebra Groups provide a mathematical model for the study of continuous symmetry.

the following: Given a set of M points, p_i in R^d, where $i = 1, 2, \ldots, M$ the *EDM* is a symmetric $M \times M$ matrix defined as:

$$EDM_{ij} = |p_i - p_j|_2, \tag{3.1}$$

where $|\cdot|_2$ denotes the Euclidean norm. The matrix *EDM* encodes the pairwise distances between all points in the set and serves as a fundamental representation of the spatial relationships within the data.

EDMs exhibit some interesting properties. First of all they are translation and scale-invariant, since multiplication of all distances by the same factor constant factor will not affect the intrinsic structure of EDMs if normalized against a known pair of points. A second characteristic is that the matrix is orientation invariant since pairwise distances do not change under affine transformations. A third attribute is that their diagonal is zero since the distance of each point from itself is always zero. Another characteristic is that they exhibit symmetry with respect to their diagonal. These characteristics have pros and cons for different applications. For example the diagonal and lower triangular EDM elements can be skipped to reduce NN

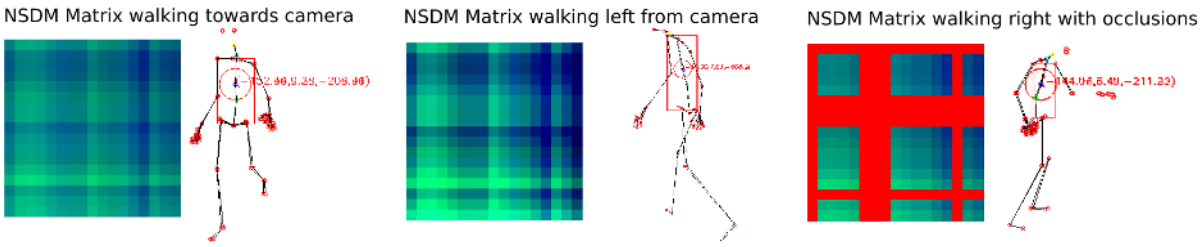


Figure 3.4: Visualization of the original Normalized Signed Distance Matrix (NSDM) encoding [6] using **RGB** images where **B** channel encodes pairwise 2D joint distance on the X axis (image width), **G** on Y axis (image height) and **R** occlusions (2D points not present). Although our encoding shares a lot of similarities with EDMs [7] our formulation maintains sign information, is more robust to scale changes and diagonal elements are 0.5 instead of 0.0, except when a joint is occluded.

complexity. The invariance characteristics can be very desirable traits such as in the case of [172], however for our work that required the regression of absolute orientation this descriptor was not ideal since e.g. all affine 2D rotations of the same 3D structure created the same EDM matrix thus not making it possible to recover the configuration of some degrees of freedom.

3.2.2 Normalized Signed Distance Matrix (NSDM)

Normalized Signed Distance Matrices (NSDMs) were proposed as part of our original 2019 MocapNET publication [6]. This early crude version of the approach handled all of the body at once, requiring a lot of 2D points, however even this approach did not utilize all N input 2D points. The selection of M to create the NSDM matrices was: hip, neck, head, l/r collar, l/r shoulder, l/r elbow, l/r hand, l/r hip, l/r knee and l/r foot. Due to the human body having most of its points along the vertical axis 2 artificial points where added to contribute with more features on the horizontal axis that better encode small variations in pose that could otherwise be lost. The generated points where positioned left and right of the hip displaced on the x axis by -0.3 and +0.3 units relative to normalized image width.

Each 2D joint p_i where $1 \leq i \leq M$ was associated with its coordinates $p_i = (x_i, y_i)$ which where normalized using Listing 1 to the input image frame dimensions and therefore bounded in the range $[0, 1]$. We also associated each such point with a visibility status parameter v_i provided by thresholding joint confidence values from our source (e.g OpenPose, with 1 for visible joints and 0 for occluded joints). An important property of NSDMs is that each dimension is encoded in its own individual matrix. For our 2D points we calculated 2 matrices $NSDM^x$ and $NSDM^y$. The entries of the NSDMs where calculated as

$$NSDM^c(i,j) = \begin{cases} 0.5 + c_i - c_j & v_i \neq 0, v_j \neq 0 \\ 0 & otherwise, \end{cases}$$

for $c \in \{x, y\}$. Using this formulation diagonal elements have a value of 0.5 except when occluded. The NSDM deviations from the original EDM formulation offer multiple advantages. They allow negative values (< 0.5 in our case) which are useful since they are a source of information differentiating symmetric changes in the configuration of limbs. Our data representation also anticipates the behavior of the SeLU [28] activation

function since occluded values fall on its non-linear part. After calculating $NSDM^x$ and $NSDM^y$ we calculate the length of the torso and use its value to normalize all matrix elements. This final normalization increases the robustness to scale changes since these affect all limbs. Sample 17x17 NSDMs are visualized in Figure 3.4 using different channels for the different coordinate dimensions in order to provide pictorial intuition about their properties.

It is also very important to once again stress that NSDMs are by their definition translation invariant and resistant to scale changes. The 2D rotation invariance of the points of EDMs is lost meaning that the matrices could be used to regress the 3D absolute orientation of the observed armature. The number of created features is much higher than EDMs with the lower diagonal featuring different signs (because of the opposite order of joints). Instead of convolving kernels in a CNN, by creating input with pairwise association of all inputs we achieve a similar effect in a shallower network with less operations. Having alternating parts of the Matrix with higher/lower values for different joint orderings, combined with the separation of the human orientation classes we will see in section 3.5 greatly simplified the task of learning-based 3D pose estimation allowing the first network to operate despite its simplistic formulation.

3.2.3 Normalized Singed Rotation Matrix (NSRM)

Building on the NSDM experience, we developed an improved descriptor we named Normalized Singed Rotation Matrix (NSRM). NSRMs were proposed as part of our 2020 ICPR publication [8] that internally marked the transition to MocapNET version 2.0.

Along with other methodological improvements, the upper body was disconnected from the lowerbody to be able to successfully tackle very commonly found video input instances where the lowerbody was occluded, since humans tend to focus on the upperbody of persons when aiming the cameras. The estimated 2D joints hierarchy was thus encoded into two Normalized Signed Rotation Matrices (NSRMs), one for the upper body and one for the lower body. NSRMs encoded 2D poses being translation and scale invariant and efficient in terms of the total element count required to encode a 2D pose. It departed from EDMs and NSDMs while still maintaining some conceptually similarities. NSDMs where normalized to be invariant also to scale changes, however, one of their disadvantage is that they required two Matrices, one for X and one for Y joint coordinates making them computationally more demanding when used with a fully connected feed-forward NN in terms of total number of elements. NSRMs on the other hand offered half of the total parameter count of NSDMs which combined with the geometric increase of connections from dense connections made them a much more computationally economical option.

Encoding a BODY25 [3] encoded standard 2D joint input that consists of $N = 25$ 2D points $J_{2D} = \{p_1, ..., p_{25}\}$ [4]. Out of those, we select two subsets of 2D joints to create an NSRM matrix. In particular, the body hierarchy is split in upper and lower body to combat occlusions, so we define one NSRM for each part. To derive the upper body $NSRM^u$ we use joints hip, l/r eye, neck, l/r shoulder, l/r elbow and l/r hand. Using just these 10 joints would create a 10x10 NSRM however in order to be fairly compared to the baseline method [8] and remove the input size as one of the comparison variables we also used a 17x17 matrix size by adding another 7 virtual joints. 5 of them are generated by creating new virtual points between two real points. Namely we create four new artificial points in the middle of Shoulders and Elbows, Elbows and Hands as well as one point in the middle of the distance between hip and neck. The rest 2 points are created by displacing the hip by (-0.15,-0.15) and (0.15,-0.15) units relative to the normalized input image dimensions. Having more points encoding the arm helps with better encoding arm poses which are our target. The last 2 artificial points contribute with more features on the horizontal axis to better encode poses with

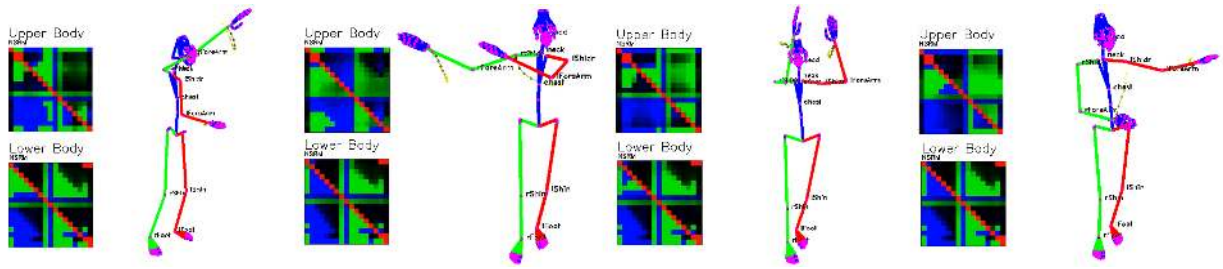


Figure 3.5: Visualization of the original Normalized Signed Rotation Matrix (NSRM) encoding [8] for upper and lower body. Notice that diagonal elements are 0.0, which is one of the main differentiatons with eNSRM descriptors.

small variations. For the lower body $NSRM^l$ joints used are hip, r/l thigh, r/l knee, r/l heel and r/l big toe. Once again we pad the joints by creating new artificial points between thighs and knees, knees and heels as well as adding virtual points that are displaced by (0,0.15), (0,-0.15), (0.3,0) and (-0.3,0) units in relation to the hip bringing the total number of points to 17 once again.

An NSRM associated with M joints is constructed as follows. The coordinates (a_x, a_y) of a participating 2D joint a are normalized to the input image frame dimensions and are thus bounded in the range $[0, 1]$. We also associate each such joint with a visibility parameter a_v provided by thresholding the OpenPose joint confidence values (1 if joint is visible, 0 if joint is occluded).

For each pair of 2D joints $a = (a_x, a_y)$, $b = (b_x, b_y)$ we define a new point $c = (b_x, b_y - | b - a |)$ that translates point b vertically by the length of vector ab. Using the three points a, b, c we can encode the relation between points a and b as well as their relative rotation towards a fixed vertical axis as follows:

$$NSRM(a,b) = \begin{cases} tan^{-1}((|\vec{ab} \times \vec{cb}|)/(\vec{ab} \cdot \vec{cd}) & a \neq b, \\ 0 & otherwise \end{cases} \tag{3.2}$$

where $\cdot$ and $\times$ denoting inner and cross products, respectively.

For each pair of 2D points a, b we can declare a new point $c = (b_x, b_y - | b - a |)$ that is the point b translated vertically by the length of vector ab. Using these three points and the atan2 function, we can encode (Equation 3.2) the relation between points a and b as well as their relative rotation towards a fixed vertical axis, that is:

$NSRM^h(a, b, c)$ is invariant to skeleton translation and scale. The representation encodes the relative position of joints (albeit using the rotation formed from triangle $a\hat{b}c$), as well as orientation (since bc is parallel to the y axis of the world). Finally, joint order is preserved through the sign of the *atan2* function.

An advantage of this encoding compared to NSDMs [6] is that we can easily force alignment of all retrieved angles using a pivot point and rotation. In the body pose estimation scenario [8] where humans typically stand upright this was not a very important characteristic but the development of the descriptor had hand pose estimation (the next milestone towards the Total Capture method presented) in mind while being developed. Using NSRMs did in fact provide benefits with its possibility of controlling the encoded rotation of the 2D joints by always aligning input matrices to a pivot point (e.g. hip to neck) in a way that makes the descriptor rotation invariant, however this design characteristic was explored in its final formulation eNSRM we will describe in the next section.

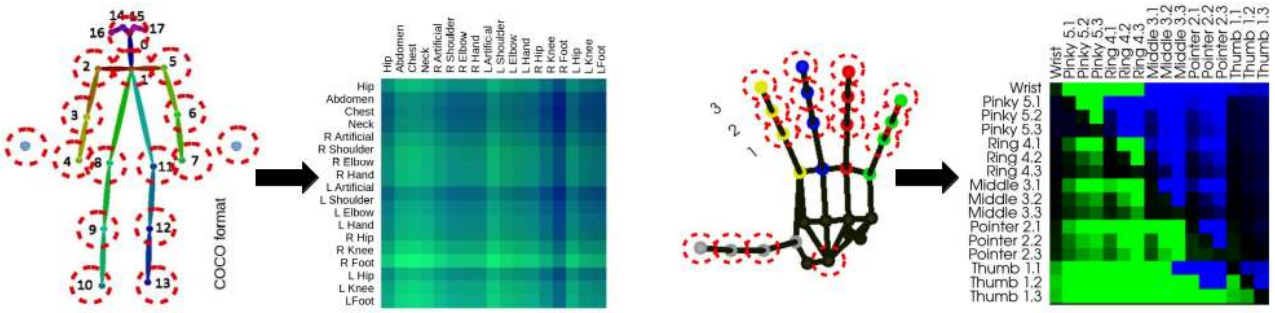


Figure 3.6: Pictorial comparison of the NSDM matrix from [6] encoding both upper and lower body, to the NSRM matrix from [9] encoding a left hand.

$NSRM^u$ for upper and $NSRM^l$ for lower body are formed by computing the respective NSRM for all joint pairs of the corresponding body part. Figure 3.5 illustrates sample 17×17 NSRMs for different human poses. Having all pairwise joint relations encoded in a matrix means that available features for the neural network can be readily leveraged in a relatively shallow and thin network without requiring too many operations. A great improvement compared to NSDMs [6] is that an NSRM is encoded in a single channel, as opposed to NSDMs that require two, one for the x and one for y joint coordinates. Thus, NSRMs use half the element count compared to NSDMs. For dense neural networks this amounts to a drastic parameter decrease. For every two consecutive layers of dense networks there is a connection between all possible combinations of elements making a massive difference in the size of the neural network since connectivity scales exponentially. The dense networks will be described later and are illustrated in Figure 3.38. Thus, NSRMs constitute a resource-efficient 2D pose encoder and one of the novelties of our method that might prove useful for similar problems.

3.2.4 enhanced Normalized Signed Rotation Matrix (eNSRM)

The presented Total Capture method follows the same 2-stage approach that generalizes across all of the observed hierarchies. The RGB image is first processed through a convolutional network yielding discrete 2D points with a detection confidence for each body region as seen in Figure 3.3.

We receive 25 body, 42 hand, 68 facial and 2 iris landmarks with normalized coordinates (a_x, a_y) in the range $[0, 1]$ for each 2D joint a. Each joint also carries a visibility parameter a_v which we threshold against a configurable lower limit marking and thus eliminating low confidence or occluded joints with $0, 0$. For each pair of visible 2D joints $a = (a_x, a_y)$, $b = (b_x, b_y)$ we can define a new point $c = (b_x, b_y - \mid b - a \mid)$ that translated the point b vertically by the length of vector ab. These three points a, b, c are used to encode the relation between points a and b as well as their relative rotation towards a fixed vertical axis as follows [9]:

$$eNSRM(a,b) = \begin{cases} \frac{2}{\pi} tan^{-1}\left(\frac{|\vec{ab} \times \vec{cb}|}{\vec{ab} \cdot \vec{cb}}\right) & a \neq b, \\ |\vec{aR}| & otherwise, \end{cases} \tag{3.3}$$

with $\cdot$ and $\times$ denoting inner and cross products.

Each resulting $eNSRM(a,b)$ angle is invariant to 2D point cloud translation and scale. Scale is encoded in the diagonal with Euclidean distances from the R root joint, the relative position of joints using the rotation formed from triangle $a\hat{b}c$, while preserving their relative orientation to the world coordinate system (with bc

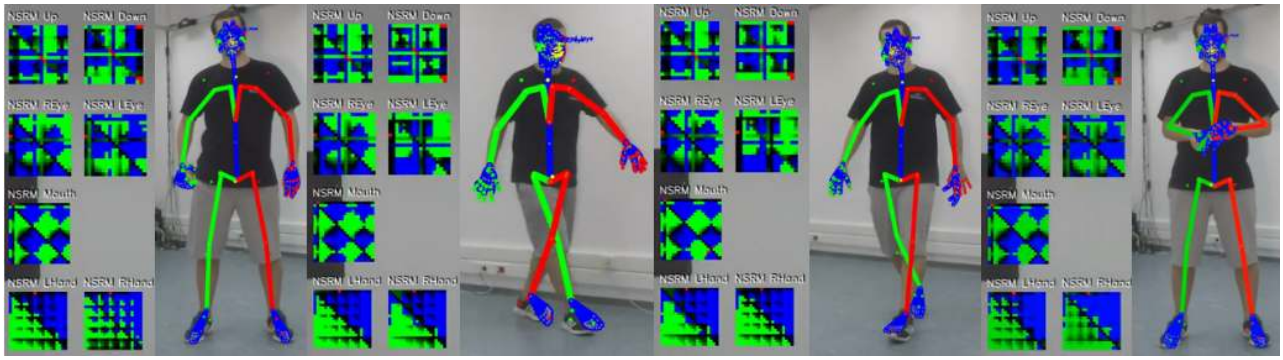


Figure 3.7: Visualization of enhanced Normalized Signed Rotation Matrix (eNSRM) encodings [9] of the complete proposed total capture method. Matrices visualized using Blue color for negative values and Green color for positive values, scaling color intensities linearly according to each matrix element magnitude. Each region of the body (upperbody,lowerbody,leye,reye,mouth,lhand,rhand) is described by an independent matrix. Encoding the points in Figure 3.3

being parallel to the Y axis). In contrast to the baseline eNSRM [9] formulation the final formulation [31] we take into account that $tan^{-1}(\mathbb{R}) \longrightarrow (-\frac{\pi}{2}, \frac{\pi}{2})$ and multiply values with $\frac{2}{\pi}$ to normalize them in the range [-1,1]. A pictorial visualization of the matrix can be seen in Figure 3.7.

Enhanced Normalized Signed Rotation Matrices (eNSRMs) for hands: Our hand model follows the MakeHuman [23] skeleton definitions and each of its five fingers consists of four joints as seen in Figure 2.3. We use the final three joints of each finger plus the wrist location to build our eNSRM. This joint selection yields a total of $M = 16$ 2D input points $J_{2D} = \{p_1, ..., p_{16}\}$. Out of those, we treat the wrist p_1 as our special root point R and the rotation angle of the wrist to middle finger proximal phalanx compared to the world Y axis as our R_r root rotation.

The *eNSRM* matrix we described above was capable to be used as a body pose descriptor in constrained orientation scenarios like 2 way [6] or 4 way [8] orientation grouping. However, hands exhibit a much larger variety of 3D rotations in space compared to human torsos that are typically upright. In our effort to disencumber the neural networks from the harder task, we perform an affine 2D rotation of all 2D joints using the inverse root point orientation $-R_r$. This brings the wrist to middle finger vector to an upright orientation parallel to the Y world axis. We record R_r and store it to the first (previously empty) diagonal *eNSRM* element. Using this transformation hand poses become 2D rotation invariant except for their first element. The problem resolved with *eNSRM*s can be better understood with the following example. Assuming a hand pose, if we perform affine rotation of the observed 2D joints, we get *NSRM* matrices with large changes in all elements even if no articulation change occurred. All this variability has to be correctly handled by the NN despite essentially encoding the same pose. *eNSRM*s are decorrelated from 2D rotation while allowing the absolute 3D rotation of the hand to remain retrievable by the NN via the first element. Figure 3.8 illustrates the orientation correction performed to reduce the complexity of the hand pose estimation task.

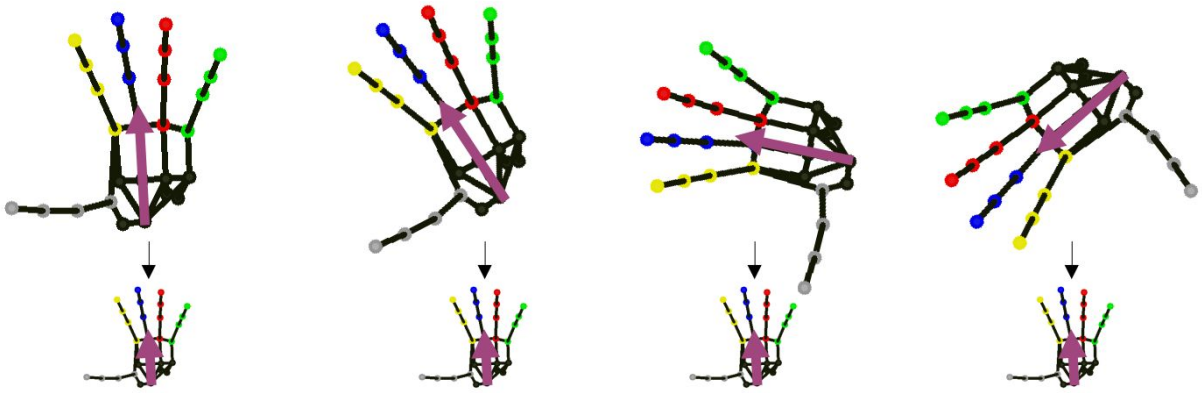


Figure 3.8: A hand configured with the same pose has different eNSRM encodings if it is rotated. In contrast to the body that is typically upright hand orientation is much different. We can align the Wrist to Middle Metacarpal vector to our camera Y axis, record the orientation correction in the first (previously empty) eNSRM element and then re-orient the 2D points to align the hand making the descriptor 2D rotation invariant.

3.2.5 Recurrence Plots (RPs)

Recurrence Plots (RPs) are a visualization and analysis technique used in the field of time series analysis. Introduced in 1995 [349], RPs provide a powerful way to explore and understand the dynamic behavior of complex systems, particularly those exhibiting recurrent patterns or periodicities. This technique has found applications in various domains [350], including physics, biology, engineering, and finance.

At its core, a Recurrence Plot is a graphical representation of the relationships between different instances or points within a time series. It visualizes the times when similar or identical states reoccur in the data. By doing so, RPs help researchers identify hidden patterns, periodicities, and correlations in time series data that might be difficult to discern using other methods.

The construction of a Recurrence Plot involves comparing each point or data instance in the time series with all other points to determine their similarity. A common approach is to measure the distance between pairs of points and set a threshold value. If the distance between two points is below the threshold, a "recurrence" is marked at the corresponding position in the Recurrence Plot. This process results in a binary matrix where the presence of "recurrences" is indicated by black or colored pixels, while the absence of recurrences is represented by white space.

Recurrence Plots provide valuable insights into the temporal evolution and behavior of dynamic systems. Researchers can visually identify diagonal lines, clusters, and other patterns within the plot, each of which corresponds to specific recurring behaviors or events in the time series. Additionally, various quantitative measures can be derived from Recurrence Plots, such as the percentage of recurrence, the length of diagonal structures, and more, enabling a deeper analysis of the data.

Recurrence Plots are created from time series data by constructing a matrix where each entry R_{ij} indicates whether the state of the system at time ii is close to the state at time jj. The closeness or similarity between two time points is typically determined using a predefined distance metric or similarity measure.

For example, if you have a time series $x(t) = [x_1, x_2, \ldots, x_n]$, you can compute the pairwise distances

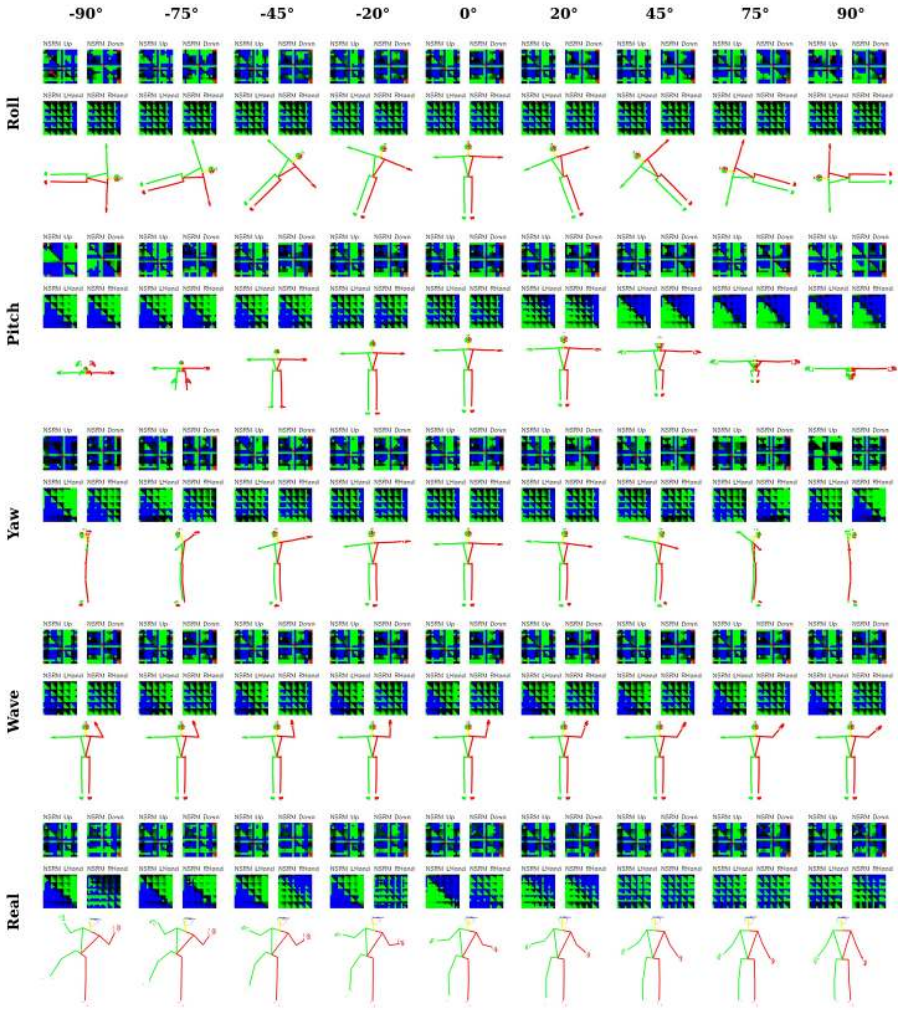


Figure 3.9: Illustration of different 3D configurations and the resulting body and hand eNSRM matrices. Rows Roll,Pitch,Yaw and Wave depict a T-Pose with only one degree of freedom varying w.r.t. the setting shown on column label. The final row depicts a real skeleton making a complex motion while also exhibiting noise .

between all time points using a suitable metric (e.g., Euclidean distance, Manhattan distance, etc.). The Recurrence Plot matrix R is then constructed as follows:

$$R\begin{cases}1 & \text{if } \|x_i - x_j\| \leq \varepsilon \text{ (a predefined threshold)} \\ 0 & \text{otherwise}\end{cases} \tag{3.4}$$

This matrix is then visualized as a binary image, where black pixels represent recurrences (similar states), and white pixels represent non-recurrences (dissimilar states). The resulting image can provide insights into the structure of the time series, revealing patterns, symmetries, and repetitions within the data.

Recurrence Plots are widely used in fields like chaos theory, nonlinear dynamics, and analysis of complex

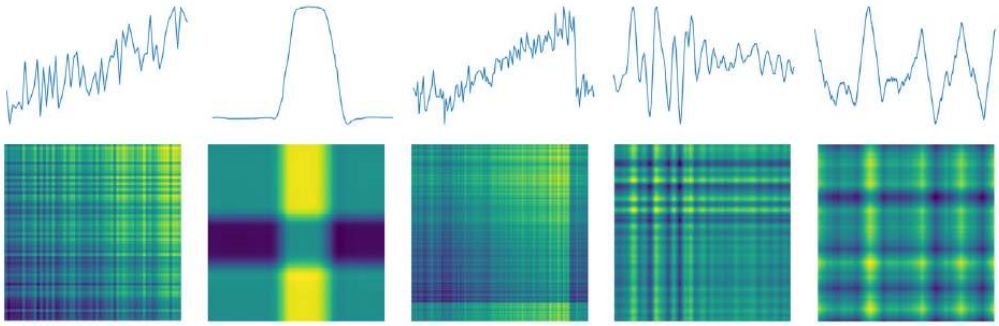


Figure 3.10: 2D Descriptor encodings of time series 1D signals using Relative Position Matrices. Illustration from the 2019 publication of Chen et.al. [10].

systems. They can help identify periodic behaviors, detect phase transitions, and uncover hidden structures in time series data. Additionally, various quantitative measures can be derived from Recurrence Plots, such as the percentage of recurrence, determinism, and entropy, which provide further insights into a system's dynamics.

In summary, Recurrence Plots are a powerful tool for visualizing and analyzing the structure of time series data, making them valuable in understanding complex systems and dynamical behavior.

3.2.6 Relative Position Matrix (RPM)

The Relative Position Matrix (RPM) is a formula used to transform 1D time series data into 2D image representations, which are then suitable for processing by convolutional neural networks (CNNs). RPM captures the relationships and differences among various data points within a single time series. It constructs a matrix that indicates how each data point relates to others in terms of their positions. This matrix essentially encodes the temporal sequence information of the time series data.

By converting time series data into RPM-based images, this approach aims to preserve the temporal characteristics of the data while adapting it to the strengths of CNNs in image recognition tasks. This transformation can help enhance the performance of CNN-based classification algorithms for time series data. The RPM method allows for more efficient processing of time series data by utilizing the convolutional operations inherent in CNN architectures.

$$M = \begin{bmatrix} x_1 - x_1 & x_2 - x_1 & \dots & x_m - x_1 \\ x_1 - x_2 & x_2 - x_2 & \dots & x_m - x_2 \\ \vdots & \vdots & \ddots & \vdots \\ x_1 - x_m & x_2 - x_m & \dots & x_m - x_m \end{bmatrix}, RPM = \frac{M - min(M)}{max(M) - min(M)}$$

3.2.7 Gramian Angular Summation / Difference Field (GADF/GASF)

The Gramian Angular Summation/Difference Field (GADF/GASF) method was developed in 2015 by Wang et.al. [351], This work led to multiple others some of which coupled with Histogram of Oriented Gradients (HOG) citexiao2020gadf or Principle Component analysis (PCA) [352] feature extraction. It is a technique used for extracting informative features from 1D time series data, particularly for tasks like electromyography (sEMG) signal classification, but also used for Speech Recognition [353], hand movement classification

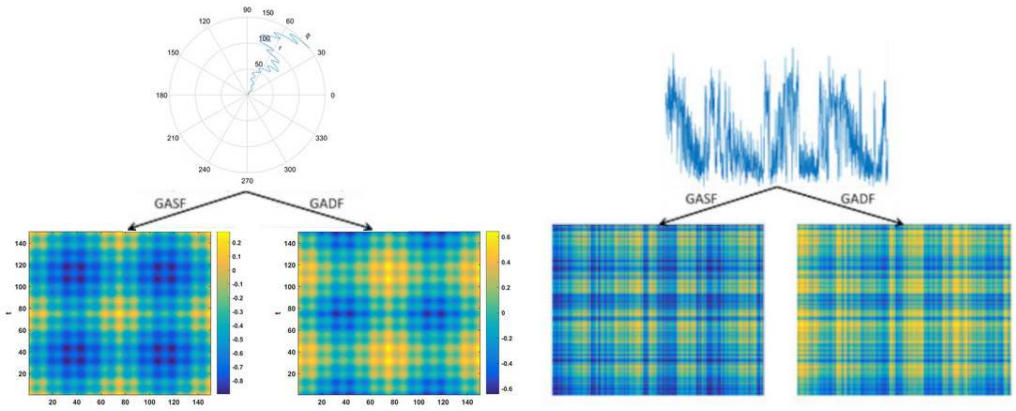


Figure 3.11: Two GASF/GADF 2D Descriptor encodings using a signal with polar coordinates and regular 1D visualizations. Illustration taken from [11] [12].

based IMUs [354] and more.

The creation of the descriptor involves a series of steps:

1. Data Normalization, with the raw time series data first being normalized using the following equation:

$$\widetilde{x} = \frac{(x - max(X)) + (x - min(X))}{max(X) - min(X)} \tag{3.5}$$

2. Angular Transformation, that converts the normalized data $\widetilde{x}$ into angular values using the arccosine function:

$$\begin{cases} \varphi = arccos(\widetilde{x_i}) \\ r = \frac{t}{n} \end{cases} \tag{3.6}$$

3. Gramian Angular Summation Field (GASF) and Gramian Angular Difference Field (GADF) Calculation using the following equations:

$$GASF = cos(\varphi_i + \varphi_j) = \widetilde{x_i} \cdot \widetilde{x_j} - \sqrt{1 - \widetilde{x_i}^2} \cdot \sqrt{1 - \widetilde{x_j}^2} \tag{3.7}$$

$$GADF = sin(\varphi_i - \varphi_j) = \widetilde{x_j} \sqrt{1 - \widetilde{x_i}^2} - \widetilde{x_i} \sqrt{1 - \widetilde{x_j}^2} \tag{3.8}$$

4. Depending on the methodology, Histogram of Oriented Gradients (HoG) (not to be confused with Hessian of Gaussians that is used to analyze the curvature of functions) calculating the distribution of gradient orientations within the matrices, capturing local data patterns when values fluctuate.

In summary, the GADF/GASF method transforms 1D time series data into informative image-like representations using GADF and GASF transformations, followed by HOG feature extraction citexiao2020gadf or PCA [352]. These features are used for accurate classification in applications such as hand prosthetics control, rehabilitation and more.

Figure 3.12: Visualization of locations, orientations and magnitudes of SIFT [13] features for an RGB image. Our method favors NNs due to their better accuracy. However SIFT features continue to offer greater computational performance compared to NNs [14] and an interesting research direction [15].

$$M_{GASF} = \begin{bmatrix} cos(\varphi_1 + \varphi_1) & \dots & cos(\varphi_1 + \varphi_n) \\ \vdots & \ddots & \vdots \\ cos(\varphi_n + \varphi_1) & \dots & cos(\varphi_n + \varphi_n) \end{bmatrix} \tag{3.9}$$

$$M_{GADF} = \begin{bmatrix} sin(\varphi_1 + \varphi_1) & \dots & sin(\varphi_1 + \varphi_n) \\ \vdots & \ddots & \vdots \\ sin(\varphi_n + \varphi_1) & \dots & sin(\varphi_n + \varphi_n) \end{bmatrix} \tag{3.10}$$

3.2.8 Image Descriptors vs. Geodesic Descriptors

Concluding our descriptor Section, after providing information about Geodesic Descriptors and Time-Series Descriptors a final category of descriptors we will briefly go through are image descriptors. Two prominent families of descriptors are the SIFT [13] and SuRF [355] descriptors, which differ from geodesic descriptors like EDMs, NSDMs, NRSMs, eNSRMs, RPs, RPMs and GASFs/GADFs in terms of their characteristics and applications. Our work capitalizes on neural networks for the task of image analysis and 2D/3D pattern recognition, and thus we do not utilize them. However they are worth mentioning in the general context of descriptors since they could be used in multiple ways (e.g. at the RGB level, or after regressing an NSDM, NSRM, eNSRM).

Scale-Invariant Feature Transform (SIFT) and Speeded up robust features (SURF) descriptors are widely used for detecting and describing local features in images. They are robust to variations in scale, rotation, and illumination, making them valuable for object recognition, image stitching, and image-based localization. SIFT descriptors extract distinctive keypoints and their associated orientations, allowing for reliable feature matching even under significant image transformations. SuRF and SIFT descriptors are designed for image data, whereas geodesic descriptors like EDMs are used for point cloud data and GASF/GADF descriptors are used for time series data.

In summary, SuRF and SIFT descriptors are tailored for image analysis, focusing on local image structures, while geodesic descriptors like EDMs and GASF/GADF are versatile tools for capturing geometric and temporal relationships in point cloud and time series data, respectively. The choice of descriptor depends on the data type, application, and the specific information one seeks to extract. Although SIFT based methods have been superseeded by NNs [356], they offer an interesting research direction that is actively being explored [15] and it would be interesting to be used in conjunction with NNs or with other descriptors like the ones we developed and presented.

3.2.9 Leveraging Symmetries and Tackling Occlusions

Regressing the lost third dimension is mathematically ill-posed due to the inherent ambiguity introduced by the projection process. When a 3D configuration is projected onto a 2D plane, multiple different 3D configurations can lead to the same set of 2D projections as seen in Figure 3.13. This results in a one-to-many mapping between the 3D space and the 2D space, making it extremely challenging to accurately recover the original 3D skeleton solely from its 2D projections. As a consequence of this ill-posed nature, traditional regression methods may struggle to produce accurate and reliable 3D reconstructions, especially in cases where occlusions, foreshortening, and complex pose variations are involved. Addressing this ambiguity requires incorporating additional constraints or information, such as prior knowledge about human anatomy, motion dynamics, or using multiple views, to improve the accuracy and robustness of 3D skeleton regression from 2D point projections.

While symmetric solutions in the axis of Z (depth) raise obstacles towards correct 2D to 3D regression, the same is not true in the X and Y axis where symmetries can be leveraged to our advantage. Figure 3.16 showcases a horizontally flipped version of Figure 3.9. Observing the eNSRM patterns recorded we can see that symmetries in conjunction with the chirality of hierarchies like the human hands allow us to perform horizontal mirroring and get ”chirality-invariant” eNSRM matrices. As we will see in the next Sections our networks capitalize on this allowing regression of mirrored regions using the same NN ensemble.

Symmetries can even be used to infer and thus fill-in missing data. As seen in Figure 3.14, our formulation splits the human head in three regions (Left/Right Eye and Mouth) encoding it using three eNSRM matrices. We can handle both eyes using the same NN by performing a horizontal flip, while in cases of one of the output matrices missing, and assuming that eyes exhibit symmetry in their motions recover the occluded eye.

Very commonly we observe input instances where no symmetry cues can help us deal with occlusions. A very common example illustrated in Figure 3.15 are videos where the lower body is completely occluded. This is very often the case since cameras tend to focus on the upperbody. All geodesic descriptors suffer in similar situations due to substantial parts of the matrix missing. The impact on the Neural Networks that handle the regression task however is great leading to degradation even in fully visible parts of the input, since large segments of the neural network cease to activate thus disrupting the operation of the whole NN. The solution to this problem is subdivision and partitioning into separated matrices that are split the effect and impact of occlusions across different matrices.

3.3 Data Representations, Datasets and Motion Capture

Motion capture, often referred to as MOCAP, is a term referring to the technology that deals with capturing and recording of human or object movements for various applications such as animation, gaming, movies,

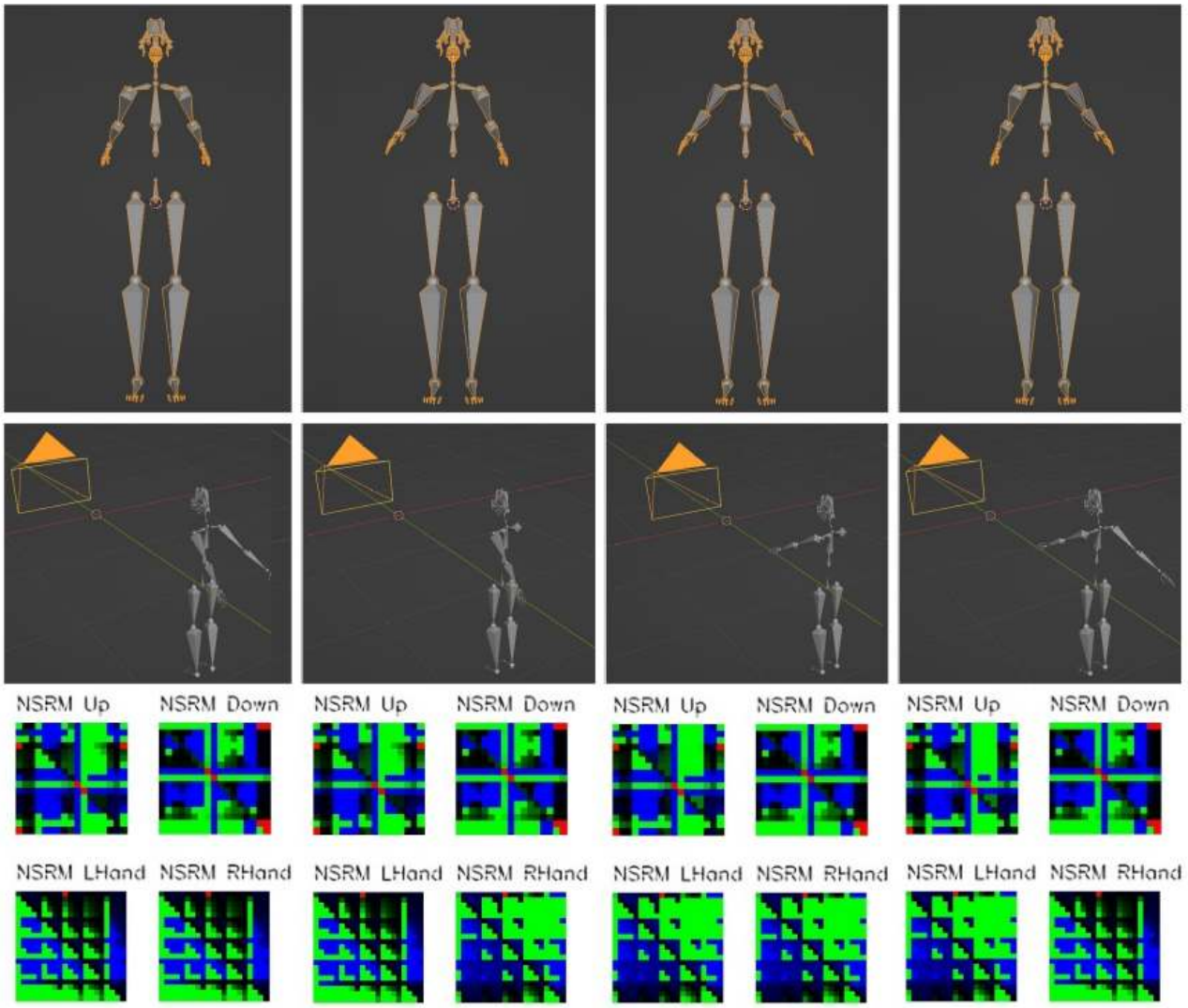


Figure 3.13: 3D human armatures exhibit symmetric configurations with respect to the Z-axis (depth) that provide identical 2D joint projections. This makes it difficult to recover them using purely 2D input since multiple 3D poses correspond to one 2D input. The illustration shows 4 columns with different skeletons with l/r shoulder rotations ±60° that resolve to the same 2D projections (and thus NSDM, NSRM and eNSRM encodings).

virtual reality, and more. Motion capture formats play a crucial role in storing and representing the captured movement data, ensuring compatibility between different software and systems. These formats facilitate the seamless transfer of motion data across various platforms and applications, allowing for effective utilization in a wide range of industries. Many computer visions define their own motion capture containers that suit specific applications and needs. Our work however was built from the ground up having the MOCAP container in mind. Using a well defined open standard with already existing bindings in popular 3D

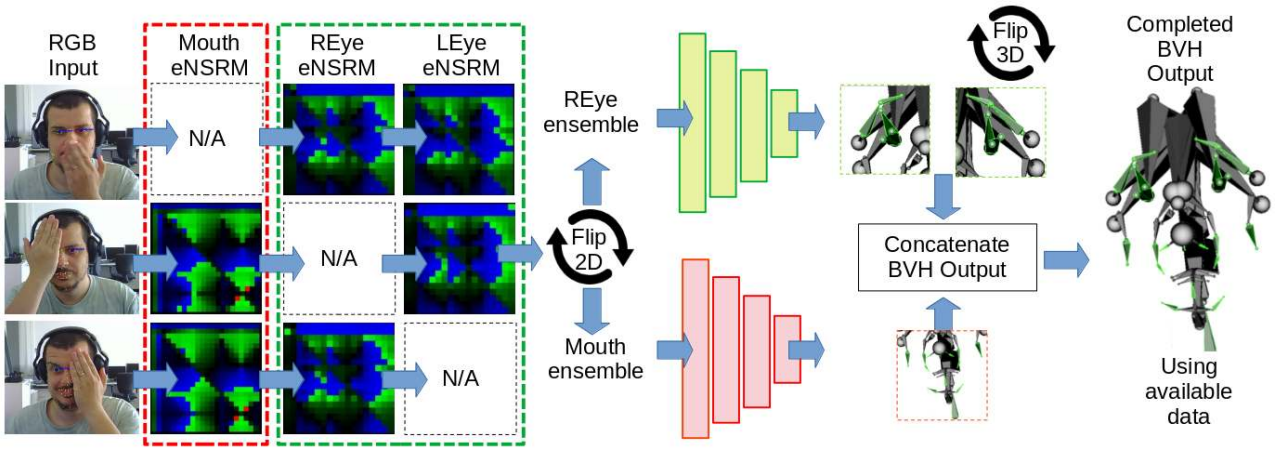


Figure 3.14: The illustration showcases the constructed eNSRM matrices for three RGB inputs on the left showcasing substantial occlusions. Our divide and conquer approach gracefully handles such cases without affecting visible portions of the image. We use the same ensemble for L/R Eye by leveraging 2D input / 3D output symmetries. We can also populate the invisible eye by mirroring the visible one. We visualize eNSRM [9] matrices using Blue color for negative values and Green color for positive values, scaling colors linearly according to each matrix element magnitude.

editing software is not only important for the dissemination of the method but also to render it useful to the broader Computer Vision and Graphics community. Maddock et.al. offers a comprehensive survey [77] explaining in detail many of the popular Motion Capture File Formats.

There is a multitude of motion capture formats available, each with its own specifications, strengths, and purposes. A brief list of the main motion capture formats include:

- **BVA (Biovision Hierarchy Animation)**: BVA is a format that stores motion capture data as a hierarchical structure, capturing bone movements and joint rotations. It is widely used for biomechanics research and medical applications due to its ability to represent complex skeletal structures.
- **BVH (Biovision Hierarchy)**: Similar to BVA, BVH is a popular format for representing skeletal animation data. It stores hierarchical information about bones and joints, making it suitable for applications like character animation and game development.
- **ASC/ASF (Acclaim Skeleton File)**: ASC is a file format developed by the Acclaim motion capture system. It contains information about joint angles, positions, and segment lengths, bone hierarchy, limits, and other properties, making it compatible with various animation and simulation software.
- **AMC (Acclaim Motion Capture)**: AMC is another format from the Acclaim system, designed to store motion capture data, including joint angles and positions. It works in conjunction with the ASF format to represent both skeletal structure and movement.
- **ASK (Acclaim Skeleton Motion Data)**: ASK is a combination of ASF and AMC formats, encapsulating both skeleton structure and motion data in a single file.

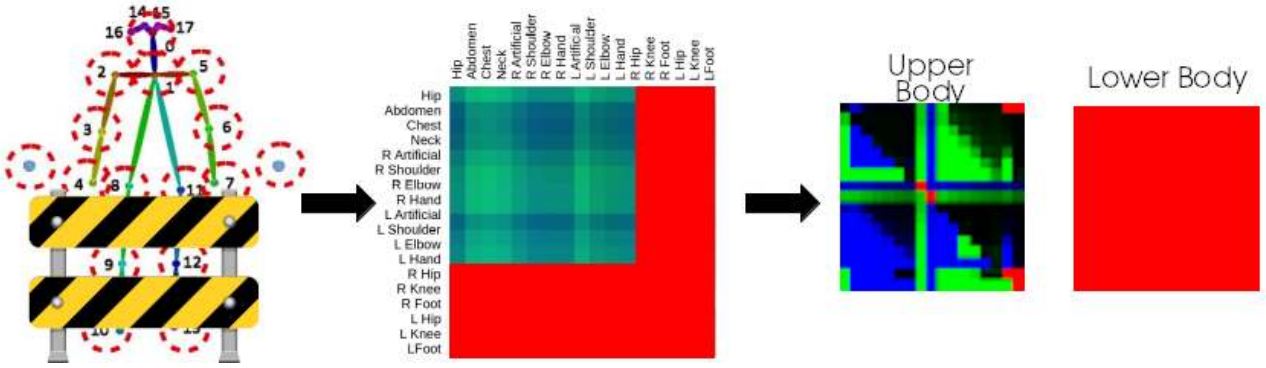


Figure 3.15: Occluded observations are very frequent in in-the-wild videos. The NSDM, NSRM, eNSRM descriptors utilized lose one row and one column per occluded joint. This makes large parts of the descriptor missing leading to operation difficulties when 2D data has substantial occlusions. A solution to this problem is to split hierarchies into multiple matrices to protect visible joints from being affected from occluded ones.

- **C3D (Coordinate 3D)**: C3D is a widely used format in biomechanics and sports research. It stores 3D coordinate data, as well as analog data (such as force plate data), making it suitable for analyzing complex movements.
- **CSM (Calibrated Systems Motion)**: CSM is used to store motion capture data from calibrated systems, preserving spatial relationships between markers. It is valuable for accurate motion analysis and research.
- **DAT, HTR, TRC, MOT, SKL**: These formats are used for storing motion capture data in various contexts, often specific to certain software or applications.
- **DAE (Collada)**: Although not exclusively a motion capture format, DAE is an open standard for exchanging digital asset information, including animations. It supports skeletal animation and is used in various 3D applications.
- **FBX (Filmbox)**: FBX is a proprietary format developed by Autodesk, designed to facilitate the exchange of motion capture data between different software applications. It supports animations, skeletal structures, and various other assets.

While these motion capture formats differ in their internal structures and data representations, they all capture and store human or object movements.

Motion capture formats find extensive applications across a wide range of industries, including movies, games, simulations, and more. Here are some specific applications within these domains:

- **Movies**:
  - **Character Animation**: Motion capture data is used to animate characters realistically in movies. By capturing the movements of actors or performers, animators can create lifelike characters that mimic human gestures, expressions, and actions.

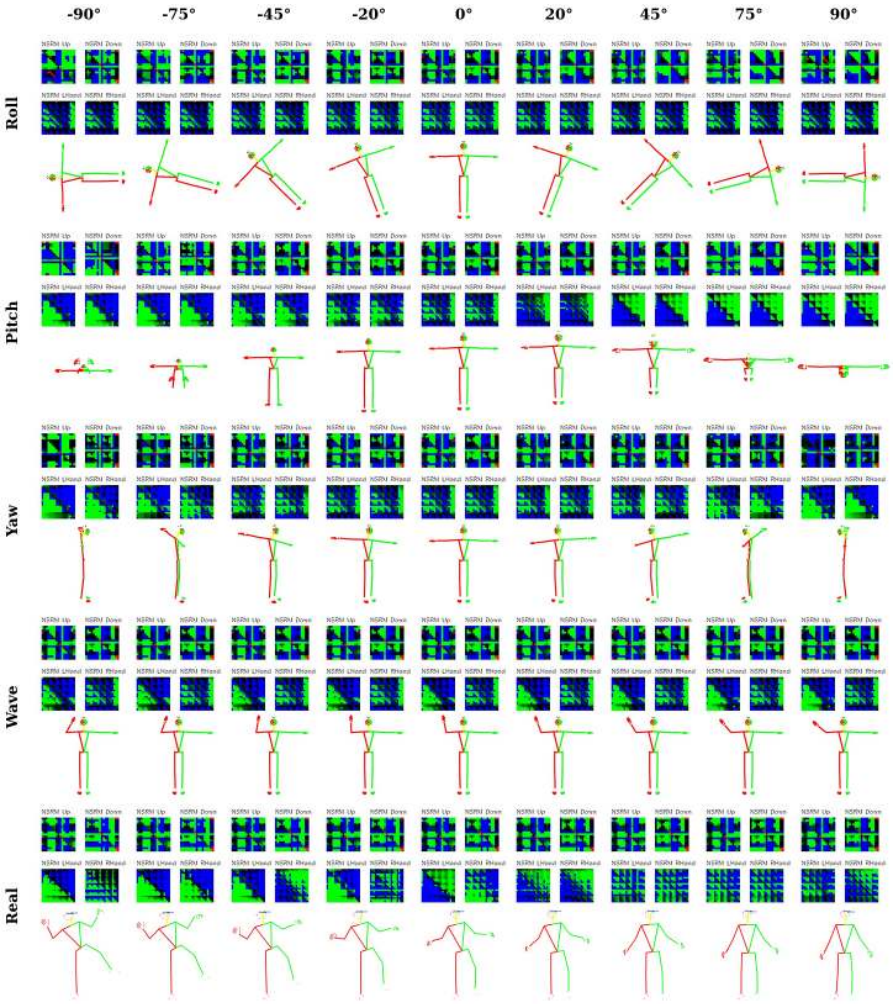


Figure 3.16: 2D symmetries can be leveraged to our advantage. By flipping points horizontally we can reduce pose space and handle multiple symmetric cases with the same NN capacity. The illustration shows horizontally flipped coordinates and the resulting flipped eNSRMs of Figure 3.9

  - **Creature Animation**: In addition to human characters, motion capture is employed to animate fantastical creatures and non-human entities, ensuring they move in a natural and convincing manner.
  - **Stunt Sequences**: Dangerous or physically demanding stunts can be captured safely using motion capture technology. This data is then integrated into scenes, reducing risks to stunt performers.
  - **Special Effects**: Motion capture can be used to drive special effects and visual elements, such as dynamic cloth, fluid simulations, and particle systems, enhancing the overall realism of a movie.

- **Games**:

  - **Character Animation**: In video games, motion capture formats allow for realistic and responsive character movements, contributing to immersive gameplay experiences.
  - **Sports Games**: Motion capture is crucial in sports simulation games, ensuring accurate representations of athletes' movements, gestures, and actions.
  - **Interactive VR/AR**: Motion capture enhances the realism of virtual and augmented reality experiences, allowing users to interact with digital environments and characters in a more natural way.
  - **Cutscenes**: In-game cinematics and cutscenes are often created using motion capture technology, adding cinematic flair to the storytelling.

- **Research**:
  - **Biomechanics Research**: Researchers use motion capture to study human and animal movements for medical, sports, and rehabilitation purposes. It aids in understanding body mechanics, improving prosthetics, and designing safer equipment.
  - **Training Simulations**: Military, medical, and industrial training simulations utilize motion capture to recreate real-world scenarios, allowing trainees to practice complex tasks in controlled environments.
  - **Robotics**: Motion capture formats assist in programming and refining the movements of robots, leading to more precise and human-like interactions.
  - **Gesture Recognition**: Motion capture formats can enable gesture recognition by computers, allowing direct HCI through physical movements and without an intermediate input device featuring keys or requiring the pressing of buttons.

- **Entertainment and Sports**:
  - **Live Performances**: Motion capture formats enable live performances where actors, dancers, and performers interact with digital environments or characters in real-time.
  - **Sports Analysis**: Coaches and athletes can use motion capture to analyze techniques and improve performance in sports like golf, tennis, and swimming.
  - **Concerts and Events**: Motion capture can enhance concerts and events by creating captivating visuals and dynamic stage effects that respond to performer movements.

- **Medical and Healthcare**:
  - **Rehabilitation**: Motion capture can assist in designing standardized personalized rehabilitation exercises for patients recovering from injuries or surgeries, ensuring proper form and movement.
  - **Gait Analysis**: Clinicians can use motion capture to assess and analyze gait patterns in patients, aiding in diagnosing and treating conditions related to walking and movement.
  - **Prosthetics and Orthotics**: Motion capture informs the development of prosthetic limbs and orthotic devices, creating more functional and comfortable solutions for individuals with limb differences.
  - **Ergonomics**: Motion capture can help assess ergonomic factors in workplaces, identifying potential sources of strain and injury for workers.

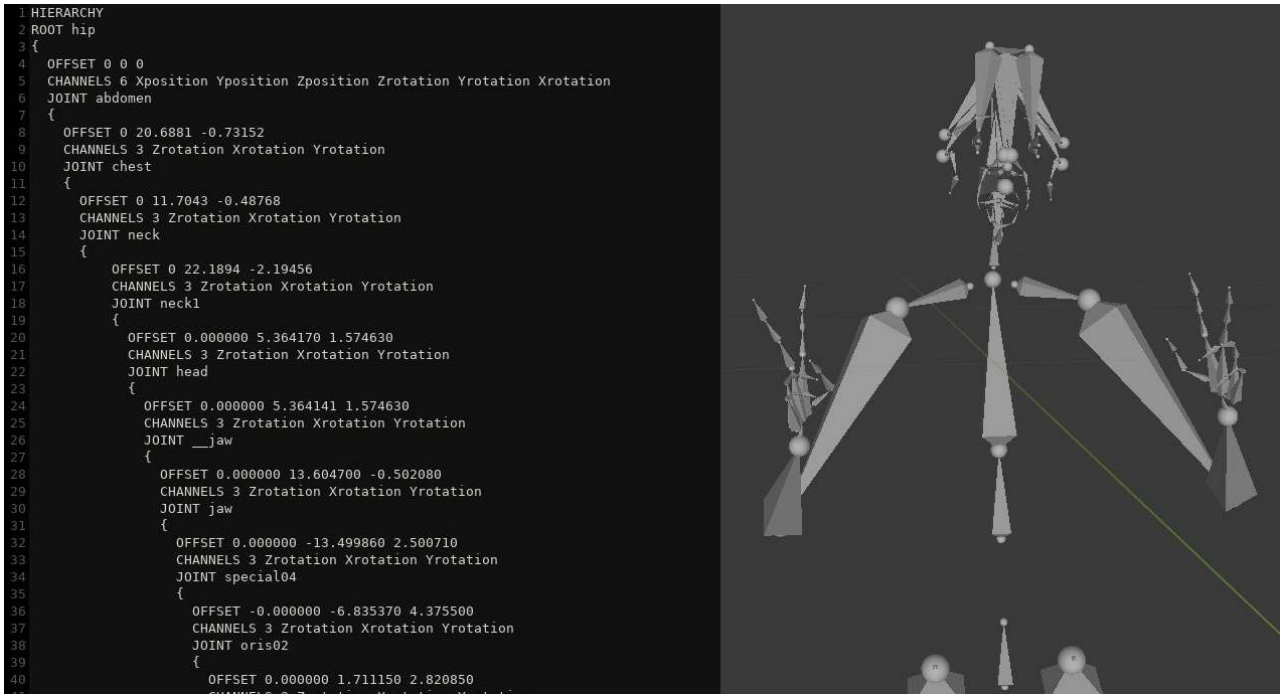
- **Art and Performance**:
  - **Dance and Choreography**: Motion capture enhances dance and choreography by capturing intricate movements that can be analyzed, studied, and integrated into artistic performances.
  - **Music Videos and Performances**: Motion capture is used to create visually stunning music videos and live performances with synchronized visuals and music.
  - **Art Installations**: Artists incorporate motion capture data into interactive installations, allowing viewers to engage with art through movement.
- **Anthropology and Social Sciences**:
  - **Cultural Studies**: Researchers use motion capture to document and study traditional dances and rituals, thus preserving cultural heritage.
  - **Behavioral Studies**: Motion capture aids in understanding human behavior, analyzing gestures, postures, and interactions in social contexts.
- **Fashion and Apparel**:
  - **Virtual Fitting**: Motion capture is employed to create virtual fitting rooms and enable customers to try on clothes virtually, enhancing online shopping experiences.

Motion capture formats are the backbone of these applications, allowing for the seamless integration of captured movements into various creative and analytical processes. Their versatility and adaptability make them indispensable tools in shaping the way we experience movies, games, simulations, and entertainment as a whole.

Keeping in mind all of the above information, our choice of MOCAP container format was influenced by openness, commercial relevance, developer popularity and implementation simplicity. At the time of writing the most popular 3D editing suites are 3D Studio Max, Maya and Cinema 4D. Currently the most popular game graphics engines are Unreal Engine, Unity and Godot while 3D human pose animation typically is built using Makehuman, Mixamo, Adobe Character animator, Poser, DAZ Studio or Motionbuilder. A very pervasive open-source 3D tool is Blender [21] which combines all of the above functionality into a free package with a multitude of plugins and extensions. Taking all of the above mentioned factors into consideration we selected the BVH format that is the most popular format despite also exhibiting some shortcomings we attempted to remedy with a small extension to the format that we will discuss in the next Section. What we managed to do is to create a NN that encodes 2D points directly in the specified output format in an end-to-end fashion a world-first.

3.3.1 The Bio-Vision-Hierarchy (BVH) Motion Capture Format

The BioVision Hierarchy (BVH) file format is one of the most widely used standards for storing skeletal animation data, especially in the field of computer graphics and motion capture. It was developed to facilitate the exchange of motion data between different software and hardware systems. The format was introduced in the early 1990s by a company named Biovision, which was involved in the production of motion capture systems. The format aimed to provide a simple and efficient way to store motion data captured from physical actors for later use in computer-generated animations. A BVH consists of two main Sections. The preamble stores a hierarchical structure of bones and joints with their offsets and degrees of freedom,



```
HIERARCHY
ROOT hip
{
  OFFSET 0 0 0
  CHANNELS 6 Xposition Yposition Zposition Zrotation Yrotation Xrotation
  JOINT abdomen
  {
    OFFSET 0 20.6881 -0.73152
    CHANNELS 3 Zrotation Xrotation Yrotation
    JOINT chest
    {
      OFFSET 0 11.7043 -0.48768
      CHANNELS 3 Zrotation Xrotation Yrotation
      JOINT neck
      {
          OFFSET 0 22.1894 -2.19456
          CHANNELS 3 Zrotation Xrotation Yrotation
          JOINT neck1
          {
            OFFSET 0.000000 5.364170 1.574630
            CHANNELS 3 Zrotation Xrotation Yrotation
            JOINT head
            {
              OFFSET 0.000000 5.364141 1.574630
              CHANNELS 3 Zrotation Xrotation Yrotation
              JOINT __jaw
              {
                OFFSET 0.000000 13.604700 -0.502080
                CHANNELS 3 Zrotation Xrotation Yrotation
                JOINT jaw
                {
                  OFFSET 0.000000 -13.499860 2.500710
                  CHANNELS 3 Zrotation Xrotation Yrotation
                  JOINT special04
                  {
                    OFFSET -0.000000 -6.835370 4.375500
                    CHANNELS 3 Zrotation Xrotation Yrotation
                    JOINT oris02
                    {
                      OFFSET 0.000000 1.711150 2.820850
```

Figure 3.17: Left: A BVH file preamble containing the hierarchy of Joints, their degrees of freedom rotation orders and offsets. Right: The skeleton visualization.

while the second defines a number of consequitive poses and their rate of change, representing the movements of a character's skeleton over time. As computer graphics and animation technologies advanced, the BVH format gained popularity due to its simplicity and compatibility with various animation software and platforms, gradually becoming a de-facto standard for sharing and transferring motion capture data. Over time, variations of the BVH format emerged, incorporating additional features and refinements while they became widely accessible and used in both professional and amateur animation communities. The open nature of the format allowed developers to create import/export plugins and tools for various software packages, promoting interoperability.

Despite their ubiquity due to their simplicity and wide support, BVH files however do have limitations, such as the lack of support for complex deformations, muscle simulations, direct vertex geometry bindings and other advanced animation techniques. As animation technology evolved, more sophisticated file formats emerged to handle these advanced features. Another issue with the format is the lack of explicit, skeleton and bone orientation declarations, while bone-sizes are not well defined in cases with multiple children joints. Another useful feature missing from the specification is scale information. Scale for BVH files is typically set to meters (due to the lack of such a header field), however in the case of our files, centimeters are used.

Despite the emergence of more advanced animation file formats however, BVH remains a commonly used format, particularly in cases where quick and straightforward motion capture data exchange is required.

A BVH file starts with the "HIERARCHY" string signaling the start of the hierarchy description. Each joint

Listing 2 Quaternion To 4×4 Matrix calculation in C

```
void c4x4FQuaternionMatrix(float * m,float qX,float qY,float qZ,float qW)
{
 float yy2 = 2.0f * qY * qY; float xy2 = 2.0f * qX * qY;
 float xz2 = 2.0f * qX * qZ; float yz2 = 2.0f * qY * qZ;
 float zz2 = 2.0f * qZ * qZ; float wz2 = 2.0f * qW * qZ;
 float wy2 = 2.0f * qW * qY; float wx2 = 2.0f * qW * qX;
 float xx2 = 2.0f * qX * qX;
 //------------------------------------------------------------
 m[0] =-yy2-zz2+1.0f; m[1] =xy2+wz2;       m[2] =xz2-wy2;       m[3] =0.0f;
 m[4] =xy2-wz2;       m[5] =-xx2-zz2+1.0f; m[6] =yz2+wx2;       m[7] =0.0f;
 m[8] =xz2+wy2;       m[9] = yz2-wx2;      m[10]=-xx2-yy2+1.0f; m[11]=0.0f;
 m[12]=0.0f;          m[13]=0.0f;          m[14]=0.0f;          m[15]=1.0f;
}
```

of the hierarchy has a string label associated with the joint and is declared as “JOINT name”. The root joint has a special invocation using ROOT instead of the JOINT description. After declaring a new node in the hierarchy its children are enclosed in curly brackets { } similar to many computer programming languages. Each joint declaration (including the root) has an “OFFSET x y z” statement that defines its distance from its parent. After defining the offsets a declaration of motion channels ensues that is typically “CHANNELS 6 XPosition YPosition ZPosition ZRotation XRotation YRotation” for the root joint in order to allow the armature to move in 3D space and “CHANNELS 3 ZRotation XRotation YRotation” for most children joints assuming constant, non deformable bone lengths. The order of rotation channels is very important since it defines the order of matrix multiplications and is often the cause of problems between different coordinate systems and linear algebra conventions. Our implementation extends the standard by adding a special rotation channel we call QRotation. When a QRotation is declared by e.g. a “CHANNELS 4 XRotation YRotation ZRotation QRotation” statement the rotation values are treated as the components to a Quaternion rotation. We use quaternions to define the root orientations of standalone 3D hands that can have any orientation, as a measure to avoid gimbal locks, while we believe they can be a minimal, but valuable contribution to the BVH standard in applications that deal with similar problems. Our addendum to the BVH format specification was implemented by adding a ”WRotation” channel to the hierarchy manifest, and implementing a quaternion to 4x4 transformation matrix conversion routine (Listing 2) in our BVH parser that enables quaternions to be handled by the BVH format. In the case of full body BVH our output is standard and directly compatible with third party software. For standalone hands we use the .qbvh extension to distinguish output from regular .bvh files since despite the trivial addition the proposed format can potentially cause incompatibilities with third party BVH loaders.

As already seen each BVH joint is defined in the context of its parent. A BVH joint however is not inherently SO(3) (see Table 3.1) invariant. Each joint in a BVH skeleton is typically represented by its local rotation, which is often described using Euler angles (e.g., pitch, yaw, roll) or other rotation representations like quaternions. Since Euler angles are not directly elements of the Lie group SO(3) (the group of 3D rotations), the representation of BVH joints is not strictly SO(3) invariant. This means that a direct comparison of rotations between two BVH joints may not yield consistent results due to the intrinsic limitations of Euler angles.

However, it is important to note that many applications that use BVH skeletons can perform further processing or normalization steps to make the joint rotations more consistent and invariant to certain transformations. For example, converting BVH rotations to quaternions or performing quaternion averaging can help mitigate some of the issues related to the non-invariance of Euler angles. Such post-processing steps can be useful when dealing with motion capture data or other animation-related tasks.

In some research or applications, people may refer to BVH-based methods as "SO(3) invariant" when they incorporate additional normalization or processing steps to make the BVH joint rotations more consistent or invariant to certain transformations. However, it is essential to understand the specific processing steps employed in each case to assess the level of invariance achieved.

3.3.2 Processing, Rendering, Filtering and Augmenting BVH Data

During the course of this PhD a C library [357] was developed to parse, process, filter and augment BVH datasets. The corpus for the original body pose data was the BVH conversion [16] of the Carnegie Mellon University Motion Capture Dataset [18]. As seen in Table 2.6 the CMU MOCAP dataset provides 3.9M samples, 473 actions from 144 subjects. Processing the poses involves calculating the individual rotation and translation matrices for each sample/frame and then performing the forward kinematic transformations by multiplying the generated 4×4 matrices.

Assuming a target camera system we can emulate a virtual camera by computing a $Projection_{Matrix}$ with a known set of intrinsics. Our method by default emulates a Full-HD GoPro Hero 4 camera, with the following projection parameters acquired after calibration using the OpenCV framework:
$width = 1920, height = 1080, c_x = 960, c_y = 540, f_x = 582.18394, f_y = 582.52915, far = 10000.0, near = 1.0$

$$Projection_{Matrix} = \begin{bmatrix} -2f_x/(R-L) & 0 & 0 & 0 \\ 0 & 2*f_y/(T-B) & 0 & 0 \\ 2*c_x/(R-L)-1 & 2*c_y/(T-B)-1 & -(F+N)/(F-N) & -1 \\ 0 & 0 & -2F*N/(F-N) & 0 \end{bmatrix}$$

The initial "DAZ-friendly" BVH release of CMU's motion capture database [17] compiled by Bruce Hahne was designed for use with the 3rd-generation and 4th-generation prerigged characters from Daz3d.com and featured 43 joints, while other BVH conversions of the data like the Motionbuilder and 3D Studio Max, an even lower count armature with just 31 joints. Using the opensource Makehuman skeleton definitions [23] and our BVH library we appended the armatures with a fully animatable head, hands and feet. Our higher detail total capture skeleton consists of 165 joints including the root [16], retains compatibility with its baseline while offering extensions to accommodate facial and hand capture.

A simple BVH armature example is a hierarchical chain spanning just the following 4 joints: Hip → RHip → RKnee → RFoot. For each motion frame, calculating the 3D coordinates involves calculation of each $Joint_{Matrix}$ and then performing chain multiplication of the hierarchical chain, e.g. $Hip_{Matrix} * RHip_{Matrix} * RKnee_{Matrix} * RFoot_{Matrix}$.

$$R_X = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & cos(x) & sin(x) & 0 \\ 0 & -sin(x) & cos(x) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, R_Y = \begin{bmatrix} cos(y) & 0 & -sin(y) & 0 \\ 0 & 1 & 0 & 0 \\ sin(y) & 0 & cos(y) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

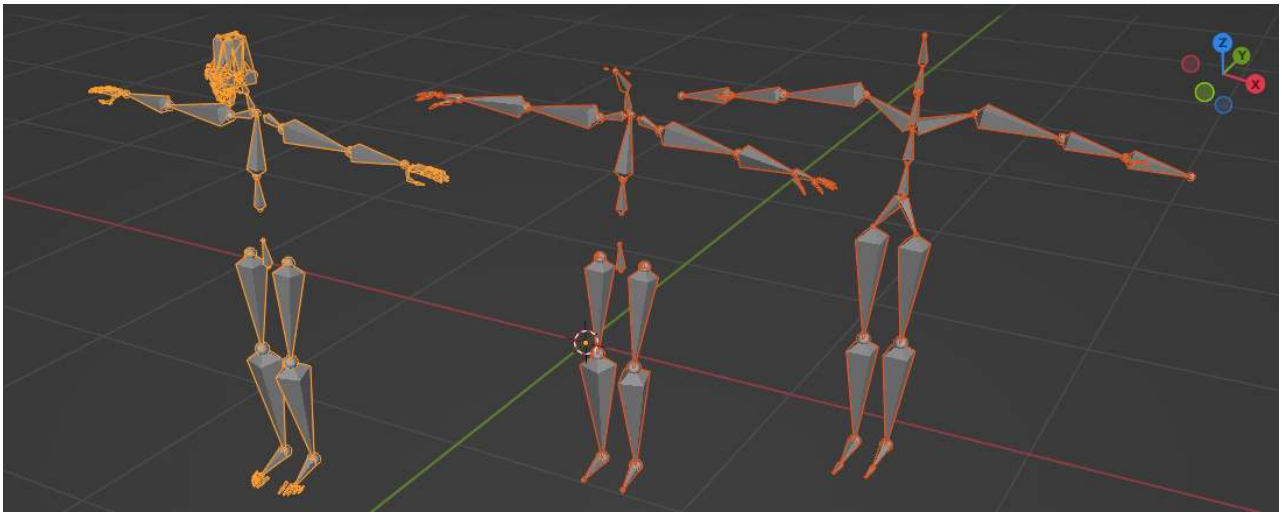


Figure 3.18: Comparison of different fidelity BVH skeletons. Left: Our proposed armature with 165 joints [16] that includes accommodations for facial, hand and feet controls. Middle: The DAZ-Friendly armature of [17] with 43 joints we use as the basis for our skeleton. Right: The 3DS Studio Max/Motionbuilder armature is even simpler with just 31 joints.

$$R_Z = \begin{bmatrix} cos(z) & sin(z) & 0 & 0 \\ -sin(z) & cos(z) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$Rotation_{ZXY} = R_Z * R_X * R_Y,$$

$$Rotation_{XYZ} = R_X * R_Y * R_Z,$$

$$Rotation_{ZYX} = R_Z * R_Y * R_X,$$

... etc. depending on Rotation order ...

$$Translation_{Matrix} = \begin{bmatrix} 1 & 0 & 0 & t_X \\ 0 & 1 & 0 & t_Y \\ 0 & 0 & 1 & t_Z \\ 0 & 0 & 0 & 1 \end{bmatrix}, Joint_{Matrix} = \begin{bmatrix} R & R & R & T_X \\ R & R & R & T_Y \\ R & R & R & T_Z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$Joint_{Matrix} = Rotation_{ZXY} * Translation_{Matrix}$$

The final projection to labeled 2D points in window coordinates is calculated using the glhProjectf function in Listing 3. We can thus convert the BVH files to 3D points and 2D points and create the required training samples for the method.

Assuming the classic pinhole camera model [358] with the k_1, k_2, k_3 for radial and p_1, p_2 for tangential distortion coefficients we can optionally further process our rendered 2D points to achieve an even more

representative dataset distribution that matches a specific camera model. Assuming that the undistorted points are x_{2D}, y_{2D} and the distorted 2D points are x^d_{2D}, y^d_{2D}. The distortion function formulation is the following:

$$r^2 = x^2_{2D} + y^2_{2D}$$

$$x^d_{2D} = x_{2D}(1 + k_1 * r^2 + k_2 * r^4 + k_3 * r^6) + 2 * p_1 * x_{2D} * y_{2D} + p_2 * (r^2 + 2 * x^2_{2D})$$

$$y^d_{2D} = y_{2D}(1 + k_1 * r^2 + k_2 * r^4 + k_3 * r^6) + 2 * p_2 * x_{2D} * y_{2D} + p_1 * (r^2 + 2 * y^2_{2D})$$

The initial version of our method [6] used subjects 1-19 with an aggregate $\pm 484K$ training samples due to memory constraints in our initial implementation. The subsequent works [8, 9] used the full CMU MOCAP database.

Studying the BVH files by projecting them on a 2D accumulator surface reveals problems in dataset uniformity as seen in Figure 3.19. The causes are multiple. The BVH files have some degree of repetition because they are recorded in high frame rate (120 Hz) and because each action is repeated several times . This skews the training procedure as it overemphasizes overrepresented poses against some interesting, under-represented ones. Thus, we constructed a filter that omitted repeated poses of the original dataset (see Figure 3.20) and discarded approximately 30% of the poses where all joints where clustered around the same positions, leaving approximately 2.2M training poses. This was another improvement compared to the baseline [6] method that, due to no dataset pose filtering, had to be trained on a much smaller selection of CMU actions leading to poorer overall training pose diversity. The clustering tool is available in the github repository [65].

Without filtering, the mini-batch stochastic gradient descent [359] algorithm used for neural network training randomly samples a number of poses for each mini-batch and weight update to perform back-propagation. By having some poses being extremely overrepresented the training procedure becomes skewed and is prone to over-fitting due to frequent updates of the weights towards biased samples. To make the situation worse, interesting and exotic poses of the dataset that we would ideally like to learn become disproportionately under-represented.

The dataset also depicts persons performing actions in fixed trajectories, so to further enrich it we need to augment it by randomizing the location of the observed skeleton for each frame. Directly randomizing the 3D coordinates of the skeletons leads to sub-optimal 2D coverage of the input frame. Instead, we pick a random 2D point on the virtual camera frame and then pick a random depth value to create our randomized point. This way the randomization covers more uniformly the whole view frustrum and gives more opportunities to the neural network to learn about how perspective projection affects 3D points.

To further enrich the dataset we created uniform random 2D image locations $P(x_i, y_i)$ and depth values d_i where $0 \leq x_i \leq W$, $0 \leq y_i \leq H$ (in pixels) and $1000 \leq d_i \leq 5500$ (in mm), assuming a video frame size equal to $W \times H$. With these random 2D points and depths and assuming a camera C with known intrinsic parameters W, H, f_x, f_y, c_x, c_y we can randomize the skeleton positions of the dataset at 3D points (X_i, Y_i, Z_i) where $X_i = (x_i - c_x) \cdot d_i / f_x$, $Y_i = (y_i - c_y) \cdot d_i / f_y$ and $Z_i = d_i$. This first data augmentation allows MocapNET to learn, how constellations of 2D points are affected by 3D translations.

A second data augmentation procedure diversifies the recorded 3D joint configurations. The perturbations use uniform random values so that the new value is at most $\pm x^\circ$ away from the original orientation. We perturb the r/l shoulder by $\pm 30^\circ$, r/l elbow $\pm 16^\circ$, abdomen and chest by $\pm 10^\circ$, r/l hip $\pm 30^\circ$ and r/l knee

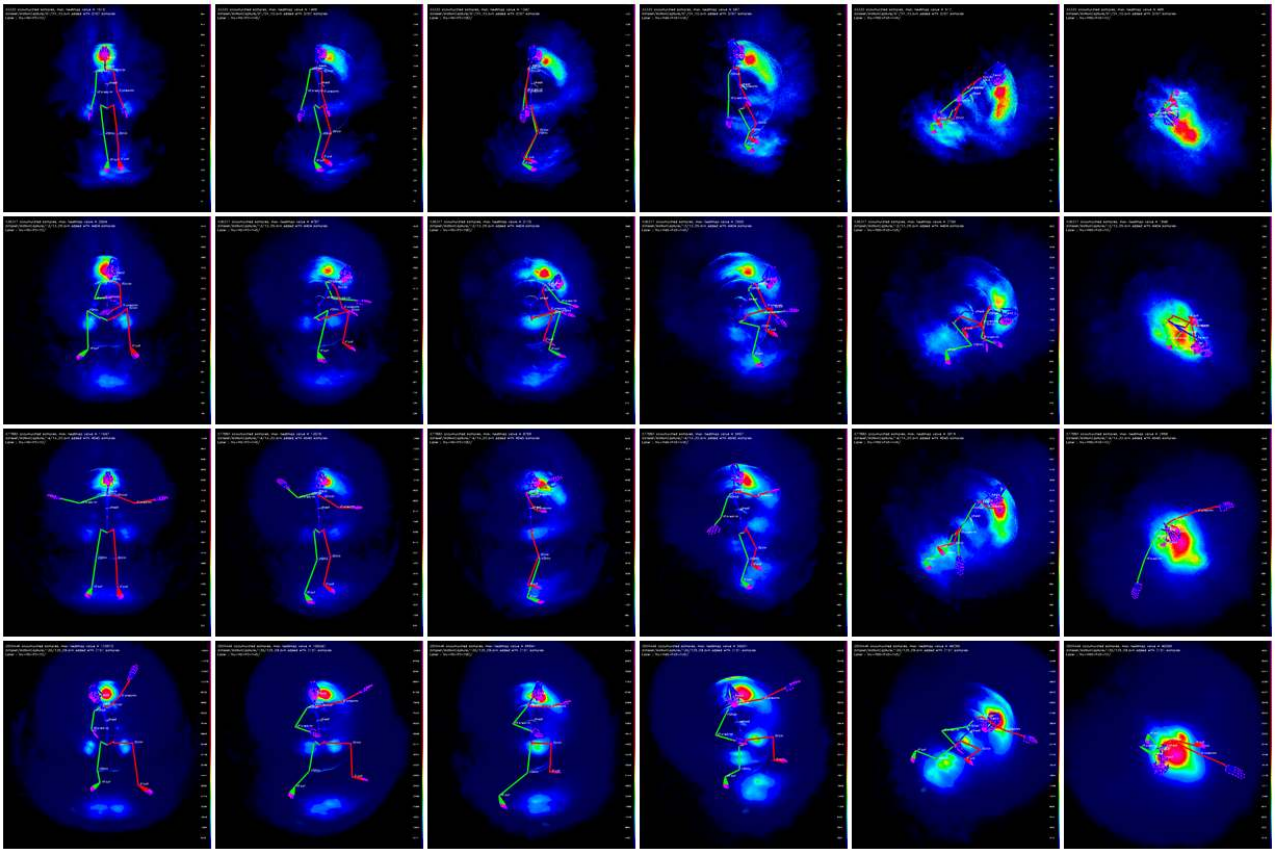


Figure 3.19: Heatmap accumulation from different vantage points over 3.9M poses of the BVH conversion [16] of the CMU MOCAP dataset [18]. We discard the translational and rotational component of the skeleton and plot joints on a 1000×1000 pixel array. First row after 33K samples, Second row 136K, Third 944K, Fourth 3.4M. We observe a large cluster of repeated poses. We attribute them to the time between sessions before actors begin moving for the recording and after they stop.

$\pm 10^{\circ}$. Larger perturbations could be employed but are not used since we don't want our perturbations to violate the physical kinematic constraints of the depicted motions captured using the MOCAP suit which obey the laws of physics, do not interpenetrate and intersect with the body and are a very desirable trait of our dataset.

For each randomized pose we simulate self-occlusions during dataset preparation, by projecting points onto a virtual camera C and then applying a depth ordering pass that erases joints hidden behind the torso or near the radius of other joints (right example of Figure 3.5). After these preparations and although our data still essentially contains the same Motion Capture seed from CMU, our dataset acquires self-occlusions and becomes translation and rotation randomized.

We can repeat the above randomization process by adding every input pose multiple times, perturbed differently, to create more training samples that offer a richer source of rotation and translation exemplars. For the second publication of our method [8] we repeated each sample 3 times under a different perturba-

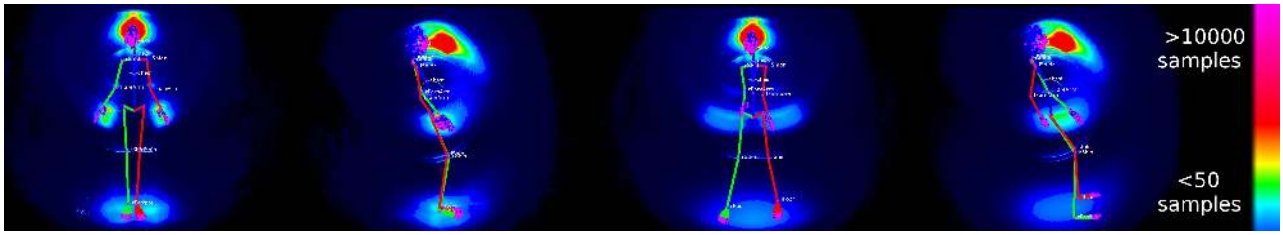


Figure 3.20: Joint location heatmaps of the 3.9M poses of the CMU dataset [16, 18] after translation and rotation normalization. Left: frontal and side illustration of the accumulated joint frequencies in the raw dataset. Right: the same information, after dataset filtering and augmentation.

tion to make maximum use of our available GPU memory.

The employed randomization scheme can result in poses where joints fall out of the bounds of the camera. We detect and omit those from our training set. The final training set amounts to approximately $1.5M$ and up to $3.5M$ training poses per orientation class depending on the number of repetitions.

Finally, we perform randomization of the orientation of the human skeletons. In order to deal with ambiguities due to symmetries and distinguishing very different poses our first works partitioned pose space randomization into different orientation classes.

As already mentioned our first implementation started as a orientation classifier with the ability to robustly identify if the orientation of an observed skeleton was frontal or backwards. Integrating this classifier in the first MocapNET work [6] meant that each NN relied on the classifier for orientation and then performed regression on a smaller pose space. The subsequent version [8] extended classification to front, back, right or left as seen in Figure 3.40. Taking this into consideration, we were able to offload some of the regression task to the classifier and split the randomized poses into three different classes thus relieving the NN from the more difficult task of handling all orientations at once. Therefore for [8], we split randomization in four quadrants (front, back, left, right). All quadrants had the same limits for rotations on the x and z axis ($-35^\circ \leq r_x \leq 35^\circ$ and $-35^\circ \leq r_z \leq 35^\circ$). The orientation r_y was split into overlapping quadrants of 100° each, to ensure proper handling in the case of marginally inexact orientation classification. This class separation scheme not only allowed smaller, more accurate and higher-performance neural networks with an easier task to accomplish, but also mitigated neural network learning problems due to angle discontinuities [164].

Assuming that after the filtering and augmentation processes we yielded $\sim 2.2M$ poses per orientation class. This meant that the final "ensemble of ensembles" would be trained with over $8.8M$ poses, however, without needing to generalize to all of them at once or contain them all in RAM during training.

Splitting the pose space in four orientations seemed to be optimal for the problem. This is not only the case in literature [44] long predating our work but is logical, since as we introduce more orientation categories the orientation classifier gets more and more encumbered eventually having to handle all of the pose complexity (which is exactly what we are trying to avoid in the regression NN). Furthermore segmentation of the space using more than 4 categories becomes not just a problem for the classifier but for the ensembles, since as they become more specialized and trained in a narrower orientation range they will inevitably start to overfit and fail when receiving input that does not strictly conform to their trained samples. Furthermore training times scale linearly with more orientation classes making it impractical and infeasible as we

will later see when further splitting the body into sub-hierarchies for different regions.

After processing the dataset we find a 187 out of 2535 BVH files to be corrupted with limbs bending the wrong way, so we manually discarded incorrect files The following list uses the notation of "Subject/Action" and details all the identified corrupted files. The corrupted data are: 22/12 23/24 27/11 28/04 28/08 28/09 28/12 28/14 28/16 28/17 28/19 29/23 30/03 30/13 30/19 31/11 31/15 31/17 31/19 31/20 32/10 33/02 34/02 35/25 35/27 49/12 49/16 49/17 49/18 49/21 49/22 54/07 54/26 55/05 55/11 55/14 55/24 55/25 60/10 69/15 74/07 74/08 74/13 75/16 75/20 77/16 77/17 78/21 80/18 80/59 81/12 81/13 81/14 82/05 83/35 83/43 83/47 83/50 83/62 83/68 84/03 84/04 84/07 84/08 84/13 87/03 87/04 87/05 88/05 88/11 89/03 90/01 90/03 90/04 90/10 90/19 90/20 90/21 90/24 90/25 90/29 90/31 90/34 90/35 90/36 93/04 102/19 103/04 104/01 104/06 104/10 104/16 104/18 104/37 104/53 104/55 104/57 104/04 104/09 104/13 104/17 104/26 104/48 104/54 104/56 106/08 106/09 106/15 106/17 106/20 111/06 111/07 111/08 111/16 111/21 111/32 111/33 111/37 111/38 113/12 113/22 114/16 117/01 117/04 121/01 121/02 121/03 121/04 121/08 121/10 121/11 121/13 121/19 121/20 121/21 121/24 121/28 122/06 122/22 122/31 122/34 122/35 122/43 122/47 122/50 122/57 122/62 122/64 122/68 127/17 127/21 127/36 137/11 137/12 137/30 138/01 138/02 138/03 138/04 138/05 138/06 138/07 138/08 138/09 138/10 138/31 139/16 139/17 139/18 140/02 140/03 140/04 140/08 140/09 141/08 141/09 141/11 141/14 141/15 141/17 141/20 143/19 143/23 144/05 144/06 144/09 144/10. Identifying them was difficult since the error appears to be caused by symmetries on the depth axis, so looking at the wireframe armature poses appeared to be correct due to having the same valid 2D projections.

3.3.3 The Different Levels of Supervision for Dataset Generation

Using a pre-recorded dataset like [16] with specific actions gathered to represent real-world motions falls into the paradigm of Supervised Learning (SL). Even all of our extra steps with perturbations, randomizations that could be categorized as semi-supervised due to being essentially unlabeled, are in fact strong supervision towards the goal of elaborate category splitting, manual determination of the structure of the problem, bias reduction and increasing the variance of data and thus the robustness of the learned-function. Achieving the best possible training outcome with the very small NN ensemble of our first two works [6, 8] meant closely supervising all of the minute details of the formulation and data preparation.

The common BVH format representation for input and output we presented could allow a form of "Reinformenent Learning (RL)" after being coupled with the generative optimizer we will introduce in later Sections. Dealing with in-the-wild unlabeled input poses our regression module could monitor incoming poses. Comparing the 2D reprojection accuracy of the NN ensemble solution against the observation versus the accuracy of the fine tuning regression module, we can identify input cases where the NN performs poorly. Having a generative fine tuned solution along with the means to create a new synthetic reprojection, we can register additional training samples for poses where the neural network performs poorly. Iterative training sessions should gradually incorporate these changes thus allowing the NN to provide answers closer to the observation. With a better starting position closer to the solution, the fine tuning module would further improve guiding the process to converge to a higher quality result. Although this capability was implemented we ended up not using it since by blurring the lines between training set and R/L set it would make conducted experiments and published papers much more complex and harder to thoroughly examine and justify. Exposing the method to more and more similar data to what is tested against would be akin to mix test samples in the training set. This was one of the main reasons for our decision not to perform this RL data generation technique. An additional concern was also that dealing with our ill-posed problem 2D to 3D problem that contained multiple symmetric solutions as already elaborated meant that if the fine

tuning module converged on a symmetric solution this (besides being technically correct in terms of the 2D projection) would disrupt solutions making them more prone to poses out of the realistic training corpus of the initial CMU dataset [16].

Figure 3.21 contains histograms of specific degrees of freedom of the problem to provide insight on their distributions and our processing. The Figure clearly showcases the effect of our root position and rotation randomization along with the effect of our perturbations and filtering that lowers the peaks of the distributions.

3.3.4 Semi-supervised Dataset Generation for Hands

In contrast to the baseline method [6, 8] that only targeted the body and employed BVH MOCAP [16] from the Carnegie Mellon dataset [18] we could not identify a similar BVH source for hands. In addition, research shift towards weakly/semi supervised methods [40, 360, 361], the need for dataset filtering [8] and works that use MOCAP data just to extract rotation limits [302] prompted a weakly supervised approach. Studying the literature we found anatomically correct hand [362] and wrist [363] kinematic models which we could enforce in our random pose generator. The baseline [8] relied on $2.2M$ samples for each ensemble, amounting to $8.8M$ samples covering all orientations. We used these sizes as reference for our synthetic dataset for hand training. After several tests we end-up with the following randomization scheme.

All added hand poses featured randomized 3D positions and rotations using the same formulation as the method for the body. We added $2M$ fully random poses, $200K$ with all finger d.o.f. set to zero and $600K$ with a naturally open hand and no finger movement. Finally each finger gets $400K$ uniform random poses with all other fingers open and $400K$ where the rest of the hand is closed. This randomization scheme emphasized learning of 3D orientation, ensured all fingers receiving equal training samples and yielded a total of $\approx 7.2M$ samples, a number close to both to the baseline sample number and our technical CPU/GPU memory limitations. Since 3D hand poses have a much larger variety of 3D orientations, our attempts to partition pose space lead us to use an Icosahedron (see Figure 3.42) with each of its faces acting as the center of a class. Unfortunately multiplying training times by a factor of 20 for each view was infeasible in the course of this PhD. After performing some very limited experiments we thus resorted to a single class containing all orientations and effectively abolishing the classifier from our formulation. Figure 3.22 shows histograms with the distributions for specific degrees of freedom. We observe that XYZ positions are similar to the baseline body distributions (Figure 3.21). Quaternion rotations however exhibit a different distribution with the W Rotation departing from the distribution of the X,Y and Z quaternion components. Finally fingers exhibit a pattern with many samples on the edges of the motion and much fewer samples in the intermediate positions. Although at this point our dataset generation contained no real samples recorded and orientation classification was discontinued having the dataset generated using the above stated rules made this dataset generation semi-supervised, since samples where mostly generated automatically.

3.3.5 Sobol Sampling for Dataset Generation

The final dataset generation technique used during this work was using Sobol sequences [19].

Sobol sequences, introduced by the Russian mathematician Ilya Sobol in 1967, are a type of low-discrepancy sequence used in numerical analysis and computational mathematics for various applications, including numerical integration, optimization, and Monte Carlo simulations. These sequences provide more uniform and even distribution of points in higher-dimensional spaces compared to traditional random sequences.

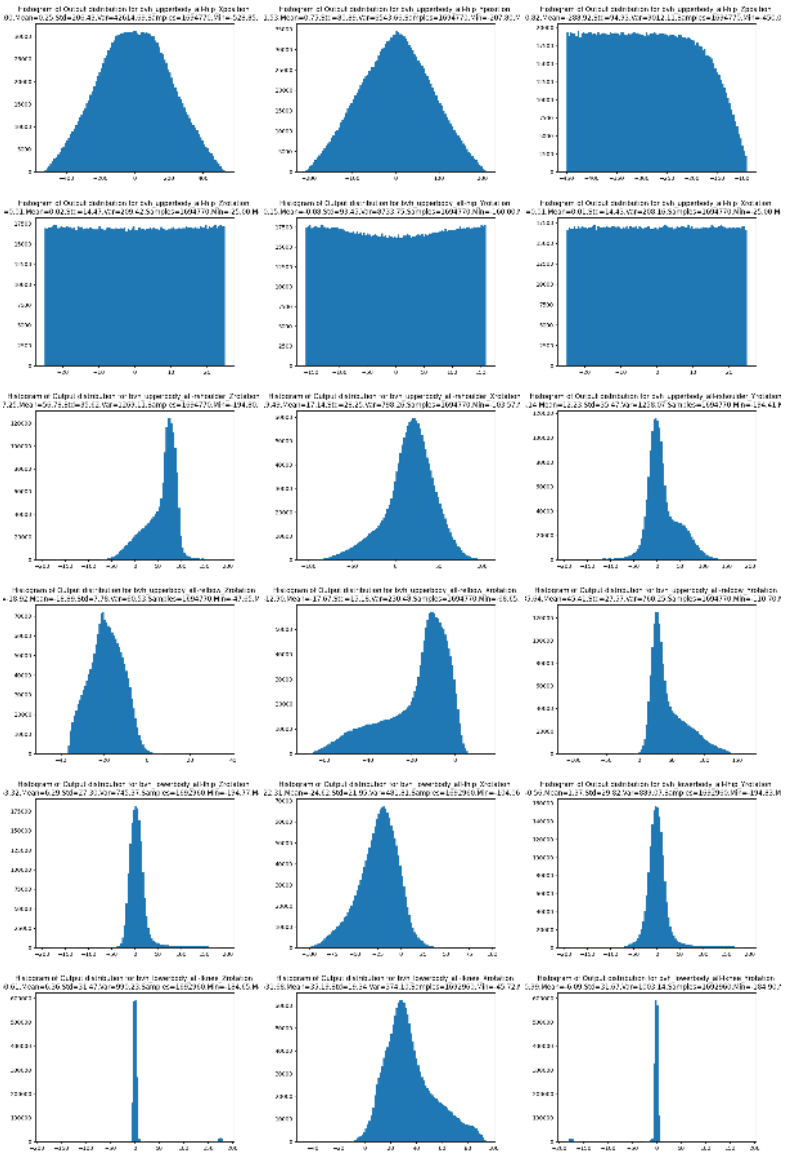


Figure 3.21: Training sample distribution for CMU-BVH [16] with our additions and perturbations. First row: XYZ root position. Second: ZYX root rotation. Third: ZXY RShoulder rotation. Fourth: ZXY RElbow rotation. Fifth: ZXY LHip, Sixth: ZXY LKnee (1 d.o.f active). We smooth distributions corresponding to realistic motions of MOCAP recorded subjects.

They are constructed using a base of two to form successively finer uniform partitions of unit intervals and then reorder the coordinates in each dimension. This way ensures that each new point added to the sequence covers a previously unexplored region of the space. This property makes them particularly useful for reducing the variance and error in numerical simulations, which is especially valuable when dealing with high-dimensional problems due to the curse of dimensionality.

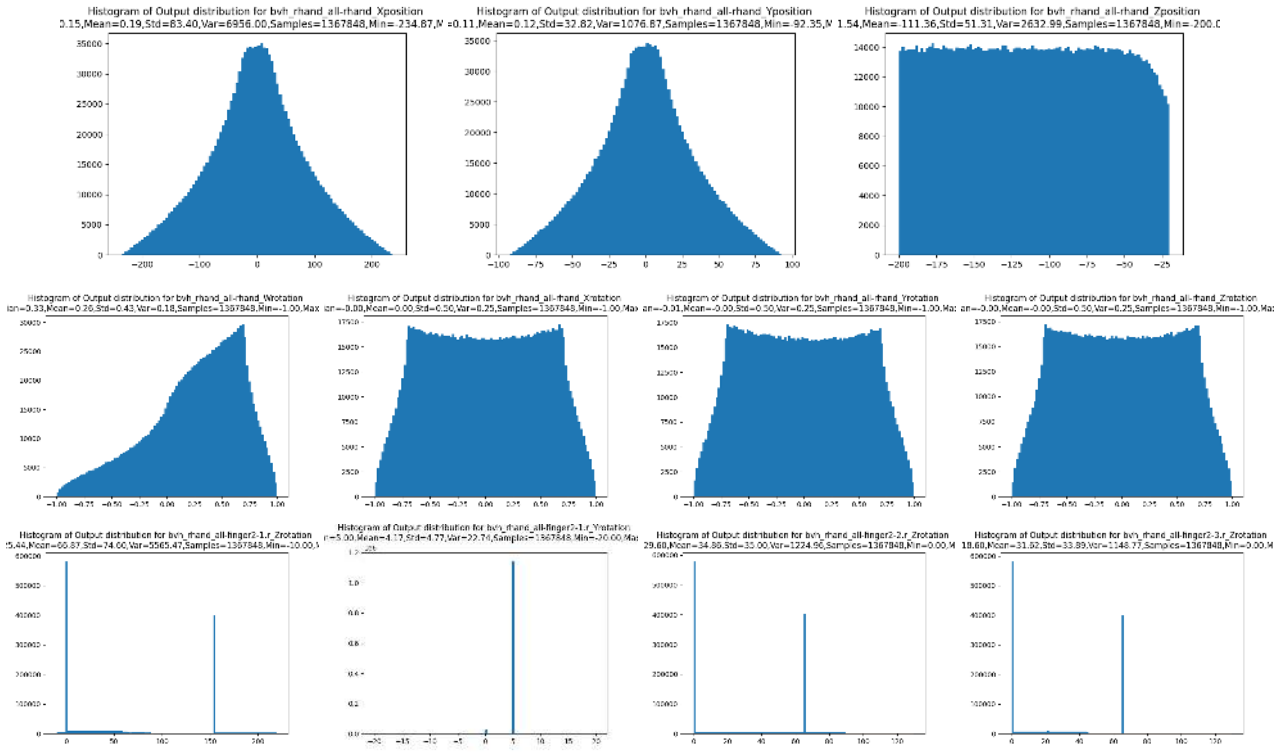


Figure 3.22: Distribution of hand samples generated by our initial randomization scheme. First row: X,Y and Z root position. Second row: q_W, q_X, q_Y and q_Z root rotation quaternion components. Third row: articulation angles for pointer finger (all other fingers are similar) with larger clusters on open/closed positions. Our initial [9] randomization scheme focused on extreme configurations of hands.

Although we kept the randomization position/rotation code to account for poses outside of the view frustum of our virtual camera (since the X,Y limits out of the field of view are different the closer an object is to the camera due to projective geometry, and Sobol sequences have no provisions for such tasks) the difference between Sobol sampling and our previous randomization scheme is very visible in the third rows of Figures 3.22 and 3.23. Not only that but the distribution of values is vastly superior to the results of the C "rand()" function and "/dev/random" Linux Kernel cryptographically secure pseudorandom number generator. Similarly even hardware extensions like the RDRAND instruction of recent Intel and AMD CPUs which is focused on a high-entropy are inferior to Sobol Sampling since our goal is to effectively sample the very high-dimensional space and rather than generating a sequence that is hard to predict, which is not a consideration for us.

Sobol sequences [19] ensured a perfect quasi-random sampling distribution when the number of samples $N = 2^x$ was initially for x up to 51 and in recent implementations up to 1111 dimensions. As a result of their formulation their sample sizes are powers of 2. Depending on the exponent however sample numbers grow exponentially which is a problem given the limited available GPGPU memory. Typical maximum values for a GeForce GTX 1070 GPU with 8 GBs of VRAM where $2^{21} = 2097152$ samples, while subsequent experiments on a server with a RTX A6000 GPUS totalling 48 GBs of VRAM allowed experiments

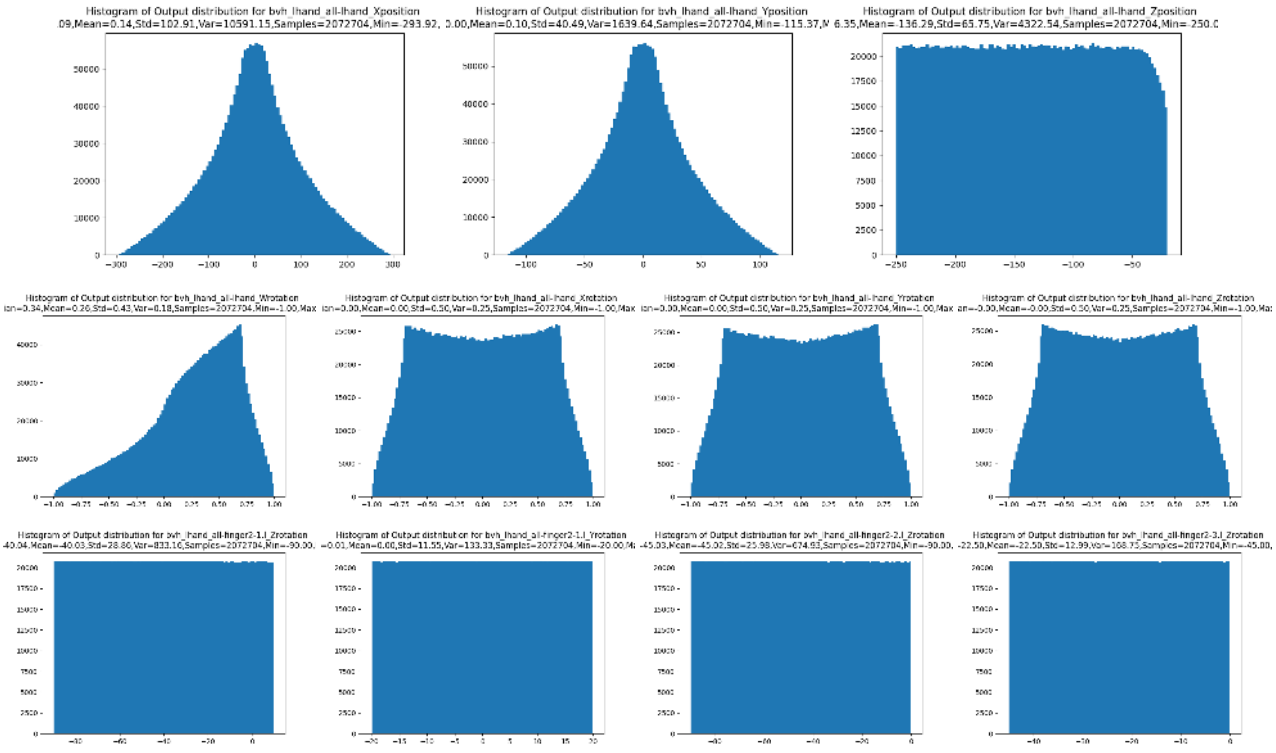


Figure 3.23: Distribution of hand samples partly generated by Sobol sampling [19] and 2^{21} samples. First row: X,Y and Z root position. Second row: q_W, q_X, q_Y and q_Z root rotation quaternion components. Third row: quasi-random perfectly uniform articulation angles for pointer finger (all other fingers are similar) generated by [19].

with $2^{23} = 8388608$ samples. Due to the exponential number of samples and limited available GPU NN training resources, picking a correct value for x was an important decision. Assuming a 3D armature of dimensionality D, in order for example to be able to sample 2 distinct configurations for each dimension, we would assume that we would need $N = D^2$ samples. Thus, the "allocated" number of samples per dimension N_{dim}, assuming perfectly equidistant samples obtained using Sobol sequences [19] should be $N_{dim} = N^{(1/D)}$. Taking into consideration our ensemble, for hands that have 23 degrees of freedom (disregarding the root 7 d.o.f. which we handle using the randomization scheme discussed earlier) and $2^{23} = 8.3M$ samples we get $N_{dim} = 2$. Even using a vastly larger sample number like $2^{24} = 16.7M$ samples would barely make a difference to $N_{dim} = 2.06$, which is only marginally larger while requiring 34.3GB of RAM to accommodate the involved training samples.

Using this thought process we created $N = 2^{20} = 1M$ samples for the R/L eye which has 12 output dimensions we get $N_{dim} = 3.17$ unique samples per dimension which roughly means that partitioning their motion range in three non overlapping regions their min/max and mean areas should be covered. Mouth controls occupy 18 DoFs and thus for $N = 1M$, $N_{dim} = 2.16$ which gives less resolution, however still adequately covers our training space. We should keep in mind that using Sobol Sequences samples always uniformly cover the whole range of values for each dimension so the N_{dim} metric is not a hard-limit but rather used as a tool

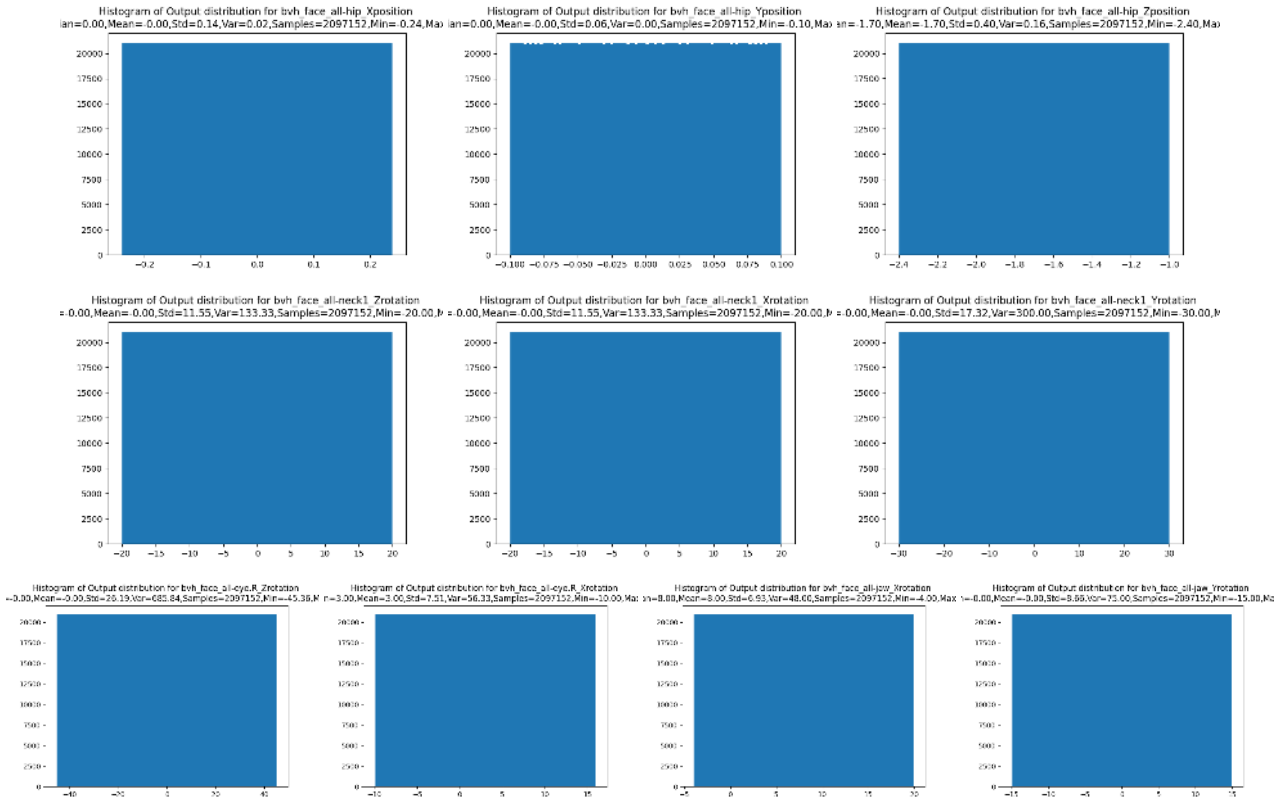


Figure 3.24: Distribution of facial samples fully generated by Sobol sampling [19] and 2^{21} samples. First row: X,Y and Z root position. Second row: Z, X, Y neck rotation. Third row: Eye gaze rotations and Jaw rotations. We observe perfectly distributed samples.

to conceptualize and quantify the effective number of unique samples per dimension. Facial distributions for the root position/rotation and the eye/jaw rotation can be seen in Figure 3.24.

A final way to study our dataset generation techniques is via the use of covariance matrices. A covariance matrix is a mathematical representation of the relationships between multiple variables in a dataset. Covariance matrices are widely used in statistics, machine learning, and multivariate analysis to understand the relationships between variables, identify patterns, and make informed decisions based on the interactions among different data points. It provides information about how pairs of variables change together, indicating the degree of linear dependence or correlation between them. It gives insights into the direction and strength of the relationships among variables with multiple ways.

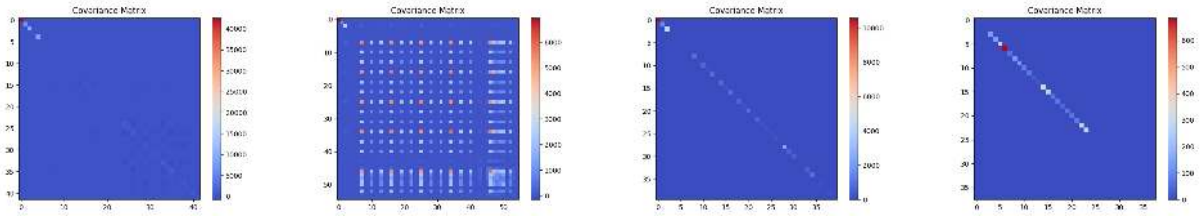


Figure 3.25: Covariance matrices for the different dataset generation/randomization schemes employed in the presented work. Left to right: (1) Body pose [6, 8], (2) Original hand randomization [9], (3) Improved hand randomization using Sobol sequences [20], (4) Facial randomization using Sobol sequences [20].

Listing 3 3D point to 2D window coordinate projection function in C

```
int glhProjectf(float * position3D,float *modelview,float *projection,
                int *viewport, float *windowCoordinate)
{
   int result = 0;
   float objX=position3D[0],objY=position3D[1],objZ=position3D[2];
   //Transformation vectors
   float fTmp[8];
   //Modelview transform
   fTmp[0]=(modelview[0]*objX)+(modelview[4]*objY)+
           (modelview[8]*objZ) +modelview[12];  //w is always 1
   fTmp[1]=(modelview[1]*objX)+(modelview[5]*objY)+
           (modelview[9]*objZ) +modelview[13];
   fTmp[2]=(modelview[2]*objX)+(modelview[6]*objY)+
           (modelview[10]*objZ)+modelview[14];
   fTmp[3]=(modelview[3]*objX)+(modelview[7]*objY)+
           (modelview[11]*objZ)+modelview[15];
   //Projection transform, the final row is always [0 0 -1 0]
   //so we optimize for that.
   fTmp[4]=(projection[0]*fTmp[0])+(projection[4]*fTmp[1])+
           (projection[8]*fTmp[2]) +(projection[12]*fTmp[3]);
   fTmp[5]=(projection[1]*fTmp[0])+(projection[5]*fTmp[1])+
           (projection[9]*fTmp[2]) +(projection[13]*fTmp[3]);
   fTmp[6]=(projection[2]*fTmp[0])+(projection[6]*fTmp[1])+
           (projection[10]*fTmp[2])+(projection[14]*fTmp[3]);
   fTmp[7]=-fTmp[2];
   //The result normalizes between -1 and 1
   if(fTmp[7]!=0.0) //The w value
   {
      fTmp[7]=1.0/fTmp[7];
      //Perspective division
      fTmp[4]*=fTmp[7];
      fTmp[5]*=fTmp[7];
      fTmp[6]*=fTmp[7];
      //Window coordinates
      //Map x, y to range 0-1
      windowCoordinate[0]=(fTmp[4]*0.5+0.5)*viewport[2]+viewport[0];
      windowCoordinate[1]=(fTmp[5]*0.5+0.5)*viewport[3]+viewport[1];
      //This is only correct when glDepthRange(0.0, 1.0)
      windowCoordinate[2]=(1.0+fTmp[6])*0.5; //Between 0 and 1
      result = 1;
   }
   return result;
}
```

The diagonal elements of a covariance matrix represent the variances of individual variables. A variance indicates how much a single variable deviates from its mean, giving an idea of the spread or dispersion of the variable's values. Specifically, each entry in a covariance matrix represents the covariance between two variables. Covariance is a measure of how much two variables change together. A positive covariance indicates that when one variable increases, the other tends to increase as well, and when one decreases, the other tends to decrease. A negative covariance indicates an inverse relationship, where one variable tends to increase as the other decreases.

3.4 Extending the BVH Armature with a 3D Skinned Model

BVH skeletons are a very good fit for the baseline body and hand pose estimation methods [6,8,9] due to a 1:1 correspondence between BVH joints and the actual human skeleton and 2D joint estimations produced by e.g. OpenPose [296] as seen in Figure 3.3. Although in our case this holds true for eye gaze vectors which are common between the BVH armature and actual observed eyes, unfortunately all other facial landmarks reside on the skin and thus cannot be directly represented using a purely BVH representation. To remedy this problem we introduce a skinned model, rig it with our BVH armature, make sure that skin vertices are correctly weighted and assigned to skeleton bones, and that facial expressions can be sufficiently represented without rendering artifacts like candy wrapping etc. We employ the Makehuman [23] creator utility which creates 3D skinned avatars that can be controlled using a wide range of controls (gender, race, weight, age) and internally maintains a 1:1 correspondence to a BVH skeleton armature. We also create a plugin [22] that uses the Blender [21] versatile Python SDK and the Makehuman For Blender module [364] that allows seamless pose control and rendering while also being open-source and extendable. We map the parametric space over the facial controls and are thus able to control the neck and head pose, eye gaze, mouth movements, eyelids, eyebrows and nose sniffling. However, having a model and a way to create training facial renderings from BVH parameters requires one more step since our method relies on 2D Joints and not synthetic RGB renderings. To do so, we manually created 2D joint associations between the skinned Makehuman models employed and our OpenPose/IBUG/Multi-Pie facial landmarks. The used facial model consists of 3 separate mesh geometries, one for body and face, one for eyebrows and one for the eyes. Having a 2D IBUG to 3D Makehuman model to BVH parameters triadic correspondence allows us to generate pixel perfect samples to train our neural network ensemble.

Our method is oriented towards in-the-wild, real-time operation using cheap off-the-shelf webcam grade RGB sources. We experimented using two different real-time RGB to 2D facial landmark estimators, OpenPose [296] and Mediapipe [46, 365]. The OpenPose [296] pose estimator produces 68 2D facial keypoints following the Multi-Pie/IBUG [5] configuration, appended by the two iris center locations. Mediapipe BlazeFace [365] produces a tessellated FaceMesh with 468 points and eye-gaze data needs to be extracted separately using MediaPipe Iris [46] output which also tracks the iris size and eyelids. We manually create 2D joint associations between the two models in order to work using the sparse Multi-Pie/IBUG [5] standard, so the two 2D joint sources become interchangeable

For reproducibility we provide the association map between OpenPose/IBUG/Multi-Pie facial landmarks (1..68) to mediapipe's FaceMesh vertices, which is (34, 227, 137, 177, 215, 138, 170, 171, 152, 396, 395, 367, 435, 401, 366, 447, 264, 70, 53, 52, 65, 55, 285, 295, 282, 283, 300, 168, 197, 5, 4, 102, 79, 2, 19, 309, 331, 33, 160, 158, 133, 153, 144, 362, 385, 387, 263, 373, 380, 61, 40, 37, 0, 267, 270, 291, 321, 314, 17, 84, 91, 78, 38, 12, 268, 308, 316, 16, 86).

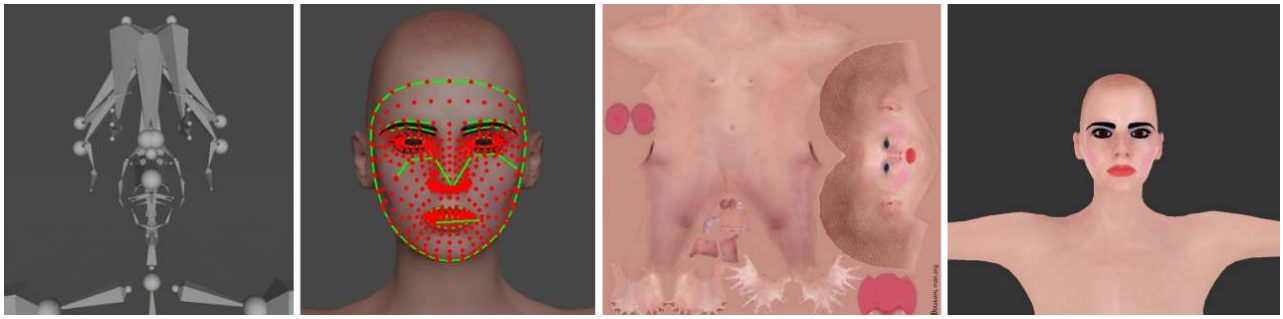


Figure 3.26: From left to right. (1) Our BVH armature is very sparse and its bones cannot be used to directly accommodate facial capture in the same way as the body and hands. (2) 2D face estimators localize keypoints on the surface of the face. We thus need to use a renderer (Blender [21]), and a plugin we developed [22] in conjunction with a (3)(4) makehuman skinned model [23].

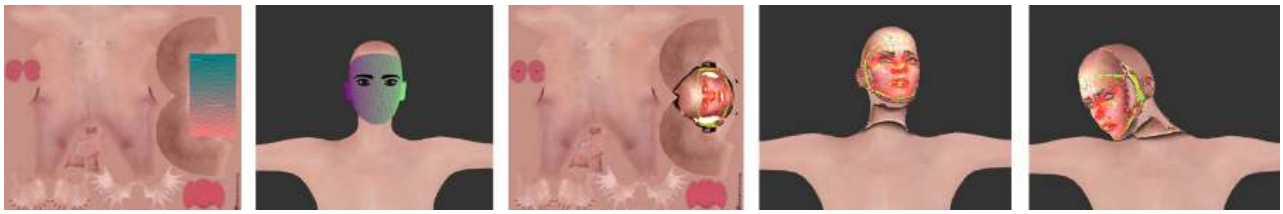


Figure 3.27: Overriding skin texture with a colored pattern, we can facilitate the manual association of 3D model vertices to 2D facial joint estimator locations.

3.5 The MocapNET Neural Network Ensemble

The problem of 3D joint estimation from 2D projections can be traced back to the classic work of Lee et.al. [366]. Their work showcased that assuming a constant known bone length configuration the solution of the problem boils down to a binary decision tree where each split corresponds to two possible states of a joint with respect to its parent. Although works like "3D human pose reconstruction using millions of exemplars" [367] based on a kd-tree [368] there have been many different approaches to tackle the problem. As stated in the literature review due to the incredible success of NNs in the field this is the formulation we chose to use to develop our solution. Major traditional non-NN formulations that successfully tackled the problem before have been Support Vector Machines (SVMs) [369], Boosting [370], Nearest Neighbor based approaches [171, 371] and the Expectation Maximization algorithm [210].

Neural Networks, or to be more precise in the context of our work, Multi-Layer Perceptron (MLP) networks, with one or more hidden layers and a sufficiently large number of neurons are universal function approximators as proven by George Cybenko in 1989 [372] and later independently confirmed by Kurt Hornik [340, 373]. This means that they can approximate any continuous function to an arbitrary level of accuracy given a bounded interval of the function. Our BVH file motions that essentially organize a series of 4×4 matrix transformations followed by a 2D rendering projection seem to meet these criteria. Their continuity is easy to observe since minute changes in rotation angles always lead to small changes in the resulting 2D projections. This is a fundamental characteristic of 3D transforms and one of the properties that give rise to

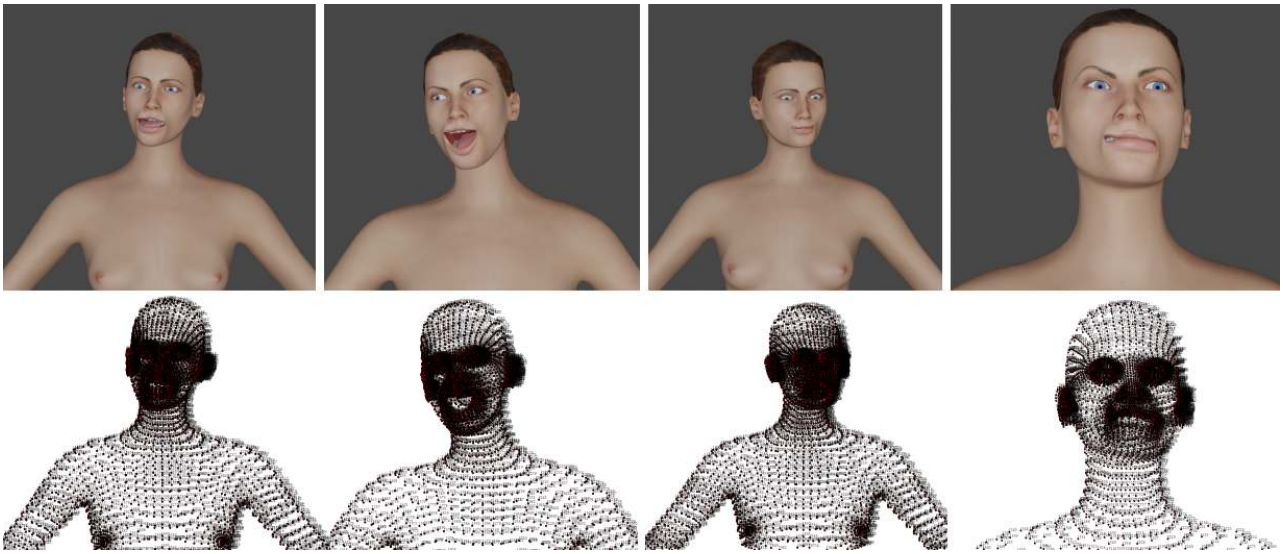


Figure 3.28: Having the Vertex IDs for each facial joint we can generate 3D skinned to 2D training samples for each BVH frame to facilitate our training.

techniques like interpolation that would not stand in a non continuous signal. Given however the fact that multiple 3D BVH configurations can result in the same 2D projections and thus the 3D to 2D mapping is not one-to-one, no loss function can mathematically guide weight updates to a complex enough internal state for this ill-posed problem since the ambiguity caused by the loss of depth information during the projection process inherently cannot be recovered just by 2D points. Keeping these facts in mind, and understanding the architectural limitations of 2D to 3D mapping we set out to derive a NN architecture able to handle the task. As already elaborated in our Literature Review there have been multiple architecture standards proposed with various pros and cons. For our work however, we set out to build a network using first principles and focusing on minimalism and high performance. Our **input** was the 2D joint locations of a body hierarchy coming from an observed RGB image appended by their NSDM, NSRM, eNSRM, PCA encoding, our **output** directly as BVH motion values, and the level of supervision depending from fully supervised in the case of body pose, to unsupervised in the case of facial capture.

Recent works prove that densely connected neural networks like the one we attempt to construct are in fact equivalent to decision trees [26]. Studying Chapters 12.9 and 18 of [25] that offers a Bayesian Perspective to neural networks, offers an insight of how Mixtures of Experts are organized [24] and how the network conditionally selects the appropriate expert “sub-network” according to a gating probability modeled by the activation function and network weights. Although the mixture of experts formulation uses a probabilistic Bayesian learning approach, it can be carried out via loss function optimization without employing probabilistic arguments as seen in [374]. Considering the network as a decision tree also might explain why NSDM/NSRM/eNSRM descriptors that contain both positive and negative correlations of all points (thus performing complementary activations or deactivations of different experts on different parts of the network) are a good fit for this task. After considering all of the above we decided on using eNSRM descriptors which we will proceed to explain in detail.

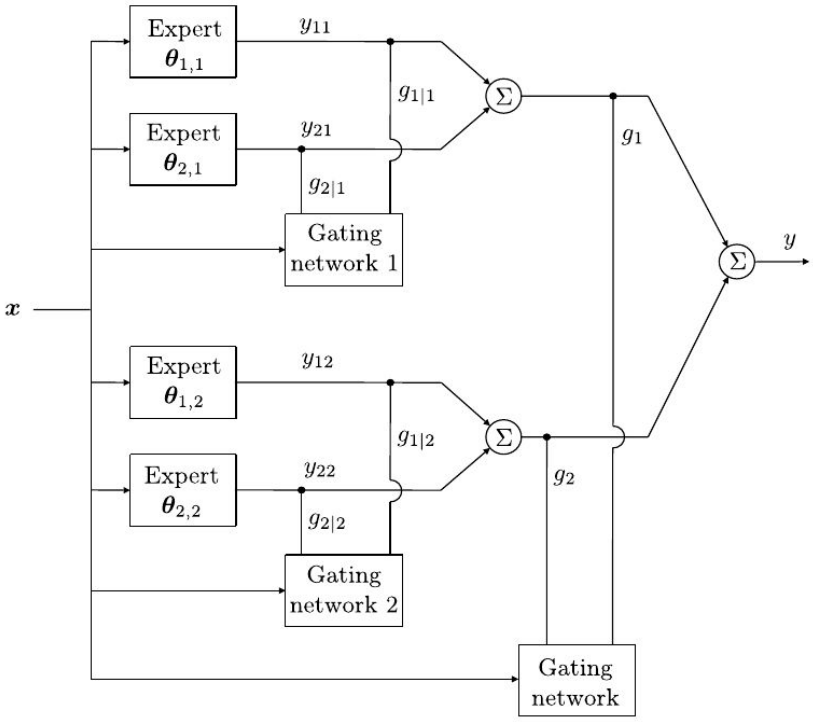


Figure 3.29: An illustration of a mixture of experts [24] network taken from Figure 12.13 of [25]. The equivalence of MLPs to decision trees [26] gives us insight on their internal organization and capacity.

3.5.1 On Multi-Layer-Perceptron Capacity

In a multi-layer perceptron (MLP), the number of such possible activation paths refers to the different combinations of weights and biases that can be activated as data flows through the network from the input layer to the output layer. Assuming **I** input nodes, each input node can either be active or inactive, resulting in 2 possibilities (on/off). So there are 2^I possible activation paths. Assuming there are **L-1** hidden layers (since we're already counting the input layer), and each hidden layer contains a constant number of **N** neurons (to make calculations more practical), there will be N weights connecting each neuron to the previous layer. Each neuron can be activated or not, again resulting in 2 possibilities per neuron. Since there are N neurons per hidden layer, there are 2^N possible activation paths for each hidden layer. Finally assuming that the output layer, similarly to the hidden layers, has **O** neurons in the output layer. Again, there are 2^N possible activation paths for each neuron in the output layer.

Calculate the total number of possible activation paths involves multiplying the possibilities at each layer together:

$$\text{Total MLP activation paths} = (2^N)^L = 2^{LN}$$

Ergo, the maximum represented number of different activation paths in an MLP with I inputs, O outputs,

L layers, and N weights per layer is 2^{LN}. Assuming that due to input correlations each of the outputs evenly requires the same fraction of the activation capacity of the network as the other outputs to be "properly" handled we can define a super-optimistic MLP capacity heuristic, $2^{LN}/O$.

Performing a toy thought experiment where for the 27 dimensional space of a human hand we just want to discern three configurations, e.g. Closed, Neutral, Open (which is very often the case for hands and their fingers) the resulting number of possible configurations is $3^{27} = 7.625$ Trillion distinct configurations. Given a network that receives 63 2D hand points, their 16×16 eNSRM matrix with 256 elements (thus $I = 319$ in total), regresses a single d.o.f. $O = 1$ output in $L = 4$ layers, we would need $3^{27} < \frac{2^{4N}}{1}$ which has integer solutions for $N \geq 11$ weights per layer. $L = 3$, gives a $N \geq 16$, $L = 2$ gives a $N \geq 24$ and $L = 1$ gives a $N \geq 48$.

Assuming that for a more decent approximation of not just 3 states but $\approx 5^\circ$ assuming joint motion ranges between 0° and 90° we would require to distinguish between 18^{27} distinct configurations and having a $L = 5$ set an $N \geq 30$. Unfortunately although these heuristics can serve as a guide to help us think about the capacity of the network, finding actual weights that can perform this task is very difficult. In practice most neural networks start from a random weight distribution that through the algorithm of back-propagation [375] is refined. NN training using backprop however is not guaranteed to converge to a global optimum set of weights. The optimization landscape is highly complex, non-convex and has multiple local minima, saddle points flat regions, valleys and mountains. The various M/L strategies that have been proposed, coupled with the mini-batch stochastic gradient descent [376] however can go a long way towards good weights with careful selection of weight initialization techniques, regularization and dynamically adapting learning rates. Another consideration is that although the possible unique activation patterns are governed by the simple combinatorial logic we presented each neuron/weight is only aware of its input and responsible for its output. This means that information is always bottle-necked by the current layer width, something not at all reflected by the above heuristic. Furthermore the input layer variance and number of inputs is also not taken into consideration. It is obvious that better and more input features will provide more data to the network, thus requiring less neuron capacity to tackle, however the correlation of input values is very hard to quantify and thus integrate into a heuristic. That being said we will revisit the input layer in Section 3.8.

For the task of numerical regression of individual BVH degrees of freedom the output layers of the neural network need to have linear activations, since we want negative values and thus not a non-linear cut-off. Another considerations is that given the $I = 319$ of our example and assuming a constant $N = 30$ for $L = 5$ layers, dropping from 319 to just 30 weights seems to cause a huge information bottleneck [377]. At the same time although going from 30 weights to 27 outputs in our output layer this also causes problems due to the additive nature of the loss function as we will see, leading us to regress 1 output value per encoder. Dropping from 319 elements to 1 thus in the smoothest possible way can be done by gradual layer width reduction in an attempt to reduce abrupt information bottlenecks as much as possible.

3.5.2 Loss Functions

There have been many proposed loss functions for different tasks, however regression losses in the literature are either MAE or Mean Squared Error (MSE).

with:

$$\text{MAE} = |y_i - \hat{y}_i| \text{ and MSE} = (y_i - \hat{y}_i)^2$$

where y_i is the ground truth value for the i-th data point and $\hat{y}_i$ is the prediction for the i-th data point.

Listing 4 Custom loss functions implemented in the Tensorflow [378] Machine Learning Application Programming Interface (API) while experimenting with our network formulation. MSE loss outperformed all of them.

```
def mean_quad_error(yTrue,yPred):
  return tf.reduce_mean(
         input_tensor=tf.math.square(
                      tf.math.square(
                       tf.subtract(yTrue,yPred) ) ) )

def mean_squared_error_modulo_360(yTrue,yPred):
  return tf.reduce_mean(
         input_tensor=tf.math.square(
                      tf.abs(
                       tf.subtract(
                        tf.math.floormod(tf.add(yTrue,180),360),
                        tf.math.floormod(tf.add(yPred,180),360) ) ) ) )

def average_error_modulo_360(yTrue,yPred):
  return tf.reduce_mean(
         input_tensor=tf.abs(
                      tf.subtract(
                       tf.math.floormod(tf.add(yTrue,180),360),
                       tf.math.floormod(tf.add(yPred,180),360) ) ) )
```

Attempting to use both we ended up using MSE since its exponent suppresses outliers in predictions without performing intense weight updates for values that are closer to the ground truth. We also experimented with a loss function called Mean Quad Error (MQE) = $(y_i - \hat{y}_i)^4$, however, we found it to perform worse than MSE possibly due to overly large loss values and too much focus on reducing large errors that derails the optimization process. Another custom loss function we implemented was a "modulo 360"

3.5.3 MLPs and Back Propagation

Our NN follows the standard MLP formulation. As seen in Figure 3.30 each neuron is equivalent to a function $\sigma(x)$ where:

$$z = \sigma(w_1 x_1 + w_2 x_2 + \ldots + w_k x_k + b)$$

Assuming a set of weights and biases and the ground truth 3D BVH to 2D pairs, training happens through an iterative process of forward and backward value propagation governed by an optimizer like SGD, ADAM or RMSPROP [379]. Forward propagation is the process of providing a set of input values (our 2D+eNSRM data) and progressively executing every function/neuron until the output layer. Although there have been network formulations like Long short-term memory (LSTM) Recurrent Neural Network (RNN) that use Back Propagation Through Time (BPTT), Back propagation performs weight updates based on the error recorded during forward propagation.

Figure 3.31 shows a disected region of our MLP network where a_k^l is the k-th activation for layer l:

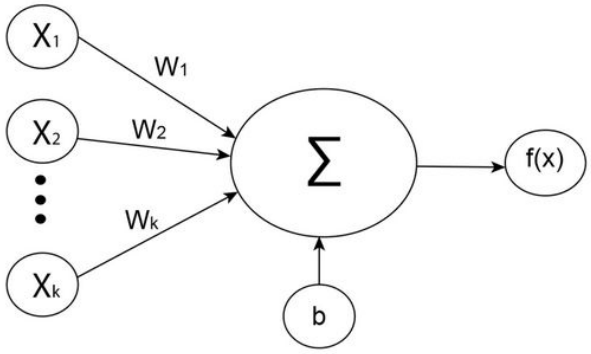


Figure 3.30: An artificial neuron is a mathematical formulation that mimics a simplified biological neuron. The neuron computes a weighted sum of its inputs adds a bias and gives output based on an activation function.

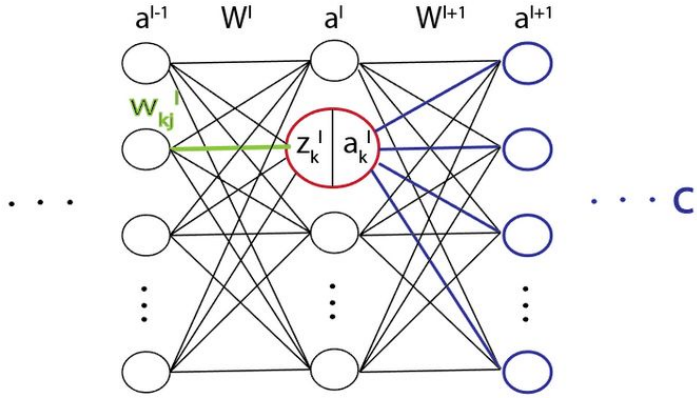


Figure 3.31: Modern GPUs are optimized for vector transformations. Performing hardware accelerated forward and backward propagation passes, involves assembling values in matrices and then performing the calculations for each neuron (Figure 3.30). The illustration pictures each sequential layer l with activation results organized as column vectors a and weights W stored as matrices.

$$a_k^l = \sigma(z_k^l)$$
$$z_k^l = w_k^l 1 a_1^{l-1} + w_k^l 2 a_2^{l-1} + \ldots + w_k^l n a_n^{l-1} + b_k^l$$

The derivative of the cost C with a respect for a particular weight is:

$$\frac{\partial C}{\partial w_{kj}^l}$$

Applying the chain rule and extending the derivative we have:

$$\frac{\partial C}{\partial w_{kj}^{l}} = \frac{\partial C}{\partial z_{k}^{l}} \frac{\partial z_{k}^{l}}{\partial w_{kj}^{l}} = \frac{\partial C}{\partial a_{k}^{l}} \frac{\partial a_{k}^{l}}{\partial z_{k}^{l}} \frac{\partial z_{k}^{l}}{\partial w_{kj}^{l}} =$$
$$= (\sum_{m} \frac{\partial C}{\partial z_{m}^{l+1}} \frac{\partial z_{m}^{l+1}}{\partial a_{k}^{l}}) \frac{\partial a_{k}^{l}}{\partial z_{k}^{l}} \frac{\partial z_{k}^{l}}{\partial w_{kj}^{l}} =$$
$$= (\sum_{m} \frac{\partial C}{\partial z_{m}^{l+1}} w_{mk}^{l+1}) \sigma^{'}(z_{k}^{l}) a_{j}^{l-1}$$

The error δ of a neuron k in layer l is:

$$\delta_{k}^{l} = \frac{\partial C}{\partial z_{k}^{l}}$$

Performing substitution on the previous chain derivation and assuming that the last layer is L we can calculate the error of neuron k in the last layer L.:

$$\delta_{k}^{l} = (\sum_{m} \delta_{m}^{l+1} w_{mk}^{l+1}) \sigma^{'}(z_{k}^{l})$$
$$\delta_{k}^{L} = \frac{\partial C}{\partial z_{k}^{L}} = \frac{\partial C}{\partial a_{k}^{L}} \frac{\partial a_{k}^{L}}{\partial z_{k}^{L}} = \frac{\partial C}{\partial a_{k}^{L}} \sigma^{'}(z_{k}^{L})$$

Substituting on the derivative of the cost C with respect for a weight started from on the previous chain derivation and assuming that the last layer is L we can calculate the error of neuron k in the last layer L.:

$$\frac{\partial C}{\partial w_{kj}^{l}} = \delta_{k}^{l} a_{k}^{l}$$

which in terms of the weight update assuming a learning rate λ:

$$w_{kj}^{'l} = w_{kj}^{l} - \lambda \delta_{k}^{l} a_{j}^{l-1}$$

Bias updates follow a similar logic, however since biases are constants we have:

$$x^{'} = \frac{\partial x}{\partial x} = 1$$

Assuming that $\lambda \neq x$

$$\frac{\partial (x+\lambda)}{\partial x} = 1 + 0 = 1$$

Given that layer l with the k-th bias is governed by:

$$\frac{\partial C}{\partial b_{k}^{l}} = \frac{\partial C}{\partial z_{k}^{l}} \frac{\partial z_{k}^{l}}{\partial b_{k}^{l}} = \frac{\partial C}{\partial z_{k}^{l}} \times 1 = \delta_{k}^{l}$$

$$b_{k}^{l} = b_{k}^{l} - \lambda \delta_{k}^{l}$$

As loss function magnitudes fluctuate the individual updates following the neuron chains are influenced in a relative manner. Activation functions play an important role and their properties define how a neural network behaves as more layers get added and different magnitude values are propagated. These problems can be exacerbated when random weight values are not properly initialized. One of these types of problem is called vanishing gradient and refers to the gradual magnitude decrease during back propagation that can become too small to meaningfully change weight values at the first layers of the NN thus prohibiting convergence to an optimal solution. The contrary (diametrically opposed) problem is when very large loss values and/or Learning Rate (LR) cause very large weight updates that cause weights to violently oscillate and the forward/back propagation procedure to stop converging.

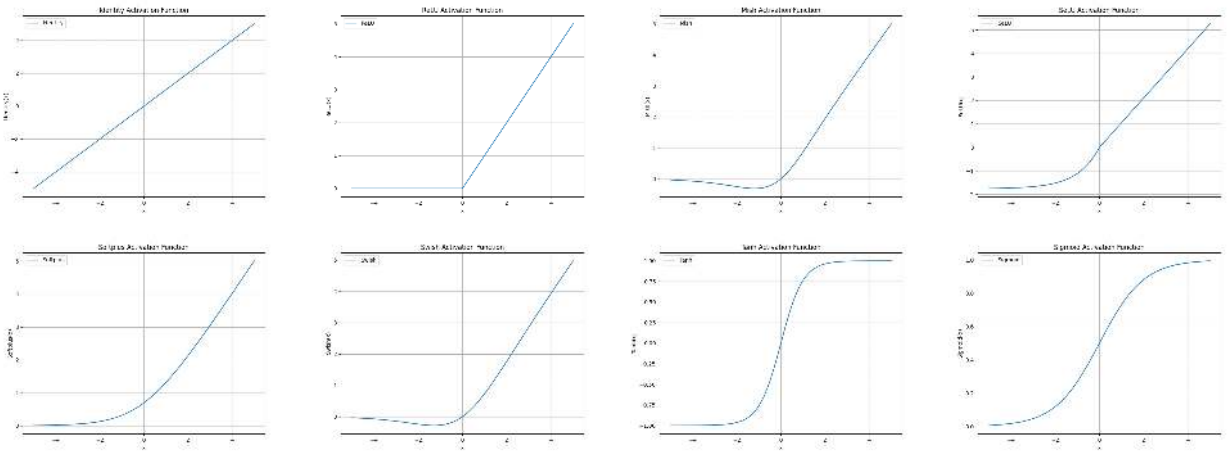


Figure 3.32: Activation function plots. Top to bottom and left to right: Identity, ReLU, MISH [27], SeLU [28], softplus, SWISH [29], tanh, sigmoid.

The main reason for vanishing gradients is that the derivative of most activation functions is much smaller than the input function itself and this leads to smaller and smaller gradient values that do not allow the network training to progress past a limited number of layers. In a similar fashion large magnitude weight updates tend to exhibit a similar cascading effect leading to a chaotic NN behavior that diverges until Not a Number (NaN) values are encountered making the continuation of the training process numerically impossible. Weight initialization techniques to combat this problem were proposed in 2011 by Glorot et.al. and the team of Yoshua Bengio [380]. The use of activation functions like ReLU that do not have value ceilings further improved the situation (although creating other side effects that where tackled using Leaky ReLU and other ReLU iterations), while Batch Normalization [381] calibrated the inputs using two additional training parameters in an effort to mitigate problems from disproportionate flows of very large or small values. A more detailed look into activation functions and their pros and cons is offered by [382], while plots of popular activation functions can be seen in Figure 3.32. Once again we observe that activation functions perform gating similar to Figure 3.29 essentially controlling the data flow in two separate ways based on the weight values of each neuron.

It is worth noting that there is a very large volume of research on similar topics with the state of the art constantly shifting towards more robust training that leads to better and higher accuracy networks. That being said due to the computational complexity of training using very high definition inputs with millions of samples a lot of methods focus on CNN networks that feature convolution operators. An especially interesting formulation similar to the Capsules [166] we have already briefly discussed in Section 3.2 is the concept of equivariant neurons like the ones proposed in [383]. Recent works [384] achieve equivariance for arbitrary Lie Groups (see Table 3.1) Other formulations use perform input tile rotations to achieve equivariance like the Polar Transformer Networks (PTN) [385] which extends Spatial Transformer Networks (STN) [386] and with methods like [387] that introduce Equivariant Transformer Layers that align input using canonical coordinates. It should be stressed that transformer networks continuing the line of research started by STNs [386] should not be confused with Transformer networks associated with natural language processing and document recognition. The more powerful Vision Transformer (ViT) [132] architecture of Natural Lan-

guage Processing (NLP) transformers [388] is also the topic of ongoing research in equivariance with works like PatchRot [389] that attempt to combine the concept of ViTs in a fashion similar to Convolutional NNs like Rotnet [390]. While currently the general belief is that a complex architecture like Vision Transformers is needed to scale to datasets of Bullions of examples, this belief is challenged by [391] and [392] that match the reported performance of Vision Transformers with comparable compute budgets using regular ConvNets.

3.5.4 Ensemble of Encoders vs. a Monolithic Network

The final aspect of our network was that given the orientation classifier we introduced and that will be elaborated on the next Section using a categorical cross-entropy loss, regressing 20 discrete high-level BVH configurations was not experimentally feasible when we initially attempted it. We attribute this to loss function "cross-talk" between different non co-varying outputs. Although CNNs manage to output large (e.g. 64x64 pixel) heatmaps using Binary or Categorical cross-entropy loss returning classification scores it was troubling how and why CNNs with MAE managed to perform this task. However CNNs typically use different channels per output, heatmap pixels typically co-vary with their neighbors, while the existence of a high value (detected feature) somewhere typically means that this feature will not be found elsewhere thus being akin to a long number of zeros with just a few values being active and contributing to the loss function for each sample. Last but not least the activations of heatmaps have the same range of values (0..1) while different degrees of freedom of our problem have much different magnitudes that when summed together lead to some aspects of the problem being more penalized than others. For example in the case of a hand described by translational 2 X,Y degrees of freedom in centimeters ranging from [-500..500], 1 d.o.f describing depth in the range [500..4500] , 4 d.o.f of quaternions [-1..1] and the rest 20 with different ranges of euler angles it is evident why all these different elements cannot be directly regressed together with a simple summed loss. The later implementations [20, 31] of the method performed thorough output normalization making joint training possible, albeit resulting in a network with less accuracy compared to the individual loss paradigm.

For a MSE^{All}_{Loss} loss that covers all N elements of an output vector, for a batch of b samples of N outputs, where i=1..N the MSE Loss is defined as

$$MSE^{All}_{Loss} = \sum_{sample=1}^{b} \sum_{i=1}^{N} (y_{i,b} - \hat{y_{i,b}})^2$$

Assuming a MSE^{Indv}_{Loss} loss for individual elements of the matrix we have of b samples of N outputs, where i=1..N the MSE Loss is defined as

$$MSE^{Indv}_{Loss} = \sum_{i=1}^{b} (y_i - \hat{y_i})^2$$

To showcase the difference between MSE^{All}_{Loss} and MSE^{Indv}_{Loss}, we generate a small simulator, with the implementation in Listing 5. Assuming a vector with N elements where N=20 (e.g. the articulation of the hand without its root), a batch size of 32 samples, random output values bounded in the range [-1..1] and F=5 output elements randomly flipped simulating errors in regression in an other optimal NN. After repeating the simulator for 10000 iterations and recording losses for the monolithic case where all outputs share the same loss MSE^{All}_{Loss} and the individual encoder case where each output has its own loss MSE^{Indv}_{Loss} we observe Figure 3.33.

The Gaussian distribution of Figure 3.33 seems consistent with the central limit theorem [393] being the sum of essentially random contributions of the loss function, however what is immediately obvious is that

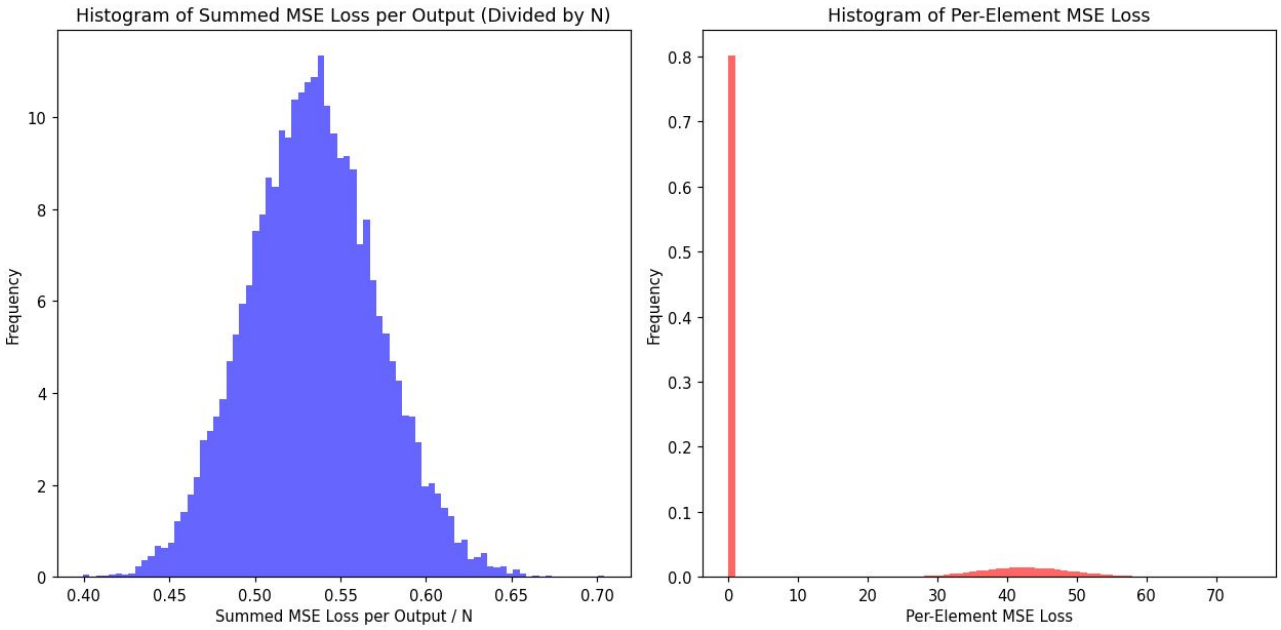


Figure 3.33: Histogram loss values of Summed MSE for a network with N outputs, versus a histogram of loss values for a network where each output has its own loss produces using the code in Listing 5. We observe a vastly different behavior, prompting us to use an encoder architecture with per sample loss function, to achieve fine-grained fitting.

using a separate loss per output has a different behavior on our simulated example, with the very large number of unflipped values being naturally filtered out and losses having larger magnitudes only on the specific encoders/output elements needed. Experimentally we observed similar behavior when training with accuracy degrading as we incorporated more output degrees of freedom using the same network. Shared MSE loss across all meticulously normalized output elements with a similarly sized network performs $\approx$ 20mm worse than the individual encoder example scenario. Despite this however the separation of the regression task to independent encoders also allowed superior control over the network parameter count sizes, while having just one output value instead of e.g. 20 allowed us to accommodate much smaller networks with more training samples in GPU memory. An additional benefit was being able to handle different training sessions clustered over more relatively inexpensive workstations. All those reasons prompted us to split each output d.o.f. to a separate "encoder" network with its own loss and isolated training procedures and weights.

Although as we observed in Figure 3.21, real Motion Capture training data follow much different distributions that the uniform random simulated distribution of Listing 5, we believe that this code stands as a valid thought experiment showcasing how d.o.f. separation with different encoders naturally filters out poses (with $\approx$ 0 losses that cause $\approx$ 0 weight updates) that would otherwise need to be concurrently handled possibly by a much more complex and large neural network. The conditional independence of encoders is also another aspect of this architecture since each joint is optimized using a loss function that ignores

Listing 5 Code simulation to produce Figure 3.33

```
def compute_summed_loss(y_true, y_pred, batchID):
  return np.mean(np.square(y_true[batchID] - y_pred[batchID]))

def compute_per_element_loss(y_true, y_pred, batchID, outputID):
  return np.square(y_true[batchID, outputID] - y_pred[batchID, outputID])

def simulate_experiment(b=32, N=20, flipNum=5, num_simulations=10000):
  summed_losses = []
  per_element_losses = []

  for _ in range(num_simulations):
     # Generate random ground truth values for each batch
     y_true = np.random.uniform(-1, 1, size=(b, N))

     # Generate random predictions for each batch based on flipped values
     y_pred = y_true.copy()
     flip_indices = np.random.choice(N, flipNum, replace=False)
     y_pred[np.arange(b)[:, None], flip_indices] *= -1

     for outID in range(N):
       loss = 0
       for batchID in range(b):
         loss = loss + compute_per_element_loss(y_true,y_pred,batchID,outID)
       per_element_losses.append(loss)

     loss = 0
     for batchID in range(b):
       loss = loss + compute_summed_loss(y_true,y_pred,batchID)
     summed_losses.append(loss / N)

  return summed_losses, per_element_losses
```

the covariances of all other outputs, while being able to have a relatively short but very wide architecture presents itself to multi-core/multi-thread processing using just a CPU at evaluation/runtime. As we will see in consequent Sections during training we perform a form of transfer learning that circumvents this total separation, while the individual independence property is another property we leverage in our fine tuning module. At the time of writing this thesis we are able to train a larger (larger λ value network) using a shared loss with similar and for some configurations slightly better accuracy than the individual trained encoders paradigm, but for the rest of this text we will consider the independent encoder scenario.

3.5.5 Model Improvements Through Time

As stated before our effort was to start from first principles and given the NSDM, NSRM, eNSRM descriptors that have some invariance attributes perform the best possible results without an overly complex architecture to ensure real-time operation.

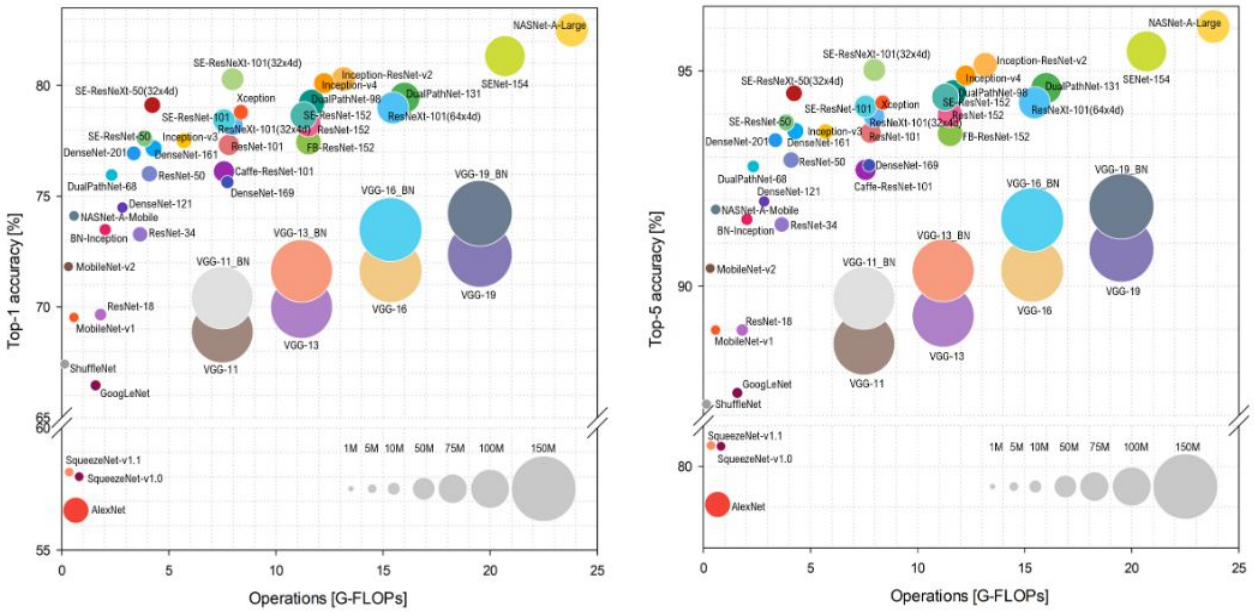


Figure 3.34: A graph of top-1 accuracy and top-5 accuracy of different NN architectures from Simone Bianco et.al. influencial work "Benchmark analysis of representative deep neural network architectures" [30]. An ideal method would be positioned on the top left of these graphs, influenced by this study, our method combines aspects of DenseNets with residual connections.

After briefly mentioning design characteristics of NNs to give insight to the reader about our thought and design process, we can return back to the topic of our MocapNET Ensemble. Trying to create a minimal Network with good properties the search for a standardized approach towards similar dimensionality problems brought to our attention the "Benchmark analysis of representative deep neural network architectures" [30] published in the same year as the time at which we where conducting our initial literature review. Figure 3.34 showcased clearly that Dense-Nets [394] and Res-Nets [395] combined the traits of high-accuracy, low computational complexity (high performance) and low parameter count (e.g. practical in terms of training). Using this as a guide we iteratively refined our network that began as a simple MLP [6] and gradually grew deeper and was enriched with residual connections [8, 9, 31]. Given that our first implementation temporally coincided with the advent of Self-Normalizing Networks (SNNs) [28] after a brief period of experimentation using multiple different activation functions we experimentally chose the SeLU (Figure 3.32) activation functions, for the method's first implementation [6] with the shape shown in Figure 3.35.

The trade-off of performance vs accuracy of the simple original network was admittedly overly skewed towards performance. The original network achieved super-real-time performance of over 400Hz in CPU execution while achieving an abysmal accuracy of ±160 mm in the H36M [38] dataset compared to the state of the art when measured against the Human3.6M dataset [38] as we will later see in the experiments Section. Not satisfied with the very low accuracy of the simple original method and keeping in mind the literature discussed we proceeded to deepen the network as seen in Figure 3.36, something that we only managed

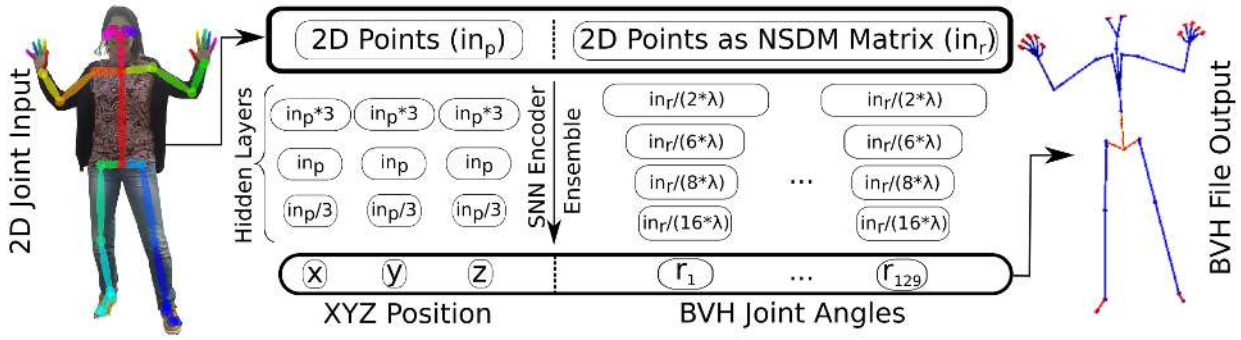


Figure 3.35: The debut of the MocapNET ensemble of SNN [28] encoders in BMVC 2019 [6]. λ was a scaling value hyperparameter allowing control of the capacity of the network in a uniform well documented manner.

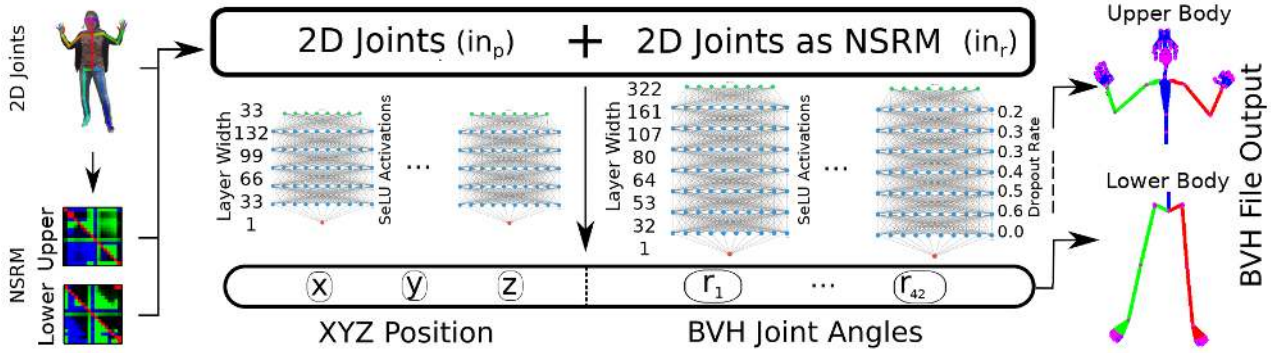


Figure 3.36: The second iteration (MocapNET 2) ensemble of SNN [28] encoders in ICPR 2020 [8], dividing the body in two regions while also deepening the network.

to do in the context of an MLP using an aggressive dropout [396] schedule across the whole network, and further dividing the problem in four orientations instead of just two as we will see in the next Section.

General NN improvement strategies include deepening and increasing layer parameter counts for improved network capacity. Attempting this in a densely connected network like [6,8] however was not straightforward. Dense network parameter counts increase exponentially with added hidden layers and their ensembles quickly become bulky, difficult to train and slow to execute. At the same time, an other adverse effect of "full layer connectivity" was over-fitting. Even applying strong regularization via dropout [396] at a high rate of 30% we got diminishing returns after 5 dense layers. Techniques to deepen networks typically improve information flow like Highway Networks [397]. Works like Deep Residual Learning [395] create residual learning blocks that achieve this effect via alternative data paths or similarly [394] consists of dense blocks punctuated by convolutional and pooling layers. Keeping these works in mind and once again influenced by the study of Bianco et al. [30] that highlights the high accuracy and low resource consumption on ResNets and DenseNets, we attempted to combine their designs characteristics with our MLP. After numerous experiments with variations of the original architecture [6] we successfully extended MocapNET

Figure 3.37: The third version (MocapNET 3) ensemble published in BMVC 2021 [9], also handling hands and featuring residual connections.

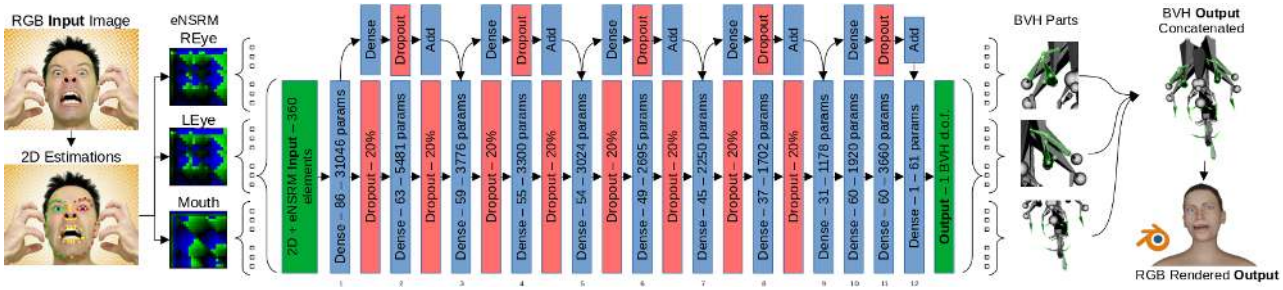


Figure 3.38: The final version (MocapNET 4) ensemble published in Analysis and Modeling of Faces and Gestures Workshop of ICCV 2023 [31].

encoders by stacking dense layers that featured residual connections to remedy vanishing gradients. We retained the beneficial high dropout [398] to deal with over-fitting and partly corrupted/missing input data in cases of occlusions. Figure 3.37 illustrates our proposed architecture, a hybrid of the above mentioned design elements.

Our proposed architecture also employed the SWISH [29] activation function. This consistently improved training accuracy compared to the SeLU [28] function of [6, 8] by 0.1° up to 1° in each recovered d.o.f. During the course of development of our method we also experimented with MISH [27], however despite its superior accuracy in benchmarks like Cifar-10 [399] we found it did not perform better in our problem.

After iterative changes and many experiments the final Figure 3.38 version of the network was published in ICCV [31] shortly before completing this thesis.

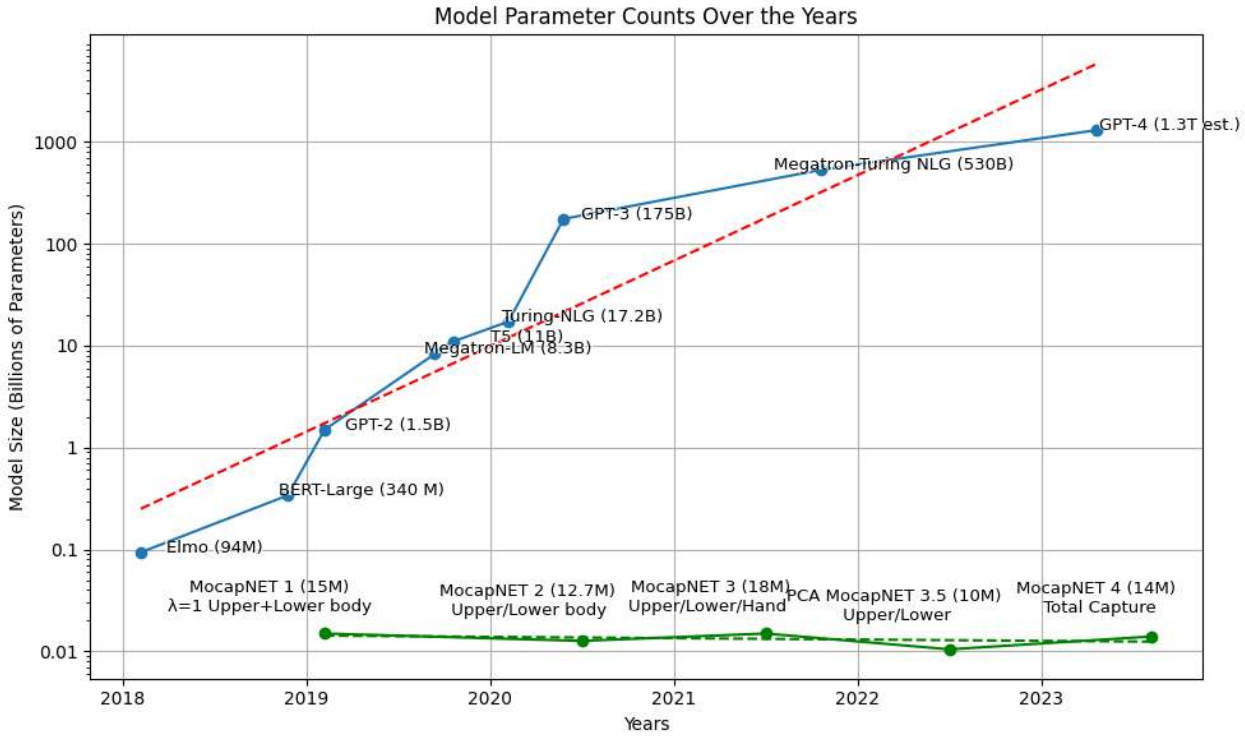


Figure 3.39: Comparative sizes (in Billions of parameters) of state of the art NNs [32] versus the size of progressive MocapNET versions [6, 8, 9, 20, 31] from the start of this thesis to this dissertation. Large language models exhibit an increasing trend and requiring huge amounts of computing resources and power to be trained and executed. Our effort with "MocapNETs" on the other hand was always to provide the best possible model accuracy/complexity ratio and ensure real-time operation on low-spec commodity hardware. Although the size of MocapNETs increased due to more targeted hierarchies (hands, face, gaze) that required more encoders and thus more ensembles and parameters, our methodological improvements and divide and conquer strategy managed to keep model complexity in check, meaning that the cumulative model size of our ensembles has stayed roughly the same across time.

3.5.6 Dividing the Task using an Orientation Classifier

Divide and conquer strategies are essential problem-solving techniques employed to simplify high-dimensional problems that are difficult to analyze or solve. Our problem in its core is highly combinatorial since the formulation of a hierarchical skeleton follows a natural and profound expansion of possible states by increasing the scope and detail of the hierarchical kinematic chain. As more degrees of freedom are integrated to the armature, pose space grows exponentially making it hard for any method to effectively tackle the problem in a resource constrained manner.

Although we will deal with dimensionality reduction techniques in Section 3.8, recounting the starting steps of our method in the introduction of Chapter 3, one of the first successful neural networks we imple-

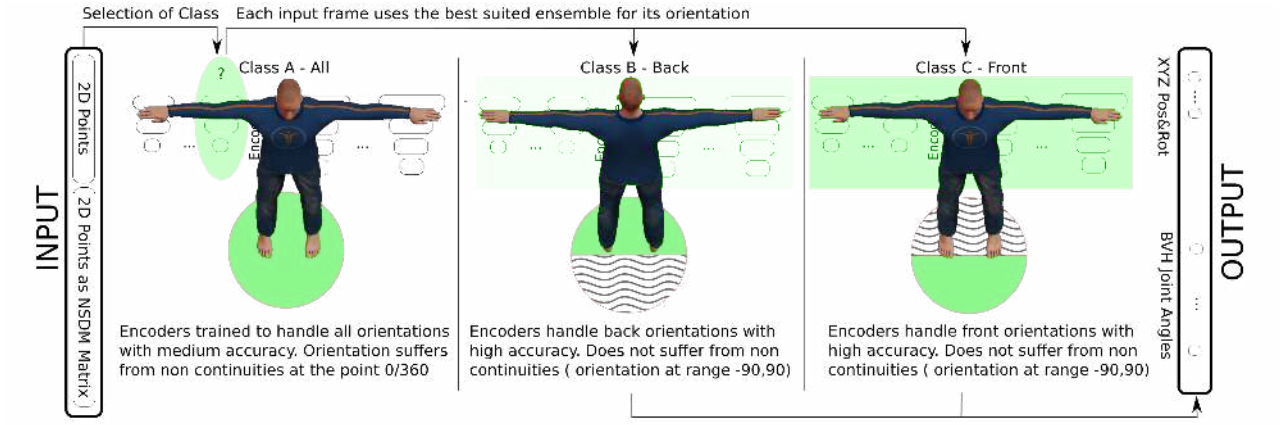


Figure 3.40: The original MocapNET [6] split pose space to two orientation classes, forward and backward. Upon receiving 2D input a classifier decided on the orientation of the subject and performed hand-over of the regression task to the expert ensemble trained for the particular orientation.

mented dealt with the problem of discerning a front or back oriented skeletons from 2D joints. Although both forward looking and backward looking skeletons have the same number of independent dimensions, and thus their combinatorial complexity in terms of the number of total possible poses handled is the same, they can help us simplify the problem. Even without reducing the degrees of freedom but rather their effective range we can expect superior accuracy due to a more constrained task. Going back to the previous Section thought experiments with regards to "Total MLP activation paths" we would expect that assuming fixed network capacity to represent a constant number of non overlapping states on a range of let's say 360° reducing this to 180° (e.g. only frontal views) would double the resolution of the method in this particular subset of the data. Comparing Figures 3.9 and 3.16, we can observe that horizontally flipped skeletons (skeletons facing the opposite direction) produce 2D descriptors with inverted magnitude values, producing the largest summed difference between poses. Having stated the original reasoning between the subdivision of pose space to individual orientation classes this approach led to a series of methodological advancements while at the same time some important drawbacks.

One of the first problems identified and triaged with this class seperation was the BVH angle discontinuity problem [164]. Due to the multiplicity of angles, (since e.g. $-360°$, $360°$ are vastly different outputs in terms of magnitude, however they resolve to the same angle and 2D projections) we initially experimented with custom loss functions 5 reflecting their periodicity. Class separation however deals with this problem without any effort since each orientation class output is constrained $-180° \leq orientation \leq 180°$ thus implicitly resolving this issue. Although the orientation classifier needs to handle all of the orientations at once, its output classification does need to involve the actual angle and thus training can be effective and seamless.

Another helpful feature of orientation separation is that given the initial training constraints (the project began on a workstation featuring an NVIDIA GTX 970 GPU with 3.5GB VRAM and 16GB System RAM), loaded training samples can also be more representative of the pose space. Assuming that the CMUMocap [18] datasets with 3.9M would fit to GPU VRAM given loading each independent output degree of freedom isolated from the others, having 2 orientations meant that the randomization scheme we employed

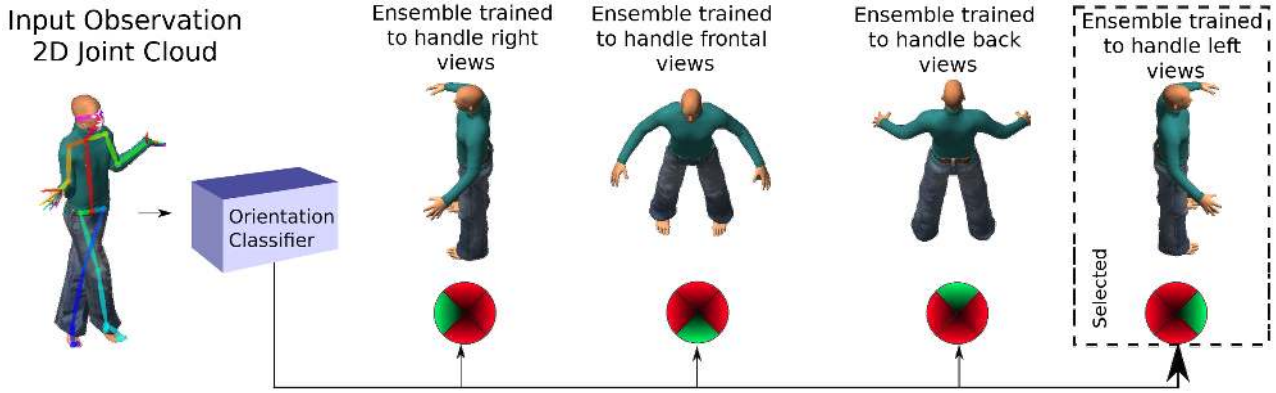


Figure 3.41: Attempting to improve regression accuracy, the second iteration of our method [8] doubled pose space divisions compared to Figure 3.40 splitting orientation classes four ways while at the same time splitting the body in two ensembles one for upper and one for the lower body.

(Figure 3.21) would produce a much denser set of samples in the specified range. This would mean that although having just 3.9M samples at any time loaded on our constrained hardware resources during training the effective combination of front/back facing ensembles would be trained on an aggregate 7.8M samples without needing to accommodate all of them at any time.

There where two main problems with this strategy. First the orientation classifier introduced a central point of failure to the otherwise "decentralized" regression from individually trained encoders. In case of an incorrect classification, although the ensembles might had the capacity to correctly regress the pose, the classification error handed the pose to a network not trained to do the job, leading to bad output with highly negative contribution to the recorded accuracy. The second important issue was that splitting the problem in two orientations essentially required double the training time in order to train each of the orientations separately.

Attempting to improve regression accuracy in MocapNET2 [8] led to further splitting of orientation classes to four. Although in the context of the original BMVC 2019 [6] publication 2x training time for both orientations was a non-issue, this quickly became a problem during the second iteration of the method. The splitting of the armature to upper and lower body each of which needed to be independently trained for 4 different orientations resulted in an 8x training time, with training sessions that would require more than a week on a single computer. In order to progress hyper-parameter optimization along with the training in a reasonable time-frame we resorted to using multiple workstations as a cluster and performing training for different orientations/hierarchies in different PCs and then gathering the results in the central machine, which also performed training but doubled as the coordinator of the process. Figure 3.41 shows a block diagram of the execution flow using four orientation classes, that was the maximum attempted degree of orientation seperation attempted since more orientations would make training unfeasible in the time-frame of this PhD.

The major early architectural design decision in MocapNETs [6, 8] to partition pose space in bounded orientation classes reached its logical conclusion during the work for MocapNET3 [9] and while attempting to apply this logic to hand orientations. Initial efforts to treat the problem in a divide-and-conquer fashion

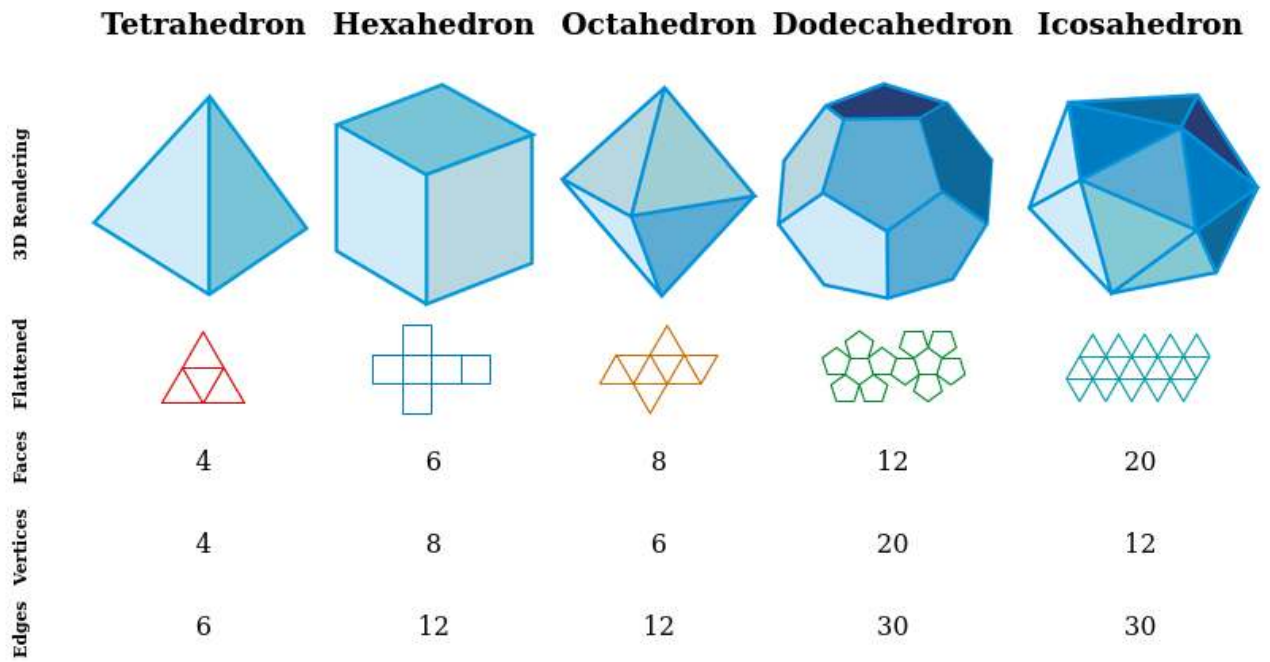


Figure 3.42: Platonic solids we used attempting to partition 3D orientations in 4, 6, 8, 12 and 20 by placing the camera on the center of each face.

in 2 [6] or 4 [8] quadrants did not produce the same results since hands in contrast to the body that is typically upright can have arbitrary rotations in all axis, and thus splits in terms of orientations do not have the same effect. Attempts to perform this logic in all three root rotation degrees of freedom led us to Platonic solids (Figure 3.42) and partitioning orientation space in 3D using an icosahedron solid with the goal of training an ensemble for each of its twenty faces. Despite the best of our efforts this approach was not viable since it apparently overly burdened the classifier creating a central point of failure for the pose estimation of each hand. The best orientation classifier we were able to train achieved a mediocre 82% classification accuracy on our randomized training data. Furthermore, training times and runtime memory demands increased twenty-fold. Thus, we opted to forego orientation classification and instead, increase the capacity of the ensemble.

3.6 Training MocapNETs

Neural network training is the bottleneck in NN development since it poses a long and arduous energy and resource intensive process. As seen in Figure 3.39 Neural Network sizes are exploding in recent years. A telling example that brings training into perspective is that **just one** training session of a generative NN such as stable-diffusion [56] against e.g. the LAION-5B dataset [400] requires 256 NVIDIA A100 GPUs working for 150.000 GPU hours which done through Amazon Web Services amounts to a cost of $600.000. This cost is in stark contrast with our presented MocapNET method that can be trained by a single commodity workstation with 16GB RAM and a very low end NVIDIA GTX 970 GPGPU with 4GB VRAM, which was 8

years old at the time of writing this thesis, in close to 2 weeks or $\approx$ 672 hours. Although actions like dividing pose space into multiple orientations or increasing sample sizes can lead to more extensive resource consumption the overall training scope of the project was a very important consideration for deciding the scope and direction of this thesis. For example directly training the RGB to 2D joint neural network or even worse merging it in a monolithic fashion with the MocapNET ensemble would require substantial computing resources that where not available at the start of the thesis. The sparseness of the 2D/BVH data we opted for using was a direct choice motivated by the lack of computing resources and since it made large training samples approachable with regular off-the-shelf (already 4 years old at the start of the PhD) GPGPUs. The independent encoder ensemble architecture meant that multiple workstations could proceed training in parallel thus providing a linear speedup to the training process, while the divide and conquer approach in terms of orientations (Section 3.5.6) combined with the independent descriptor decomposition in separate hierarchies(Figure 3.7) meant that each of these could be tackled independently with minimal compute power. Finally leveraging properties of the problem such as horizontal symmetries we were able to completely forego the need for training by utilizing the same ensembles and performing 2D/3D transformation to respectively flip the inputs and outputs of the ensembles. During the second year of development of the thesis we received a generously donated NVIDIA Quadro P6000 featuring 24GB of onboard VRAM that greatly enhanced the training effort while towards the latter part of development the procurement of a system featuring dual A6000 GPUS with 98GB VRAM resolved all constraints allowing training sessions with ¿8M samples.

Although training is an expensive effort it is an essential reality of using NNs. It might take years to find a NN architecture with an appropriate experimental configuration and a unique set of weights that performs in an efficient way. Despite this heavy initial cost upon developing and training a NN it can be quickly shared across millions of devices. A NN that can accurately compute a highly non convex function and give solutions to a difficult (sometimes otherwise intractable) problem and be evaluated in milliseconds during its deployment makes training worthwhile. What is more is that improved versions of NNs can be easily and effectively communicated by the scientific community saving the trouble and cost of "reinventing the wheel". Having the aspiration of properly communicating our training protocols and results we dedicated the required resources in the hope that our NN benefits vastly outweigh its cost. Our training effort was punctuated by the intermediate publications of the method [6, 8, 9, 20, 31]. Building the method from scratch meant that we did not rely on experiments done by others and had a very big surface of experiments to cover. Some important experiment parameters such as software versions, edge case handling, pipelining and internal configurations of the training code are very hard to document. We began our training effort using Tensorflow 1.5 with CUDA 9.0, and at the time of writing use Tensorflow 2.13.0 and CUDA 11.8. Given that the Tensorflow Github repository has one of the most active development efforts globally it is no wonder that subsequent versions behave slightly differently even when used with the same configuration and dataset from our side. Having said that however our training was robust and reproducible throughout all of these software versions.

Early training attempts during the time of developing the method from scratch did not have a clearly defined training protocol since what in hindsight is now known to be possible by the method, at that point it time was unknown. Due to the incredible complexity and number of options explored in the context of this thesis, after our first publication [6] it became essential to keep track of the various subsequent ideas explored with experiments. This was done with a series of measures. Every experiment received a unique serial number, a lab-book was kept with a summary of experiment purpose, notes and results. The first entry of this logging system was done in 24-2-2020, prepended by the MocapNET legacy v1 [6] configuration.

Hyper-Parameter	Tested Range	Value	Description
precision	fp16/fp32/int8	fp32	Floating Point Accuracy used
outputMode	BVH/3D	BVH	What does the network resolve
veryHighNumberOfEpochs	32..512	512	Maximum number of Epochs (difficult joint)
highNumberOfEpochs	32..512	256	Medium number of Epochs (medium dif. joint)
defaultNumberOfEpochs	32..512	128	Low number of Epochs (easy joint)
defaultBatchSize	10..1024	64	Training Batch-Size
learningRate	$10^{-}5..10^{-}3$	0.00017	Learning Rate for Training
minEarlyStoppingDelta	0.001..0.1	0.001	Early stopping minimum loss delta
earlyStoppingPatience	1..10	5	Consecutive Epochs where loss delta is insufficient
trainingSeed	$\mathbb{N}$	0	Randomization Seed for training reproducibility
activationFunction	SWISH/MISH/SELU/RELU	SWISH	Activation function for hidden layers
optimizer	RMSPROP/ADAM/ADADELTA	ADAM	Training Optimizer
dropoutRate	$0 \leq 0.40$	0.20	Dropout Rate for hidden layers
λ	$0 < 4.40$	1.0	Layer Width multiplier for hidden layers [6]
loss	MAE/MSE/MQE	MSE	Loss function
inheritWeights	No/Continuous/Parent	Cont.	Inherit previous encoder weights when initializing
groupOutputs	1..Outputs Or All	3	Outputs regressed by a single encoder
eigenPoses	True/False	False	Use Eigen Poses (Section 4.3)
probabilisticOutput	True/False	False	Regress Probability Density Function
onlineHardExampleMining	True/False	False	Use online hard example mining (Section 4.1)
hardMiningEpochs	1..5	5	OHME(Section 4.1) hard epochs
normalMiningEpochs	1..2	2	OHME(Section 4.1) normal epochs
ignoreOcclusions	True/False	True	Ignore occlusions recorded by 3D renderer
descriptor	NSDM/NSRM/eNSRM	eNSRM	Descriptor used [6,8,9]
EDM	True/False	False	Also append an EDM for experiments
skipConnections	True/False	True	Use skip connections
decompositionType	No/PCA/ICA/...	No	PCA/NMF/Dictionary/Factor/Dirichlet/SVD decomp.
dimensionsKept	1..All	200	Dimensions kept after dimensionality reduction
alsoKeepRawData	True/False	True	Prepend raw input before our descriptor data
outputValueDistribution	No/-/+/Balanced	Balanced	Normalize output in [-1..0],[0..1] or [-1..1]
outputMultiplier	0 .. 5	1.5	Multiplier for outputs

Table 3.2: Summary of hyper-parameters used when training MocapNETs. Each training session is tagged with a unique serial number and contains a .json file with the above stated configuration flags. Over 700 experiments where conducted in our effort to find a good configuration. It should be noted that there are many other pre-processing hyper-parameters in different parts of the dataset preparation.

Every new major idea introduced a new configuration flag in a central .json file (Table 3.2) The Tensorboard package was used to record details during training, while a package was produced after each training session allowing mixing and matching of different ensembles with various configurations in order to facilitate iterative improvements to the method.

The individual training hyperparameter tunings documented in our publications [6,8,9,20,31] gradually led to the final configurations reported in Table 3.2.

We used the Tensorflow [378] deep-learning framework, the ADAM optimizer with batch sizes of 64 samples, learning rate of 0.00017 and trained each encoder for up to $128, 256, 512$ epochs depending on their distance from the torso. Our training order follows the hand joint hierarchy order. For example for hands, we start from the wrist and progressively go through each finger from pointer to pinky with the thumb trained last. Each independent encoder for each d.o.f. of each joint is trained separately but initialized using its parent's network weights in an attempt to transfer knowledge previously acquired on our training session and to avoid trying to win the randomized initialization "lottery ticket" [401] for each and every encoder. Closer joints to the root typically have bigger motion ranges and thus require more training resources thus assigned more epochs. We terminate training if loss improvement (minEarlyStoppingDelta in Table 3.2) is less than 0.001 in 5 or more consecutive training epochs. We also use model checkpoints [402] as an ex-

tra precaution against over-fitting selecting the weights that achieved minimum loss in the training session regardless of chronological order.

Each joint hierarchy (e.g. lhand, upper body, lower body etc.) has its own dedicated training session and configuration. During early stages of the method that performed orientation class separation, we train one encoder for each degree of freedom of each joint hierarchy for each orientation class also in dedicated sessions. This means a long training procedure that took over a week to conclude when e.g. considering all the encoders for both upper and lower body in all four orientations. Smaller batch sizes could produce better training results but became impractical since they greatly decrease GPU utilization and lead to even longer training times. Similarly different activations function used required different randomization schemes. SNN [28] layers for example are effective when initialized with random samples from a truncated normal distribution centered at 0 with $\sigma = \sqrt{1/N}$ where N is the number of input units in the weight tensor.

We will deal with training experiments and experimental results in Chapter 5, however a final important consideration for MocapNETs was that being an end-to-end encoder from 2D points to BVH angles the loss optimized was done so in terms of BVH motion channels.

3.7 Coefficient of Determination

Although a loss for a positional motion channel in millimeters can give values directly interpretable as MPJPE or other error metrics, angular channels where difficult to interpret since e.g. a 1° can result in a much less pronounced absolute positional error in terms of millimeters. In order to have a better understanding of what the angular error after training meant we employed the coefficient of determination or R^2 metric. This metric provides a measure of model accuracy compared to the variation of outcomes compared to the training corpus of the model.

Assuming a dataset with N samples $y_1, \ldots, y_N$ each for each of which our model made a prediction $f_1, \ldots, f_N$ we can define the residual errors as $e_i = y_i - f_i$. We can compute the mean of the ground truth:

$$\tilde{y} = \frac{1}{N} \sum_{i=1}^{n} y_i$$

The variability of the regression output compared to the variability of the dataset can be calculated with the following formulas.

$$SS_{residual} = \sum_{i=1}^{n} e_i^2$$

$$SS_{total} = \sum_{i=1}^{n} (y_i - \tilde{y})^2$$

Finally the definition for R^2 is:

$$R^2 = 1 - \frac{SS_{residual}}{SS_{total}}$$

This metric makes overseeing network training a much more intuitive process regardless of the the peculiarities of an individual regressed degree of freedom. It provides a good intuition about the quality of model training due to the properties of R^2. A perfect fit of the model to data yields $R^2 = 1$ since it means that all error residuals (e_i^2) tend to be near 0. An encoder that cannot perform a detailed response but instead predicts values close to the mean value $\tilde{y}$ yields $R^2 = 0$, while values $R^2 < 0$ mean a bad fit to the data with predictions worse than the mean value of the data.

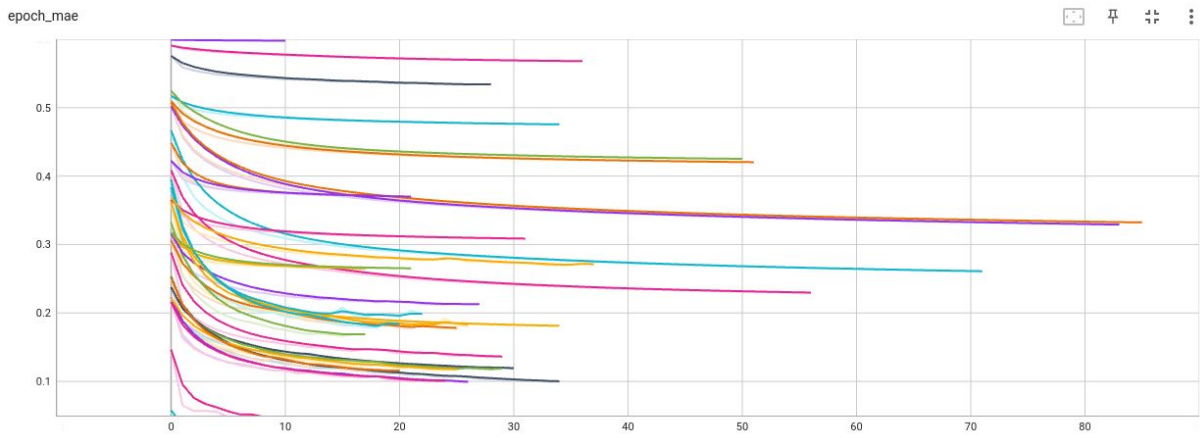


Figure 3.43: Cumulative training MAE plot for Hand training. Y axis is MAE in normalized units and X axis the number of epochs until training termination.

Plotting individual encoder training sessions using Tensorboard reveals generally smooth training sessions like the ones in Figure 3.43, with each encoder exhibiting slightly different behavior based on its difficulty and order of training. Although the independent encoder training could produce immediate BVH frames that could be consumed by a BVH player with no additional processing, later attempts to train MocapNETs that combined all outputs in a monolithic network 3.44 required normalizing outputs to the same range to make them contribute equally to the loss function. These training sessions required more memory but were completed in a lower number of epochs due to the more convex loss landscape, reaching a difficult to surpass optimum in an earlier stage of training.

3.8 Neural Network Compression

The great success of Transformer based [388] Large Language Models (LLM) networks led to complex language models like BeRT [403] and eventually GPT, LLama etc. [404]. The computational complexity of these networks combined with hundreds of thousands of users simultaneously accessing them through cloud computing, showcased the profound importance of NN efficiency to conserve computing resources. In a similar manner we envision our 3D pose regression service running on the cloud and being transparently used by a large number of client devices. Our implementation's portability allows deployment in distributed computing platforms like Google Collab where we already offer such a demo [405]. Shankar et.al. offer a comprehensive survey [406] that deals with trends in resource consumption in AI. It is very interesting that regardless of the efficiency of the computers that actually run the required computations all NN solutions consume resources that as a bottom line amount to electric power usage (and cooling) which is the main recurring cost [407] of inference.

Recent work towards the goal of better efficiency clearly showcases the importance of NN compression. The work of Jiang et.al. [408] shows that text classification is possible using just the GZIP compression algorithm and a Nearest Neighbor classifier on in-distribution data in contrast to complex LLM models. At the same time work on MLP Mixers [409] showcases that overly complicated networks can be counterproductive in-terms of inference speed. This is an additional important motivation for refining MocapNETs

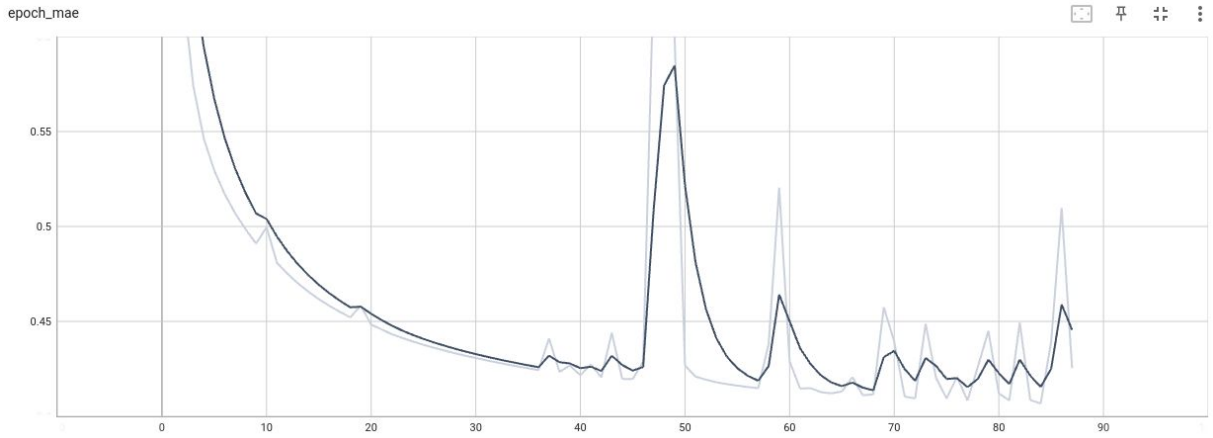


Figure 3.44: Cumulative training MAE plot for the same Hand training dataset as in Figure 3.43 when treated with a monolithic network regressing all outputs at once instead of an ensemble of independent encoders. We use the same training configuration altering just the learning rate to half of the per-encoder ensemble approach. Even with this change, training still exhibits problems after 40 epochs. Y axis is MAE in normalized units and X axis the number of epochs until training termination.

that are conceptually similar to MLPMixers. Furthermore, recent jumps in NN parameter counts highlight the importance of NN optimization/pruning as a research topic. For example, even moderate weight pruning yields substantial compute savings in massive networks like GPT-3 [410] that have 1.75×10^{11} parameters due to their immense scale. On the other hand, in smaller networks like MocapNET [9] with 1.3×10^{7} weights, there is simply not the same space left for improvement. Applying a variety of different approaches used in the literature like, model weight-clustering [411], training-aware quantization [412] and training aware pruning [413], yielded no significant performance improvements in our case. Our observations are similar those reported in works such as [414–416] and others. Although the exact factors that govern representational capacity and make compression mechanisms effective are not yet completely understood, there is active research [417] to understand, visualize and modify the internal structure of neural networks and recent works prove that "any NN with any activation function can be represented as a decision tree" [26]. These recent findings, prompt us to examine classic ML optimization methods beyond the network itself to increase regression efficiency. We. thus, decide to operate at the input level, exploring dimensionality reduction as a means of understanding and controlling the amount of sample variance contributed by each specific input degree of freedom. Then, we progressively fine tune the number of NN inputs and layer weights so as to control and optimize the network's representational capacity.

Although MocapNETs have clear computational efficiency advantages compared to other methods as we will see in the Experiments Chapter, their initial implementation was limited to operating solely on desktop computers. Motivated by mobile device use-cases and the rationale presented in the introduction of this thesis, effort was dedicated to exploit dimensionality reduction techniques to shrink MocapNETs even more and make them a viable solution for embedded devices like VR headsets, mobile phones, automotive applications, smart sensors, HCI applications, and a building block for more complex recognition applications.

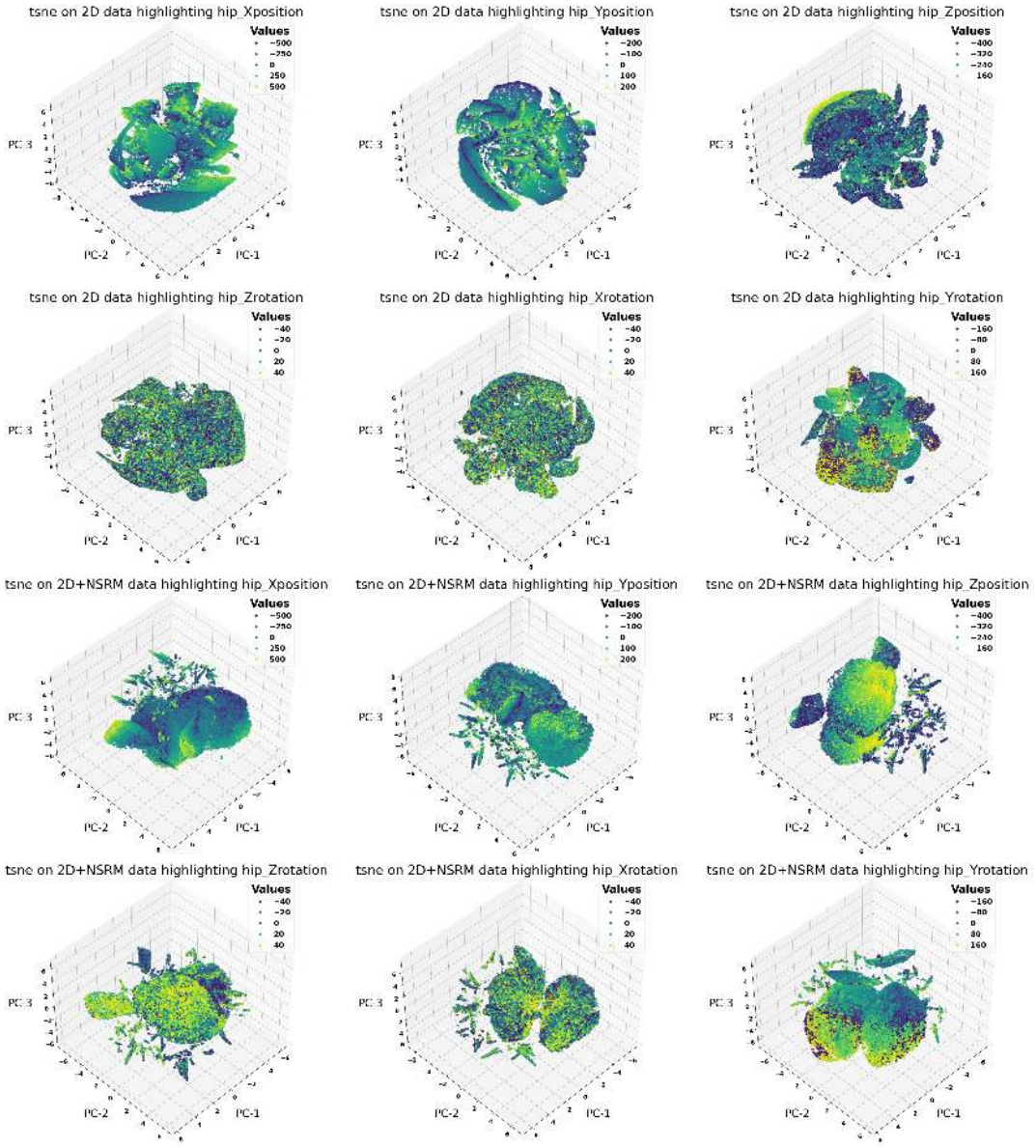


Figure 3.45: TSNE [33] 4D (3D+color) clustering of our training samples. First 2 rows of raw 2D data, Bottom 2 rows of 2D data + eNSRM. Each plot represents a different degree of freedom values. We observe that the eNSRM matrices promote better data clustering.

Our NN compression effort is also motivated by the Ambient Intelligence Program [418] of FORTH. Most of its current modules involve RGB-D sensors that are costly, do not operate correctly in spaces with sunlight since it interferes with their infrared signal, and need a standalone small form-factor computer to perform vision. Dimensional reduction enables our 3D body pose estimation to become suitable for off-the-shelf mobile devices. We experiment using a cheap camera-equipped mobile device, streaming poses through

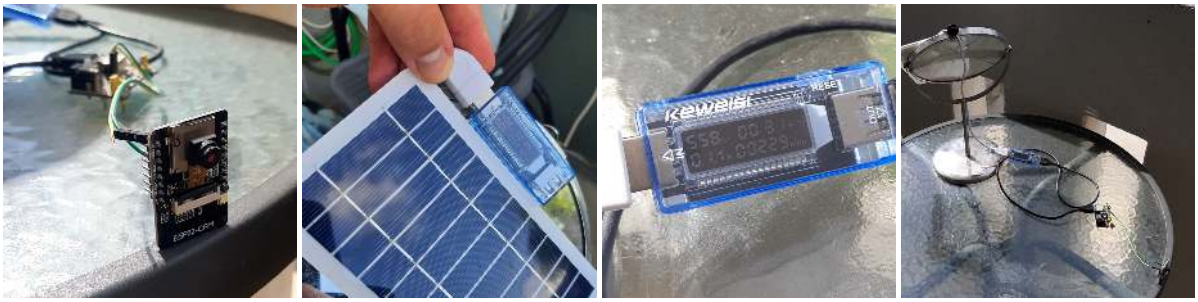


Figure 3.46: Low cost camera sensors like the ESP32-CAM consume so little power that they can be directly powered from a small USB photovoltaic cell. The picture shows the components of an experimental setup that combined with the flexibility of our method allowed us to perform successful 3D pose estimation experiments streaming images via WiFi at a fraction of the cost that would otherwise be needed.

WiFi at interactive framerates to the Meta Quest 2 VR headset, while following the optimizations presented in this Section we where able to deploy MocapNET to a Raspberry Pi 4 ARM based computer. A solution based on a RPI4 computer would involve a total cost of $\approx 80 Euros$ per module at the time of writing, while a cloud based solution with a central pose regression server and camera sensors such as the ESP32-CAM systems that cost $\approx 10 Euros$ and have enough capacity just to encode and transmit images through WiFi would be orders of magnitude more cost-effective allowing for a more immersive ambient intelligence experience.

A key design characteristic of any neural network is the number of its network parameters or weights. With a larger number of parameters, a network gains more representational capacity and can accumulate more information during the training session. Higher parameter counts result in larger networks that can tackle more complex tasks. At the same time, these are slower to execute, harder to train and may become prone to over-fitting. In an attempt to retain a small number of weights, convolution operations combined with pooling layers allowed [419] for reducing the aperture of the receptive field of each layer parameter, keeping a smaller number of model parameters that after multiple convolution and pooling layers, gradually learns input global features. However, this technique mostly applies to pictorial input due to the locality of the color/luminance information. There are methods that can regress 3D output from RGB image source. However, MocapNET as well as many other methods use the so-called 2-stage architecture regressing the 3D points from 2D input as a discrete step after extracting the 2D points from an RGB image. The 2-stage architecture enables much richer augmentation during training and compatibility with a wide range of 2D joint estimators. Since the 2-stage problem departs from the original design considerations of convolutional neural networks, it requires a different solution. The design philosophy of MocapNETs is, instead of correlating inputs inside the network, to perform feature correlation directly at the input layer and then just use the effective number of network parameters to regress the output 3D angles. Since all input fields are immediately correlated and there is no spatial locality like RGB input, the network is densely connected (each layer weight is connected to all weights of the subsequent layer). This, however, creates the disadvantage of a geometric growth of added connections with each additional layer. Combined with the ensemble architecture, this has the potential of creating very large networks. Furthermore, since the operation of the network is opaque, we have no insight on how to trim the input layer (which is the largest) in a meaningful

way.

Due to this issue, the original MocapNET formulation proposed the use of a parameter λ (explained in depth in [6]) that acted as a scaling factor on hidden NN layers. This was a coarse solution that did not take into account the input distribution at all. It should be noted that state of the art (in terms of 3D accuracy) methods like Elepose [76] are also attempting similar dimensionality reduction approaches even while following the convolutional paradigm, instead of a densely connected network as the one we use. However, they do not address IK and do not use a 2D joint descriptor but perform PCA on just raw 2D input.

3.8.1 Dimensionality Reduction vs. NN Capacity

We study the application of dimensionality reduction techniques like Principal Component Analysis (PCA) to the 323 MocapNET input elements of 1.5M available training samples with the goal of reducing and optimizing the parameter count of the network thus making it applicable on mobile devices. Through PCA we gain knowledge of the percent of input variance maintained, despite the lower input parameter count. The observation of the PCA Scree plot reveals that just using the first 20 PCA dimensions we can retain 95% of the variance of the network input, 80 PCA dimensions can encode 99.8% of the variation while 210 PCA dimensions encode 99.9999% of that variation. These results show that we can effectively discard a very large amount (more than 35% reduction) of our input since the information it contributes is already present in other inputs. This is a very significant reduction, especially in the context of an expensive densely connected architecture.

However, naïve input reduction has unforeseen consequences, since NNs shrink so much that they no longer maintain the capacity to solve the problem. The baseline method [9] using a $\lambda = 1.0$ [6] created encoders of 152K parameters each, which combined in a full ensemble amount to 5.5M parameters for the upper body and 3.7M for the lower body. Using a PCA basis of 50 dimensions that maintains 98% of input variance, and keeping the network exactly as described in the baseline [9] encoders degenerate to ~ 4.5K parameters, each. For comparison, the first 323 inputs from the first layer to the second feature close to 40K parameters, an order of magnitude more. The concatenation of shrunk encoders creates ensembles of 162K parameters for the upper body and 120K parameters for the lower body. To better understand the extreme degeneration of the network we can think about the following: with naïve input reduction we are essentially trying to tackle the whole 2D to 3D BVH estimation problem using the number of weights previously devoted for a single degree of freedom of the baseline approach. This is clearly not feasible and the NN is unable to correctly respond to input due to its diminished capacity. To remedy NN shrinking and since PCA enables fine tuned control over input and knowledge of its explanatory impact, we also resort to using the λ variable [6]. However, in [6] λ assumed values greater that 1, to shrink the network [6]. In contrast, in this work we set $\lambda < 1.0$ in an effort towards maintaining the NN size, despite the very small input.

3.8.2 The MocapNET Special Case

In spite of the great success of Transformer [388] networks, recent work on Multi-Layer-Perceptron Mixers [409] showcases that overly complicated networks can be counter productive in terms of inference speed. This is an additional important motivation for refining MocapNETs that are conceptually similar to MLPMixers. Furthermore, recent jumps in NN parameter counts highlight the importance of NN optimization/pruning as a research topic. For example, even moderate weight pruning yields substantial com-

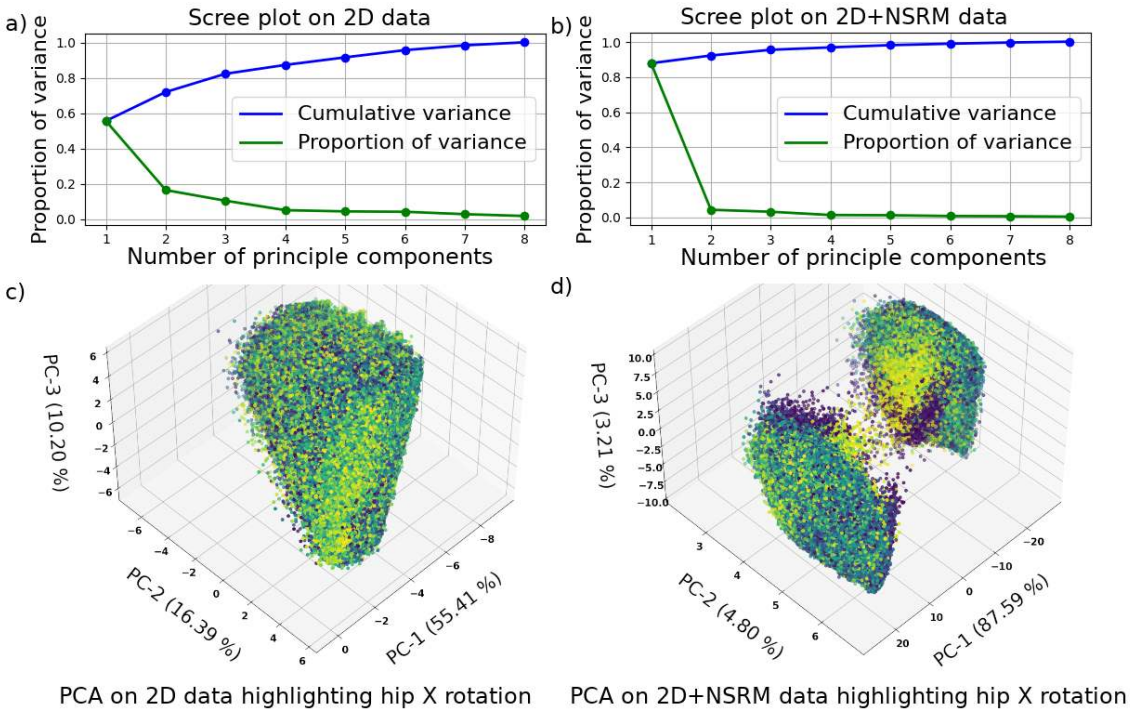


Figure 3.47: PCA results of 605K CMU BVH [16] pose training samples. From left to right. a) Scree plot using only 2D points as PCA input. b) Scree plot using both 2D points and NSRM matrices. c) 3D clustering of PCA samples using only 2D input with positions corresponding to the 3 most significant PCA dimensions and color highlighting X (pitch) rotation values. d) Same plot using both 2D + NSRM matrices. We observe that the NSRM formulation causes a clearer separation of samples in contrast to raw 2D points.

pute savings in massive networks like GPT-3 [410] that have 1.75×10^{11} parameters due to their immense scale. On the other hand, in smaller networks like MocapNET [9] with 1.3×10^{7} weights, there is simply not the same space left for improvement. Applying a variety of different approaches used in the literature like, model weight-clustering, training-aware quantization and training aware pruning [413], yielded no significant performance improvements in our case. Our observations are similar those reported in works such as [414–416] and others. Although the exact factors that govern representational capacity and make compression mechanisms effective are not yet completely understood, there is active research [417] to understand, visualize and modify the internal structure of neural networks and recent works prove that "any NN with any activation function can be represented as a decision tree" [26]. These recent findings, prompt us to examine classic ML optimization methods beyond the network itself to increase regression efficiency. We. thus, decide to operate at the input level, exploring dimensionality reduction as a means of understanding and controlling the amount of sample variance contributed by each specific input degree of freedom. Then, we progressively fine tune the number of NN inputs and layer weights so as to control and optimize the network's representational capacity.

At the same time, sizeable input descriptors that correlate all 2D input points using NSDM [6], NSRM [8]

or eNSRM [9] descriptors combined with the densely connected layer architecture of [6,8,9] generate a large number of NN parameters that leave potential for improvement. Dealing with the high dimensional space of 87 degrees of freedom of the BVH representation of the upper/lower human body [8] naturally motivates the investigation of mathematical techniques to correlate samples and project them to a smaller domain. Although there is active research [417] to understand, visualize and modify the internal structure of neural networks, a mathematical formulation that can allow us to perform similar analysis to the NN input can also be useful to that end. By understanding and controlling the amount of sample variance contributed by each specific input degree of freedom we can attempt to reverse engineer the neural network progressively fine tuning the number of network inputs and layer weight count and invariably controlling and optimizing the network's representational capacity.

3.8.3 Overview of Common Techniques

A classical tool for dimensionality reduction is Principal Component Analysis (PCA) [420]. We choose to employ it in the context of our work because (a) it preserves the global structure of the data, (b) it does not involve hyper-parameters other than the number of dimensions maintained, and, (c) it is a deterministic algorithm that allows us to decide how much variance of the input data to preserve. Dimensionality reduction can be performed in several other ways. A popular technique is via auto-encoders which are symmetric NNs that feature an information bottleneck which acts as the lower dimensional encoding.

However, such encoders enable no transparency to their internal organization and the identification of the contribution of each of the reduced problem dimensions. What is more, being densely connected, they further degrade parameter count and, thus, performance due to their input size. T-distributed Stochastic Neighbor Embedding (t-SNE) [33] is another option for dimensionality reduction which we experimented with as seen in Figure 3.45. Although it provided valuable insight for exploratory data analysis by allowing us to study how the various degrees of freedom gain cohesion using the eNSRM descriptor, we could not adopt it as a dimensionality reduction method for a series of reasons. These where its non-deterministic (randomized) operation, its inability to preserve variance (instead using sample distance) and its maximum of 4 output dimensions. Finally, UMAP [421] is an improved algorithm with similarities to t-SNE constructed from a theoretical framework based in Riemannian geometry and algebraic topology that preserves more of the global structure with superior run time performance. However due to UMAP being a relatively new algorithm we where not able to properly incorporate its implementation and dependencies to the rest of the developed code-base given our time constraints and thus we did not use it for technical reasons. Non Negative Matrix Factorization [422] is not applicable in our case since our input matrices contain negative elements. Similarly, although we attempted to use Dictionary decomposition [423], the size and variance of our data did not allow the algorithm to converge. Finally, Principal Component Analysis offers many variants out of which we selected Full SVD PCA [424] and Randomized PCA [425,426] omitting other variants like KernelPCA, SparsePCA based on applicability for our data. Finally we also experimented with Factor Analysis [427] and Independent Component Analysis [428] in an attempt to further study how different reduction algorithms impact the solution of our problem.

3.8.4 Sample Visualizaton Using Different Decomposition Techniques

Quantitative results using various different dimensionality reduction techniques will be presented on Chapter 5. Concluding this Section we present visualizations for the Root X,Y,Z position and Z,X,Y root rotations

using ICA (Figure 3.49), Factor Analysis (Figure 3.50), Dictionaries (Figure 3.51), PCA (Figure 3.52), Sparse PCA (Figure 3.53) and Incremental PCA (Figure 3.54).

Each image uses 3 dimensions to position samples in the 3D plot and colors them based on the magnitude of the ground truth value for the specific degree of freedom of the problem. Although high-dimensional space is very difficult to comprehend and visualize, we can immediately observe that 2D points on X and Y axis exhibit regularities in all cases. We also observe that the Z component (depth), Z Rotation and X rotation are very noisy in the raw 2D point case. On the other hand the Z Pos, ZRot and XRot seem to show much better clustering under all decomposition methods. Root Y Rotation also seems to benefit from decomposition while it displays an intermediate clustering quality in the raw 2D plots. We believe that the combination of the use of a descriptor with a dimensionality reduction technique improves the locality of samples therefore allowing neural networks to perform regression on a more predictable input signal that allows smaller networks.

We successfully reduce the size of the MocapNET NN [9] while marginally improving its accuracy in terms of Procrustes M.A.E [42] by applying dimensionality reduction on its input. We achieve NN weight savings from 25% up to 80% depending on size/accuracy trade-offs. Extensive experiments summarized in Table 5.20 reveal sweet-spots on configurations that use 105K params and 60/150 PCA dims. We also open the way to 32K encoder networks that can effectively tackle aspects of the problem while being much easier to compute. Despite high-dimensional pose space and densely connected neural networks being opaque, we gain insight on the impact of various input and network sizes. PCA allows us to visualize the high-dimensional input and observe the improved sample coherence when utilizing NSRM descriptors (Figure 3.47). This newfound knowledge enables creation of compressed MocapNETs deployable on embedded devices. This in turn, enables user perception on mobile devices for 3D avatars in VR, automotive applications, robotics, human computer interaction and more.

Utilizing the python TF-Lite backend of Tensorflow, the ensemble with 150 PCA input dimensions achieved an inference rate of 23Hz on a Raspberry Pi 4 (RPi4) device (Figure 3.48) with 2GB of RAM. Given that flagship smartphones commonly feature double the number of cores clocked at higher frequencies compared to the relatively low-power RPi4 Broadcom BCM2711 Quad core Cortex-A72 SoC, the work we presented now makes MocapNET a viable 3D Pose estimator for mobile devices.

3.9 Generative Pose Fine-Tuning

We have thus far described a neural network architecture that assuming a specific armature and its 2D projections can regress a 3D configuration and inverse kinematics in an end-to-end fashion. A formulation like this allows tackling a very high dimensional problem with unprecedented effectiveness. However such an approach has still some fundamental limitations that lead us to the need for a complementary generative fine tuning module. One of these problems is that due to the lack of any appearance encoding or queues from the original RGB input, the BVH armature dimensions which the NN is trained against might not properly model an observed subject in-the-wild. Using an “average” human phenotype improves the situation since on average it will be closer to most observed humans compared to e.g. training on a person that is very tall or short, however to get better result accuracy we need to perform retraining for each specific subject. Another problem is that despite the precautions taken like high-dropout, early stopping etc the network can do nothing to overcome corrupted 2D input. 2D joint estimators are noisy and since each frame is regressed in isolation there is no way for motion queues over multiple frames to be used to reduce these errors. Furthermore estimations on exotic 2D pose inputs can often be biased, since poses that are not present in the

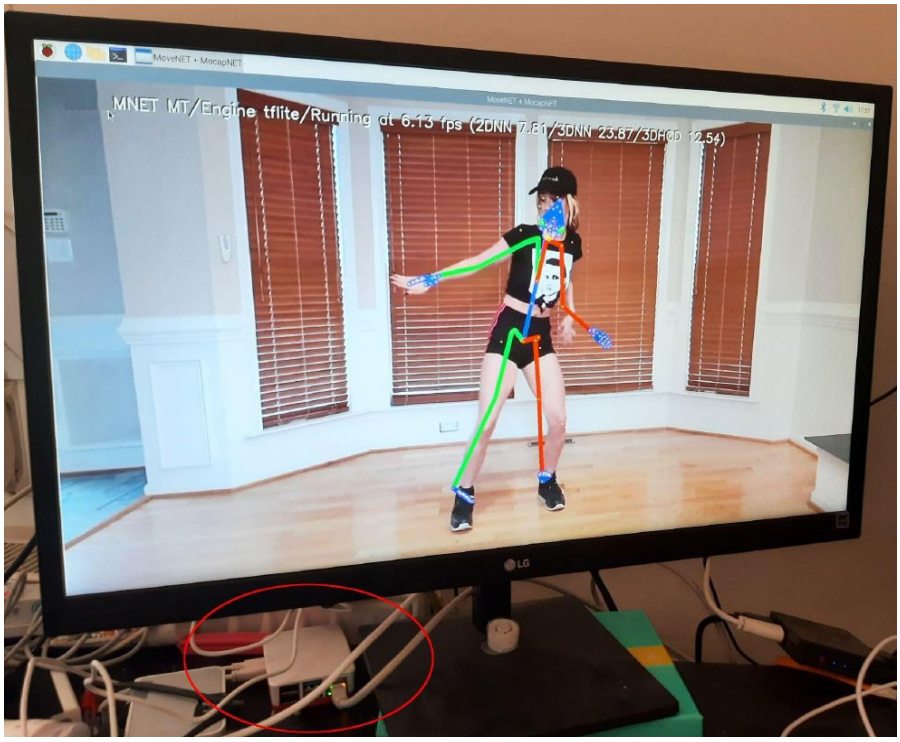


Figure 3.48: Multithreaded MocapNET deployment on a Raspberry Pi 4 running at 23.87Hz, with the BlazePose [34] 2D joint estimator running at 7.81Hz and achieving an interactive 6.13Hz update rate on screen (including visualization time).

training dataset will sometimes be regressed to a lower fidelity. Regardless of erroneous input, the network itself can some times give noisy results with slight input changes producing noisy results even with relatively accurate input, a common issue with NNs as in the case of "one pixel attacks" [429]. Finally an important consideration for us, towards the development of a generative module was accuracy improvement without overly complicating the NN. As we saw on Section 3.5.1 having the capacity for finer resolution pose accuracy requires neuron and layer increases that drastically decrease computational performance for marginal accuracy benefits, while at the same time making training harder and making the network prone to overfitting, vanishing gradients etc. Dividing the networks for different orientation classes 3.5.6 can mitigate this situation while increasing the training time to infeasible (in the course of a PhD) levels and introducing a classifier that becomes a central point of failure. Training with more samples as seen in Section 3.5.1 has its own limitations that are showcased using the R^2 metric (Section 3.7). Achieving an R^2 score of 0.95 means that our network is already performing very well without leaving a lot of room for improvement.

3.9.1 Overview of Common Generative Techniques

Hybrid approaches like the "Robust model-based motion tracking through the integration of search and estimation" of Lowe et.al (1992) [430] have been long standing. Coming from a background of classic computer vision techniques with experience in Hybrid methods from our WACV2018 publication [215] a gener-

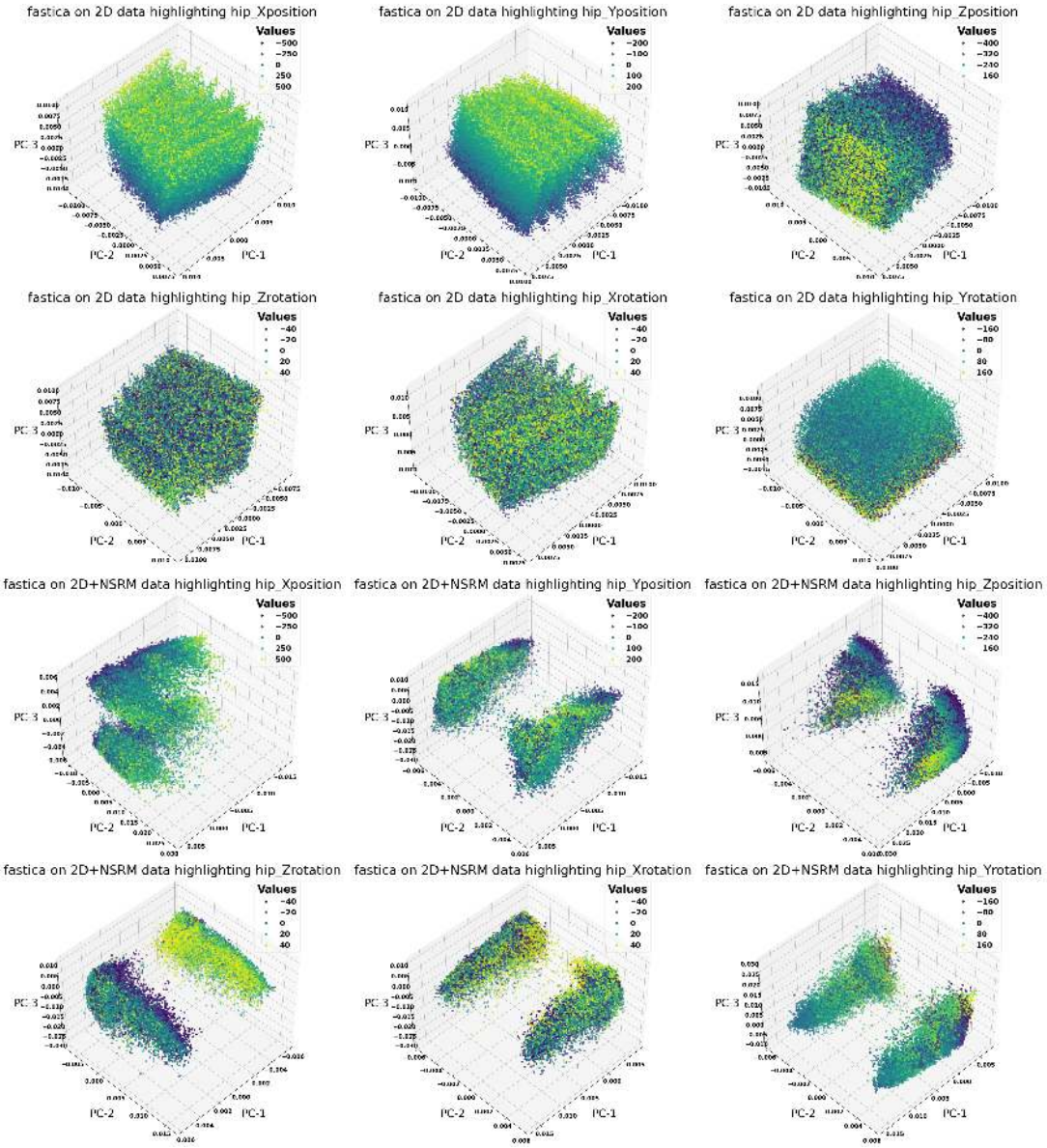


Figure 3.49: Fast-ICA clustering of samples higlighting different degree of freedom values

ative module was an obvious addition to the original MocapNET [6] formulation to build on the strengths of the NN output while complementing their above stated weaknesses. Hybrid approaches combine both discriminative (neural networks or other machine learning models) and generative (optimization-based) components for improved accuracy, robustness, and efficiency. In the case of MocapNET the Discriminative Component of the method is the 2D Joint Estimation NN followed by the Descriptor/Dimensionality reduction step and the MocapNET ensemble. The generative component relies on being initialized very close to the "correct" solution and then performing minor refinements to improve the initial estimate provided by the discriminative component. This refinement is essential to improve the initial estimates that

may be noisy, imprecise, or subject to errors.

Prior to the advent of Neural Networks, iterative pose refinement methods where widely used in computer vision and robotics for solving pose estimation problems. Very popular techniques include:

- **Levenberg-Marquardt Algorithm**: Widely used for non-linear least squares optimization problems, including inverse kinematics (IK) and pose refinement.
- **Iterative Closest Point (ICP)**: Primarily used for aligning point clouds or 3D models, ICP iteratively refines the transformation between two point sets to minimize the distance between corresponding points.
- **Particle Swarm Optimization (PSO)**: A genetic population-based optimization algorithm inspired by the social behavior of birds flocking or fish schooling, often employed for pose estimation problems.
- **Sequential Monte Carlo Methods (Particle Filters)**: Probabilistic methods that use a set of particles to represent a distribution and iteratively update the particles based on observed data, applicable to pose estimation in dynamic systems.
- **Gauss-Newton Method**: Iterative optimization algorithm for least squares problems, frequently used for pose estimation tasks.
- **Bundle Adjustment (BA)**: An iterative optimization technique used to refine the parameters of a 3D reconstruction model based on observed feature matches, widely used in structure-from-motion and visual SLAM applications.
- **Stochastic Gradient Descent (SGD)**: An iterative optimization method used in training deep learning models, and can also be adapted for pose refinement tasks.
- **Covariance Matrix Adaptation Evolution Strategy (CMA-ES)**: An evolutionary optimization method used for non-linear and non-convex problems, including pose estimation.
- **Nonlinear Conjugate Gradient Method**: An iterative optimization technique for solving large-scale non-linear unconstrained problems, also applicable to pose refinement tasks.
- **FABRIK (Forward and Backward Reaching IK)**: An iterative algorithm for inverse kinematics of articulated structures, commonly used for real-time animation and control of robotic arms and characters.

These methods differ in their underlying principles and application domains, but they all share the common characteristic of iterative optimization to refine pose estimates and achieve better alignment with observed data. After a brief examination with methods like Levenberg-Marquardt least squares optimization (using the CERES [431]) and Particle Swarm Optimization we found them to be overly complicated and not a good fit for our MocapNET formulation. While they exhibited the capability of convergence even when initialized with poses far from the optimum, their slow convergence speed and complexity killed real-time operation.

3.9.2 Hierarchical Coordinate Descent (HCD) Inverse Kinematics

The inverse kinematics solver is used to refine the pose regressed by the neural network. IK solvers typically rely on a non-linear least squares optimizer, with the CERES [431] solver being a very popular choice. However, from a computational point of view, this is an expensive operation. The problem is expressed as a series of 2D Euclidean distances that we need to minimize by properly tuning parameters that describe the underlying 3D model. These optimizers in turn typically internally rely on the Levenberg-Marquardt [432, 433] algorithm which in a nutshell can be described using the equation $x_{n+1} = x_n - (H + \lambda I)^{-1} \nabla f(x_n)$. Depending on the objective function f of the sum of 2D distances of observations and hypothesis and the Hessian matrix H (of second order partial derivatives) we can guide the optimization procedure towards x values that offer a better explanation for our observations. This procedure is non-trivial since it involves calculating matrix inversions among other expensive operations but can robustly handle non-linear multidimensional problems. One of our observations was that the underlying equation of levmar is actually very close to the Newton-Raphson method [434] when $\lambda \to 0$ and Gradient Descent when $\lambda \to \infty$. In our 3D human pose estimation problem, we obtain consistently, fairly accurate joint estimations. Moreover, each neural network encoder is conditionally independent from the others. This suggests the appropriateness of an iterative solution to the pose refinement problem. Additional inspiration comes from the performance of efficient, heuristic IK methods like FABRIK [435] that iteratively traverse the kinematic chain by making individual improvements to each joint, also inspired us to formulate our method that also shares some conceptual similarities with Semi-stochastic coordinate descent [436]. Our method relies on output that is relatively close to the correct solution, that has conditionally independent errors across the joint parameters (i.e. the rotation r_x of the shoulder might be miscalculated but that does not necessarily mean that angle r_y for the same joint will also be incorrect) is highly efficient in computing and parallelizable.

Specifically, we think of our IK problem as the refinement of a hypothesis vector h that consists of individual 3D human pose parameters resulting from the neural network of the previous step. We also consider the objective function E_{2DJ} that quantifies the mean squared error (MSE) of the 2D joints projected, compared to the 2D joints observed.

$$E_{J2D}(h, o) = \frac{1}{m_J} \sum_{i=1}^{m_J} |j_i^h - j_i^o|^2. \tag{3.11}$$

We assume that changes in, e.g., the parameters of the left arm will not affect errors on the right leg, therefore we decompose the human body into 6 kinematic chains. The first kinematic chain C_1 consists of hips, shoulders and neck. The rest of the kinematic chains are C_2 (abdomen, neck, shoulders), C_3 (right shoulder, elbow, hand), C_4 (left shoulder, elbow, hand), C_5 (right hip, knee, heel, toe) and C_6 (left hip, knee, heel, toe).

For a certain kinematic chain, we define an iterative error minimization scheme. At the n^{th} iteration of this process, we modify each parameter c of the chain by d_c^n defined as:

To summarize the IK method we use familiar terminology from the gradient descent formulation assuming that each parameter is serially labeled the d_c the attempted delta correction for parameter c is:

$$d_c^n = \beta d_c^{n-1} + l_r \left(\frac{E_{J2D}(h_{n-1}, o) - E_{J2D}(h_n, o)}{2(d_c^{n-1} + e)} \right). \tag{3.12}$$

In the above equation, β is a momentum control parameter we set to 0.9 and $e = 0.0001$ is used to avoid division by zero. $l_r = 0.001$ is a learning-rate-type of parameter that controls the rate at which the change of

the error affects the change of a parameter. For a certain joint of the kinematic chain, we alternate between its x, y and z rotational parameters for 30 epochs and only accept combined value updates if the achieved objective function is improved compared to the initial starting point.

We finish the procedure after going through every joint of the kinematic chain for 5 iterations or after 3 consecutive unsuccessful loss updates. We experimentally identified the 5 iteration sweet-spot after synthetic experiments on CMU [18] data as seen in Fig 3.60 (right). Kinematic chains are optimized in groups, with the first being C_1, C_2, followed by C_3 to C_6 which can be considered in parallel.

First, we align the torso of our hypothesis to the observation. We consider this to be the kinematic chain C_0 which consists of hips, shoulders and neck. The parameters we need to optimize in this chain is hip translation $t_x^{hip}, t_y^{hip}, t_z^{hip}$ and hip rotation $r_x^{hip}, r_y^{hip}, r_z^{hip}$. Without having correct values on these first chain all other joints are bound to never be properly aligned to the observation.

Then, we proceed with the second "group" of sub-problems that need to be solved to improve the rotations of the abdomen, neck and the 4 limbs of the armature. These are kinematic chains C_1 (abdomen, neck, shoulders), C_2 (right shoulder, elbow, hand), C_3 (left shoulder, elbow, hand), C_4 (right hip, knee, heel, toe) and C_5 (left hip, knee, heel, toe). The leaves of each chain are only considered to be end-effectors for the particular sub-problem, do not have parameters to optimize but are very important terms of the loss function E_{2DJ} since errors that stack across each kinematic chain are very indicative when being compared to observations. Chains C_0 to C_2 are considered as the first group for optimization and chains C_3 to C_5 as a second group that relies on the first. This is the reason why the two groups are optimized iteratively taking turns.

In contrast to the baseline body [8] estimation that always executes on the same chains, the extension of the method to Hands [9] implemented a logic switch that alters C_1 from standalone to holistic operation depending on occlusions. In case of an upper body present in our observations, C_1 begins at the shoulder joint including elbow, wrist and finger metacarpal bones as end points. If 2D observations carry no body information C_1 begins at the wrist extracting an absolute 3D orientation of the hand. Thus, C_1 acts as a mediator chain, propagating knowledge of the upper body to hands and vice versa. Regardless of C_1, HCD concurrency is not affected since e.g. hand optimization has no bearing in lower body results. Dependencies only rise on upper body+hand execution but we are still able to execute all HCD sessions in parallel syncing solution updates on each iteration.

For a given hypothesis h, which consists of the vector $t_x^{hip}, t_y^{hip}, t_z^{hip}, \ldots, r_x^{lheel}, r_y^{lheel}, r_z^{lheel}$ we render the skeletal model $\mathbf{H}$ to obtain 2D joint locations directly comparable to the actual, observed joint locations, trying to minimize the error E_{j2D}. The objective function

$$E_{J2D}(h,o) = \frac{1}{m_J}\sum_{i=1}^{m_J} w_i |j_i^h - j_i^o|^2 .$$

While examining each parameter of each chain we are faced with the choice of increasing it or decreasing it by a very small amount that we call lr, since it resembles a learning rate. We can then calculate the loss function again after the change and calculate the parameter change gradient based on the observed change.

The method iteratively optimizes each parameter of each joint of each of the kinematic chains by performing small delta changes to each coordinate and studying how they affect our loss function. If the loss function is negatively impacted the gradient sign will be negative appropriately shifting deltas to the other direction. Since the neural network has brought us close to the solution and the gradient of our loss function captures the changes that need to be performed on our parameter vector by making small steps in the

right direction for each of the parameters we gradually improve the initial solution. The name "hierarchical coordinate descent" briefly describes accurately reflects the method that performs gradient descent in coordinates while leveraging the joint hierarchy for performance reasons.

This particular IK formulation is very practical to compute since at each point only the joint and its conditionally independent children need to be modified and thus re-computed. This allows us to perform IK in parallel for chains 3 to 5, needing to perform only a very small total number of operations.

Even in a naive serial implementation of the algorithm using simple floating point arithmetic (without utilizing vector extensions like MMX,SSE,AVX etc) to perform all of the 4x4 matrix multiplications needed in the CPU of on an outdated Pentium Core 2 6600 @ 2.40GHz the optimization code runs at over 50 fps for 3 iterations of the above mentioned procedure when using a lr of 0.001. Further decreasing the learning rate leads to smaller steps executed which take more time to improve and large learning rates lead to divergence since the optimization procedure starts oscillating further and further from the initial solution as is the case with regular gradient descent.

A final minor improvement compared to the baseline method is that we introduce a Butterworth filter [437,438] on derived parameters when processing video files in order to have maximally flat magnitudes on the recovered motion components, something that translates to smoother BVH recordings. They exhibit less noise and are more visually pleasant on playback.

As already mentioned the task of converting a 2D pose to a 3D skeleton is ill-posed and very challenging since each 2D projected skeleton can be derived by arbitrarily many different 3D skeletons depending on the intrinsics of the camera system and the subject limb dimensions. This very difficult task is mainly handled by MocapNET ensemble giving a coarse 3D pose estimation.

The HCD refinement module receives the 3D pose regressed by the neural network and refines it, optionally also having information about skeleton dimensions and camera intrinsics to improve estimation accuracy.

The two modules operate in tandem and complement each other. The deployed neural network ensemble can recover a plausible 3D human pose in an incredibly complex high=dimensional landscape, but is trained with a fixed skeleton and camera model and, thus, cannot by design produce very accurate results.

The IK module, on the other hand can be configured on the fly, with no additional training to target any camera system or observed skeleton dimensions but is unable to perform 3D pose estimation if not properly initialized close to the actual solution. It is especially capable to correct individual encoder error like the ones produced by our MocapNET ensembles. By having a neural network that can robustly handle the bulk of 2D joint input regression and an inverse kinematics module that can be easily configured to closely match any combination of arbitrary skeleton dimensions and camera system regardless of the executed action we get the best of both worlds, providing an accuracy boost to the final method that is otherwise not attainable using just one of the building blocks of the presented method.

3.9.3 Multiplexing HCD After Studying Our Motion Capture Domain

Having implemented a BVH library that performed 2D projections and 3D transforms of armatures both in order to create the training samples used to facilitate training the NN and for the interpretation and playback of the resulting poses, this library presented the opportunity to also perform back-projection and compare 2D projections of a regressed pose compared to the 2D observation using the Mean Per Joint Position Error (MPJPE) during runtime. One of the early observations using the reprojection module was that the independently trained encoders tended to also independently produce errors that did not affect the rest

of the BVH motion vector. An example to make this clearer is that assuming that the right arm consists of RShoulder, RElbow, RHand for a total of 9 degrees of freedom it would often be the case that just one of the values would exhibit a high degree of error. Although errors in e.g. the RShoulder would disproportionately influence MPJPE error because it would affect the kinematic chain of all of the children of the joint and penalize all of their positions. Recovering after this types of errors could be done without exploring the very high dimensional space by only iteratively tweaking the particular degree of freedom that caused the problem. As we will see later in this Section in Figures 3.55, 3.56 and 3.57 when altering one degree of freedom the optimization landscape can be easily traversed. Given the multi-threaded capabilities of modern CPUs these error corrections can also happen for big chunks of the kinematic chain in parallel without significant performance penalties (except for the lack of communication that will also be discussed). After these gradual observations the implementation took form in a multi-threaded iterative optimization loop. Studying the breakdown of processing time across functions and CPU instructions (Figure 3.63) another optimization to improve efficiency quickly became apparent. During calculation of BVH frames each joint/node had its own allocated local transform. Although the armature consisted of 165 articulated joints, the alterations performed when refining e.g. the RShoulder would only affect its own kinematic chain. This allowed an incredible speedup that became even bigger after switching the 4x4 matrix routines from regular unoptimized C code (Listing 6) to SIMD SSE2 capable code (Listing 7).

Designing a gradient descent-based inverse kinematics method for high-dimensional inputs involves several crucial concepts that play essential roles in the optimization process:

- **Convex Function Local Minima = Global Minima**: In the context of inverse kinematics, the objective function, which represents the error between desired and actual end-effector positions, should ideally be convex. Convexity ensures that any local minimum found during gradient descent is also the global minimum, guaranteeing convergence to the desired solution.
- **Graph Above Tangent**: This concept signifies that the graph of the convex objective function lies above its tangent line at any point. This property ensures that the function is always increasing as we move towards the minimum, aiding gradient descent in finding the optimal solution effectively.
- **First Order Optimality**: First order optimality conditions, like the gradient being close to zero, are crucial for convergence. In inverse kinematics, this implies that the Jacobian matrix's elements should approach zero to indicate that the system has reached a solution or a local minimum.
- **Jensen Inequality**: Jensen's inequality is used to relate the expected value of a convex function to the convex function of the expected values. In the context of inverse kinematics, it can help in establishing bounds or expectations on the error between desired and actual end-effector positions.
- **G-Lipschitz Continuous**: Ensuring that the objective function is Lipschitz continuous with a Lipschitz constant 'G' helps in controlling the rate of convergence during gradient descent. Smaller 'G' values imply faster convergence, which is valuable in real-time applications like robotics.
- **L-Smooth**: An L-smooth function has Lipschitz continuous gradients with a Lipschitz constant 'L'. In the context of inverse kinematics, this property ensures that gradient-based optimization methods, like gradient descent, can efficiently approach the solution without oscillations or slow convergence.
- **m-Strong**: In optimization, a function is m-strongly convex if its Hessian matrix is bounded below by a positive constant 'm'. Strong convexity accelerates convergence by providing a tighter control over the curvature of the objective function, making gradient descent more efficient.

These concepts can collectively ensure that the optimization process is both efficient and effective. They help guarantee convergence to a minimum of a convex objective function, provide control over the optimization process, and establish mathematical foundations for the reliability of the method.

Taking these properties into consideration and also taking into account the conditional independence of our encoder outputs as well as the parallel processing capabilities of multi-core systems we set to study our novel generative inverse kinematics refiner built from the ground up to complement our MocapNET ensembles.

Having the rich CMU Mocap Database [16] we sample from "the posterior" distribution of our problem, iteratively rendering ground truth while altering specific degrees of freedom and monitoring the MSE loss fluctuations. We observe different behaviors for different poses and different parts of the body. Indicative of the loss patterns observed are Figures 3.55, 3.56 and 3.57. Due to the discrepancies of multiple degrees of freedom there were no single rules that can be used to improve the optimization procedure. Some important common features over all data where that the recorded loss landscape was (predictably) continuous, that in most cases it exhibited multiple minima due to symmetries and inherent 2D point ambiguity.

After a first study of our loss landscape, to further constrain our problem we introduced two sets of optimization limits to enhance HCD operation. The first set enforces mechanical limits as hard constraints across all updates. This measure excludes portion of pose space that might some times even exhibit lower loss than valid configurations and thus both occupies optimization time and can produce lower quality results. Guaranteeing output in the range of valid rotations of the BVH training set effectively mitigates this kind of problems.

A second optimization adopted after not being able to easily derive a set of gradient parameters by studying the loss landscape was to once again look for clues in our MocapNET output in the hope of increasing HCD efficiency. We often observed exploding gradients that needlessly occupied CPU time despite losses increasing and errors diverging, since HCD execution continued for a preset number of iterations and epochs. To combat this, we note loss/MAE achieved for each NN encoder which informs us about the average expected error and thus HCD correction. Using training MAE as a maximum delta limit for each HCD epoch, gradient explosions are suppressed, while again improving efficiency at the cost of quickly reaching outlier poses that drastically depart from what the ensemble can correctly handle.

The original HCD formulation was tailored to the independent encoder ensemble architecture of MocapNET. Subsequent improvements to the method included multiplexing encoder outputs in an effort to regress one, two, three or even all outputs using a monolithic encoder. In a similar fashion we explored potential room for similar improvements in the HCD module. Due to the nature of kinematic transformations it quickly became apparent that repeating "essentially the same" calculations to perform matrix multiplications for each joint 3 times, one for each degree of freedom we could perform "loop-unrolling" and do all of them during the same epoch. This measure not only has benefits because of better data caching and more efficient access patterns in the low-level implementation but it also lowers the "cost of anarchy" a concept we will examine in the next Section.

After the transition to 3x multiplexed HCD we once again get the chance to re-examine the loss landscape in a richer environment that however is still trivially visualiseable due to its low dimensionality. Figures 3.58 and 3.59 show indicative plot results. We observe 3D Blobs of low loss that are often interconnected by narrow loss paths that exhibit web like structures. Some 3D plots exhibit symmetries with multiple 3D configurations resulting in similar losses due to the ill-posed nature of the problem. Subsequent frames exhibit continuities between the 3D structures a fact that makes sense given that the individual 1D landscapes that comprise the 3D one are also continuous.

3.9.4 Kinematic Chain Optimization as a Multi-Agent Game

Our generative HCD module, initially puzzled us with its effectiveness. Algorithms like PSO have a stochastic nature with randomized components that do not guarantee convergence. HCD in contrast is a deterministic algorithm which runs for a fixed number of iterations, however an important research question for us was what guarantees we can get with regards to its convergence. Turning to game theory for answers, each kinematic chain can be considered as a "player" iteratively performing small local changes (adjusting their joint angles) to improve its fitness (shared pose error), in an analogy to multi-agent game theory. Just like players in a game iteratively make moves or decisions to improve their position, kinematic chains make small, (locally optimum given their limited knowledge) changes to their parameters in each iteration to minimize the pose error. These changes are guided by the gradient information from the objective function. While each kinematic chain is optimizing its own subset of parameters, they are collectively working towards the common goal of minimizing the overall pose error. This cooperation is implicit in the fact that their actions end-up affecting the entire pose, and their collective efforts lead to the refinement of the global body pose. The dynamics of this system, where multiple players (kinematic chains) are iteratively optimizing, resemble the dynamics of multi-agent games. Like players in a game, kinematic chains are seeking local improvements in fitness (reduction in pose error). The iterative nature of their adjustments and the shared objective function leads to the convergence of the system to a local minimum of the error landscape.

The presence of a Neural Network boot-strapping optimization close to the optimum is what makes navigation and convergence easier. Since the independent encoder ensemble tends to produce sparse errors in only limited elements of the pose vector, the local search of kinematic chains becomes more efficient in maximizing fitness. The fitness landscape in this context is defined by the pose error and the parameter space of the kinematic chains. The players (kinematic chains) collectively navigate this landscape to find a configuration that minimizes the error.

In summary, the analogy of kinematic chains as players in a cooperative optimization process aligns with the principles of multi-agent game theory. The iterative adjustments and shared objective function create a dynamic system that aims to converge to a local minimum of the pose error, showcasing the synergy between "players" in achieving the optimization goal. Although there is no guarantee that the algorithm reaches the global optimum configuration, an equilibrium closer to the optimum configuration is a welcome addition especially given the limited time for computations.

"Cooperative Multi-Agent Learning: A Survey" by Liviu Panait and Sean Luke provides an overview of cooperative multi-agent learning methods, including cooperative optimization techniques offering a solid starting point for understanding cooperative optimization within the framework of multi-agent game theory and its applications in various domains.

3.9.5 The Price of Anarchy (PoA)

A fundamental concept in the study of multi-agent systems and game theory is Price of Anarchy (PoA) [439]. This concept explores the trade-off between individual agent [440] selfish actions and the overall system's performance. In our context, the individual agents can be seen as the kinematic chains, each optimizing its parameters independently and iteratively. The "selfish action" here refers to each chain's pursuit of local optimality, given that the optimization process is done sequentially.

The cost of the worst equilibrium achieved by this sequential, iterative optimization can be compared to the socially-optimal cost that could be achieved if all chains cooperated simultaneously to find a global optimum. This comparison provides insights into the efficiency of the iterative approach and its potential

deviation from the best possible solution.

The price of anarchy can be enumerated as a fraction between the cost of the worst equilibrium divided by the socially-optimal cost. Given simple graphs of joints and transition weights performing the necessary calculations is feasible, however given the very high dimensionality of our problem and the ill-posed nature of our problem with multiple 2D projections corresponding to the same 3D ground truth while also having the same perfect score made attempting to quantify this was intractable. Doing synthetic experiments against ground truth we recorded a substantial deterioration of results (indicating a high price of anarchy in our particular problem/setup), with 2D MPJPE doubling and in some cases the leftover 3D error compared to the optimum even tripling. Given that the serialized form of HCD is highly efficient thanks to the data locality and various other optimizations and that our formulation also provides ample parallelization in larger chunks of tasks as seen in Figure 3.62, we propose the use of its serial form, pipe-lined in parallel with other steps such as RGB to 2D regression and 2D to BVH regression.

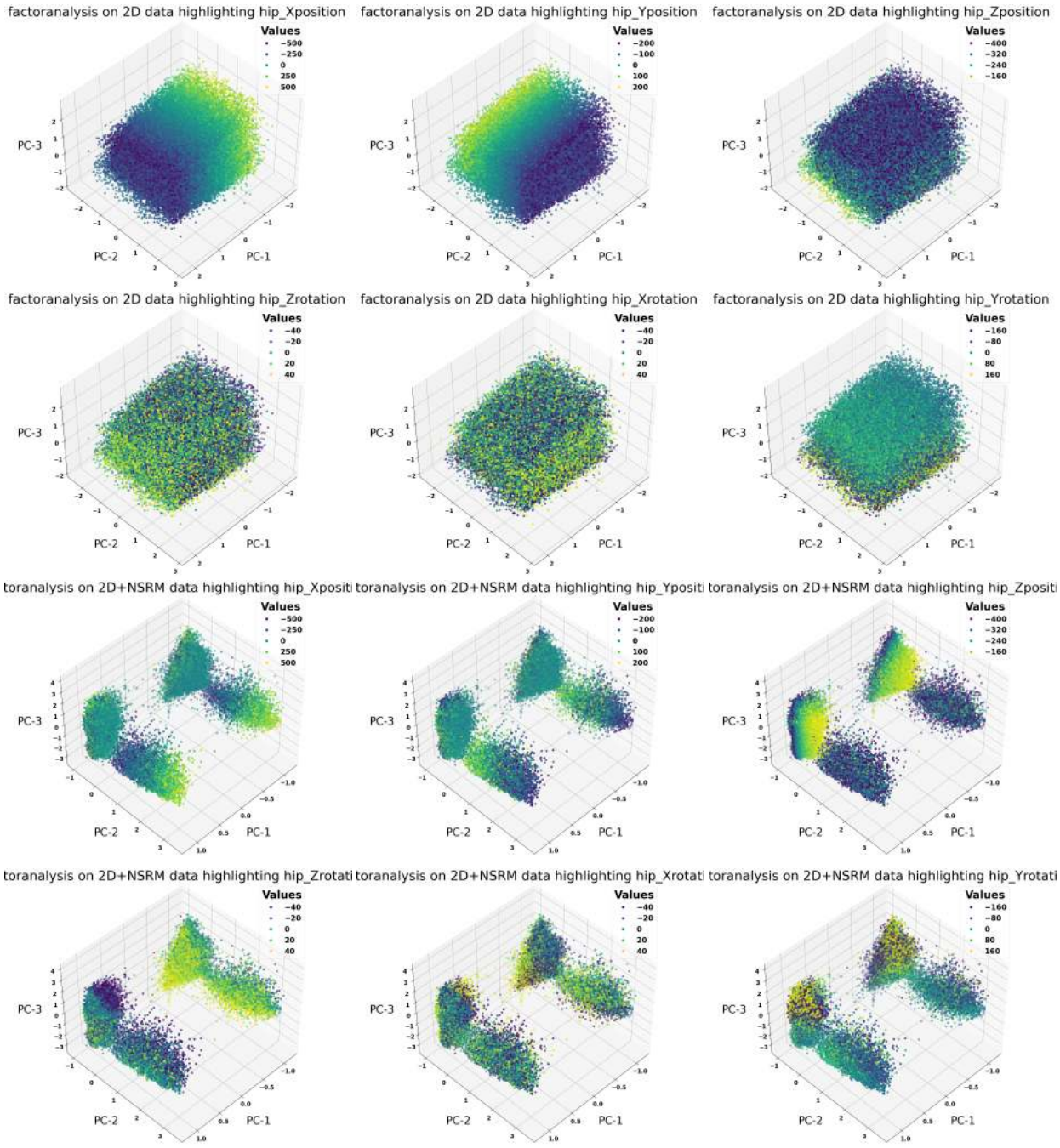


Figure 3.50: Factor Analysis clustering of samples higlighting different degree of freedom values

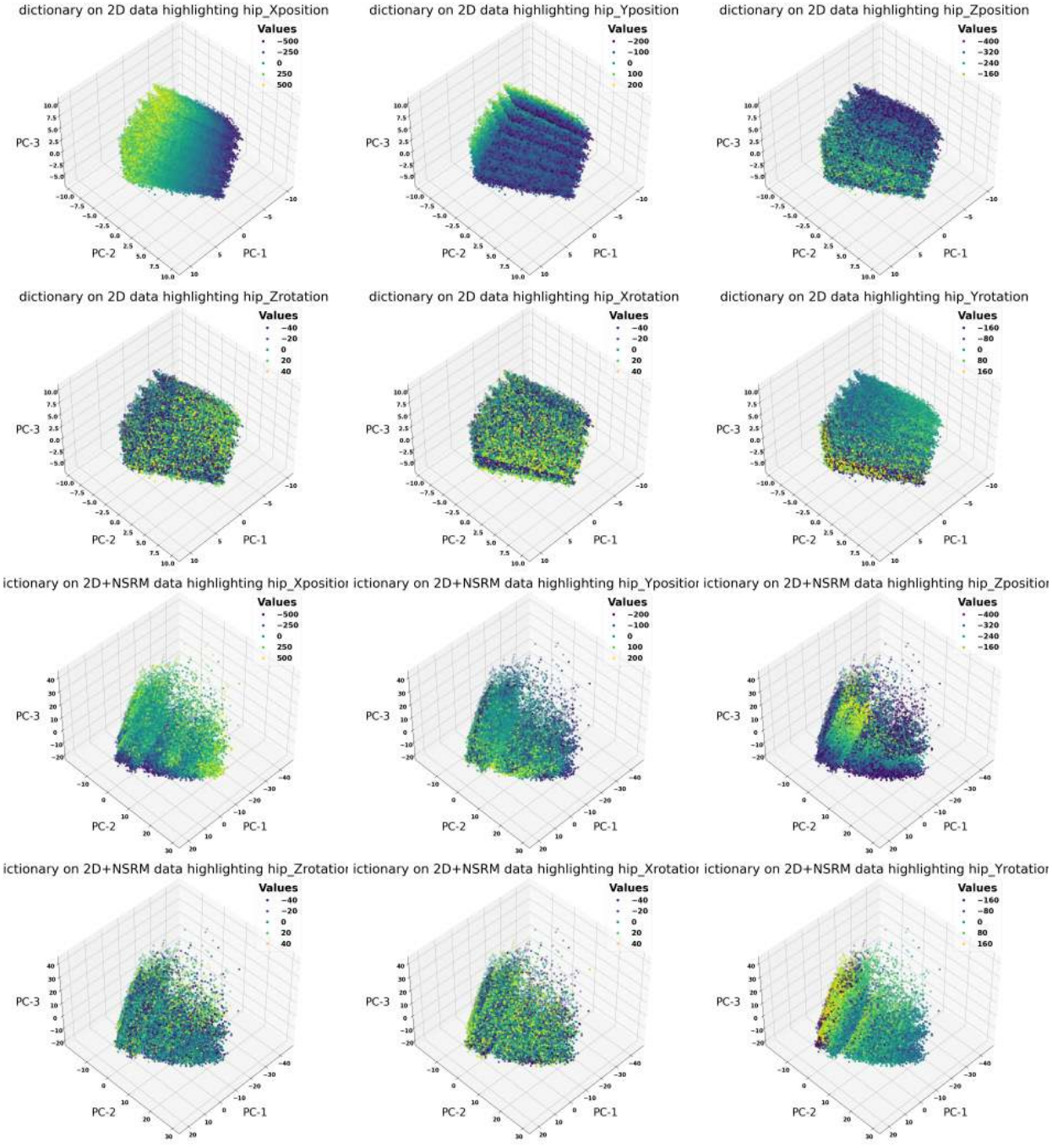


Figure 3.51: Dictionary decomposition clustering of samples higlighting different degree of freedom values

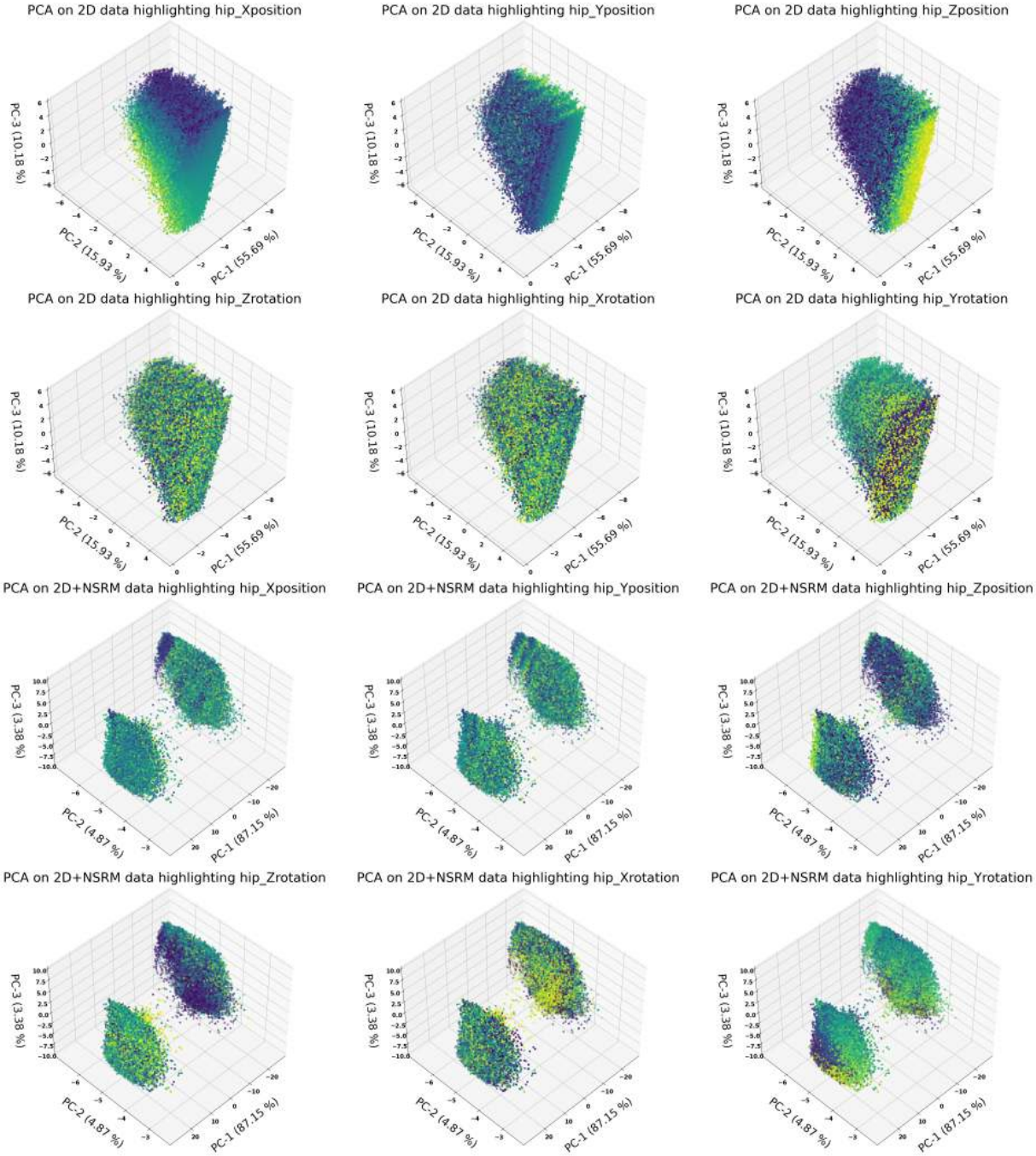


Figure 3.52: PCA decomposition clustering of samples higlighting different degree of freedom values

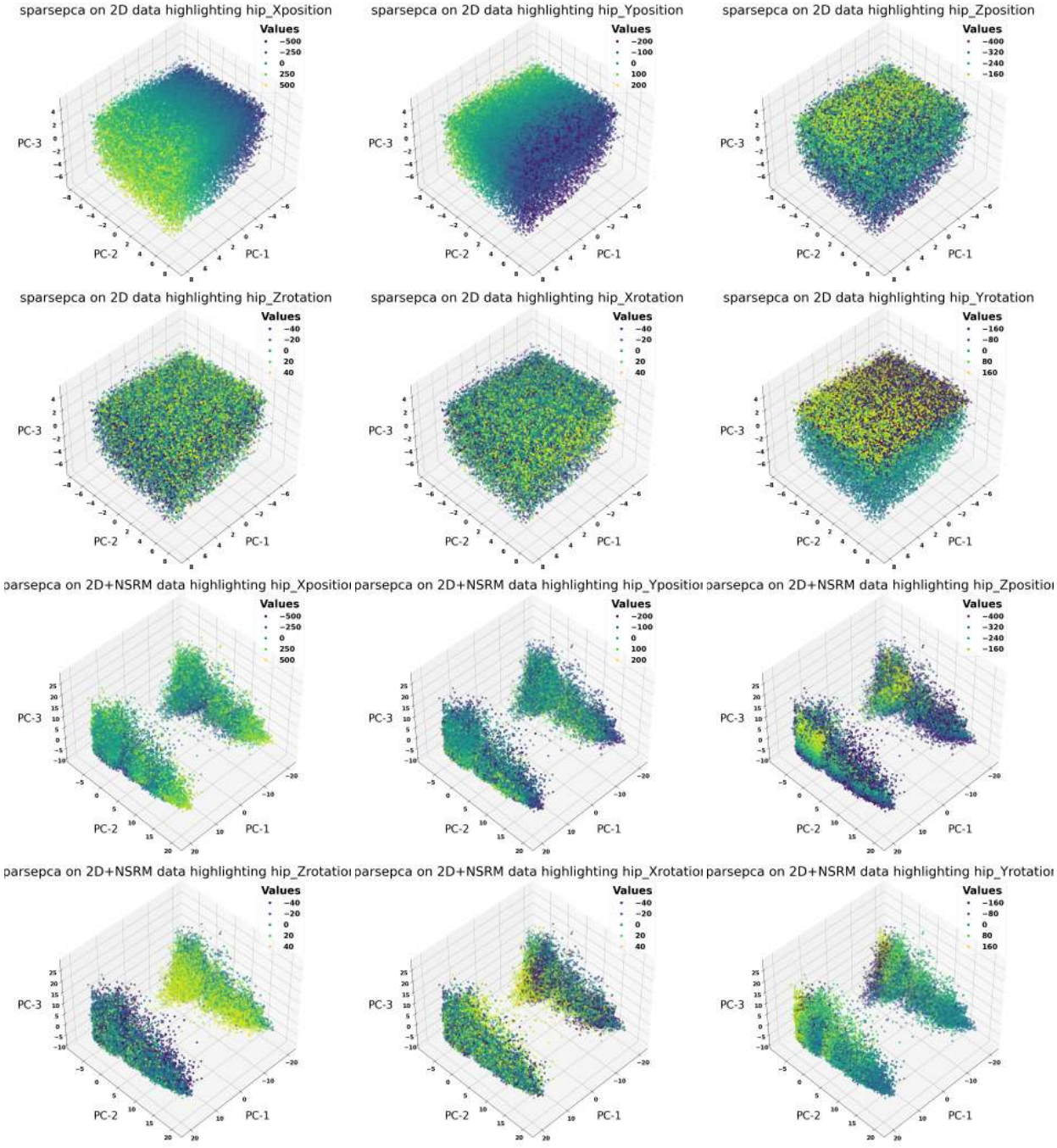


Figure 3.53: Sparse PCA decomposition clustering of samples higlighting different degree of freedom values

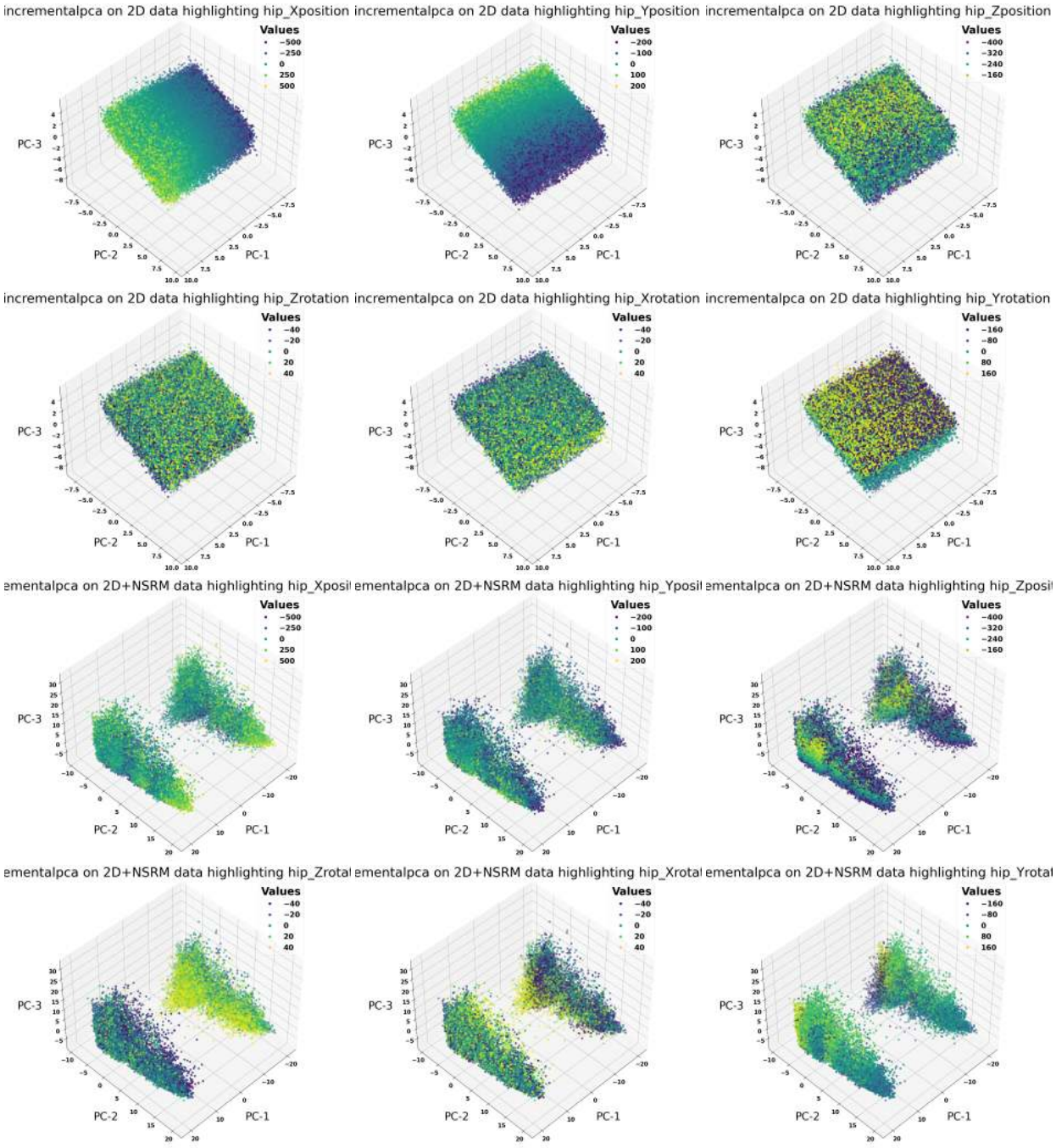


Figure 3.54: Incremental PCA decomposition clustering of samples higlighting different degree of freedom values

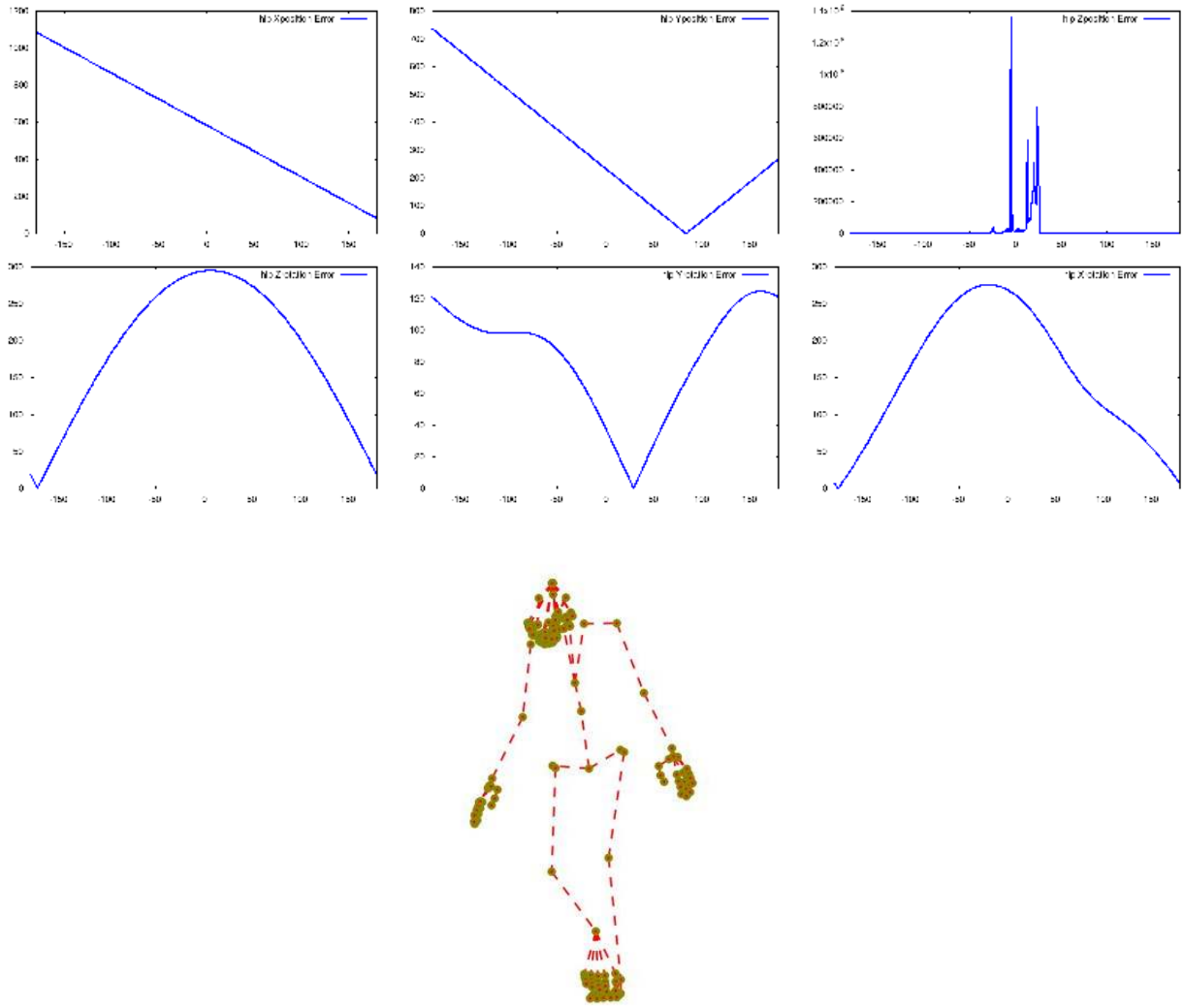


Figure 3.55: Loss fluctuation when altering the values of the X,Y,Z position and Z,Y,X rotations (First six plots) of the root bone of a BVH ground truth frame (Bottom image) and recording the recorded loss yields different landscapes for each degree of freedom. We observe that the Z position exhibits erroneous loss behavior when joints are outside(behind) of the view frustrum the camera, requiring additional checks to be filtered.

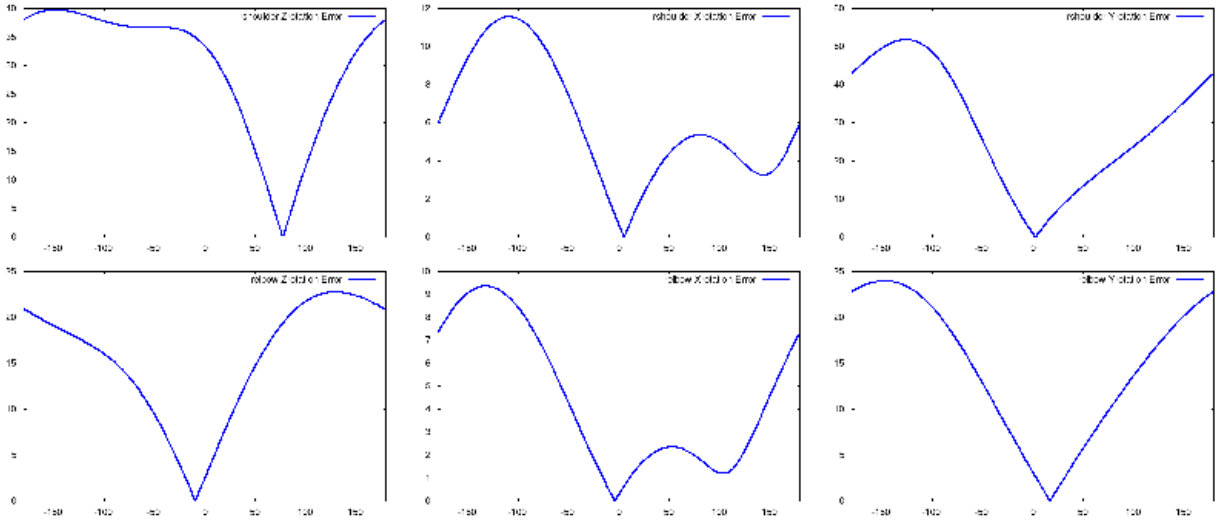


Figure 3.56: Similarly to the rotations in Figure 3.55 when plotting Z,X,Y rotations of RShoulder and LShoulder we observe complex continuous loss functions, sometimes with multiple local optima.

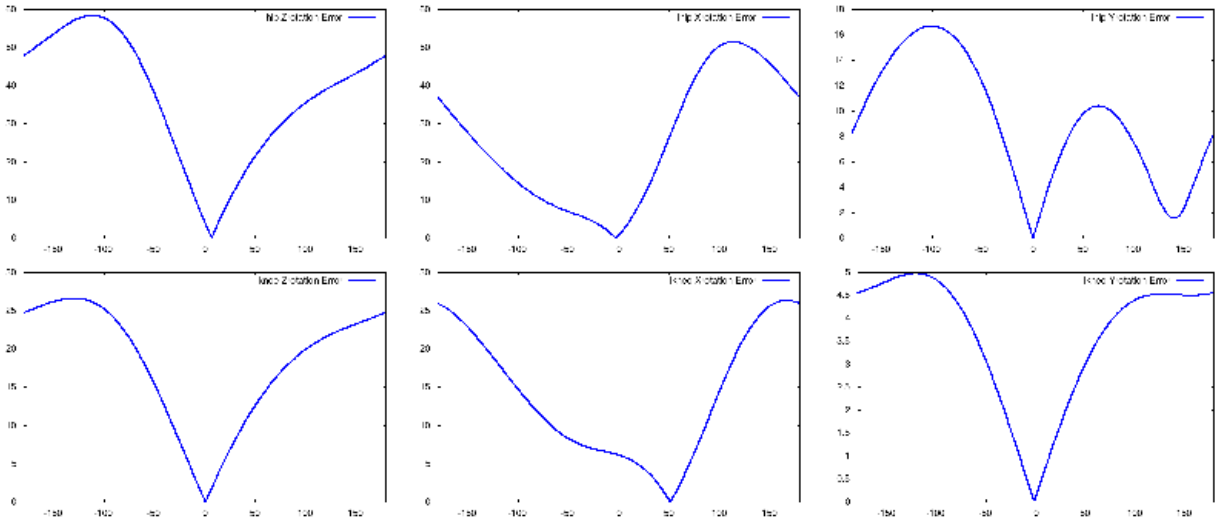


Figure 3.57: Similarly to Figures 3.55 and 3.56 when plotting Z,X,Y Rotations of LHip and LKnee we observe complex loss functions, sometimes exhibiting multiple local optima.

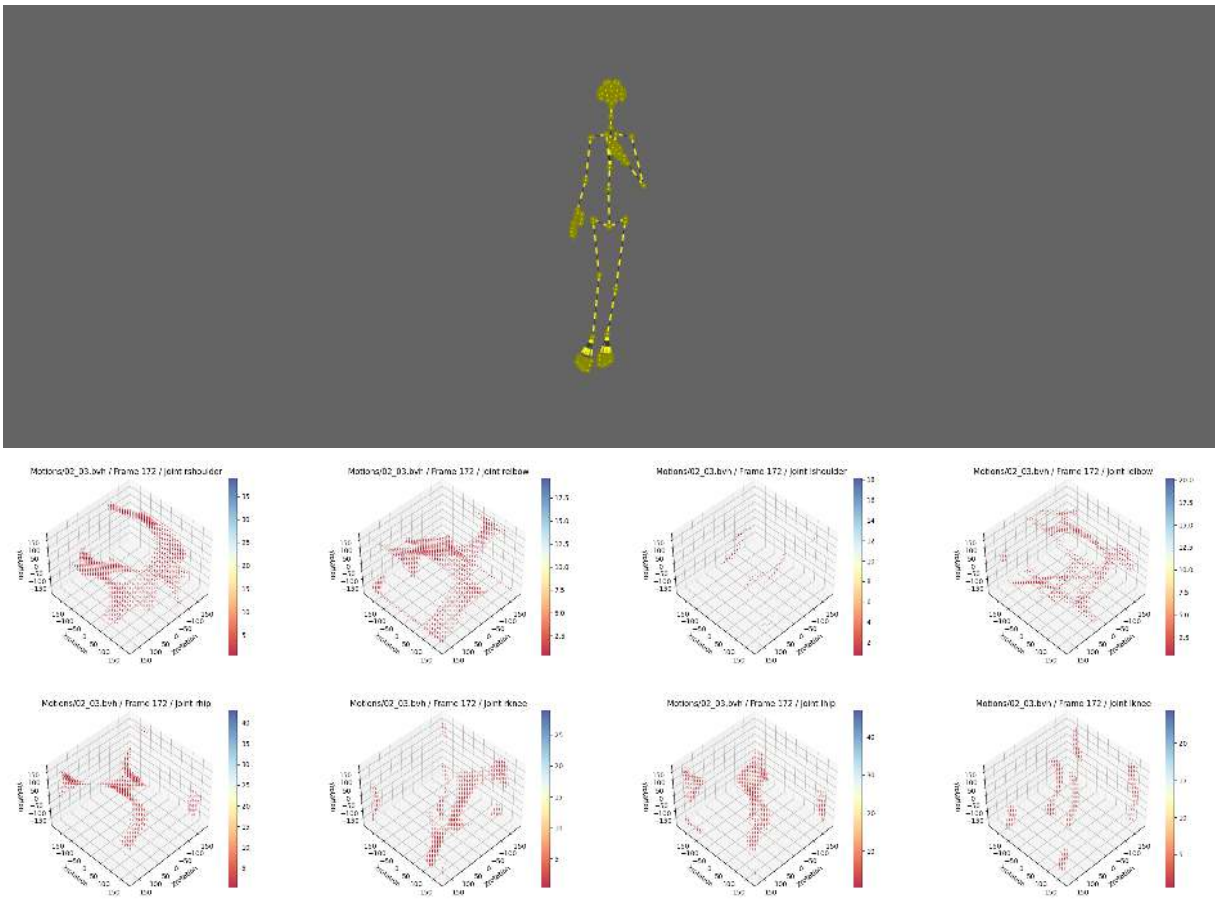


Figure 3.58: 3D loss magnitude plots after loading a BVH ground truth pose (Top) and manipulating RShoulder, RElbow, LShoulder, LElbow (first row of plots) and RHip, RKnee, LHip, LKnee (second row of plots). We observe multiple continuous clusters of values that present symmetries and wrap around the edges of the plot.

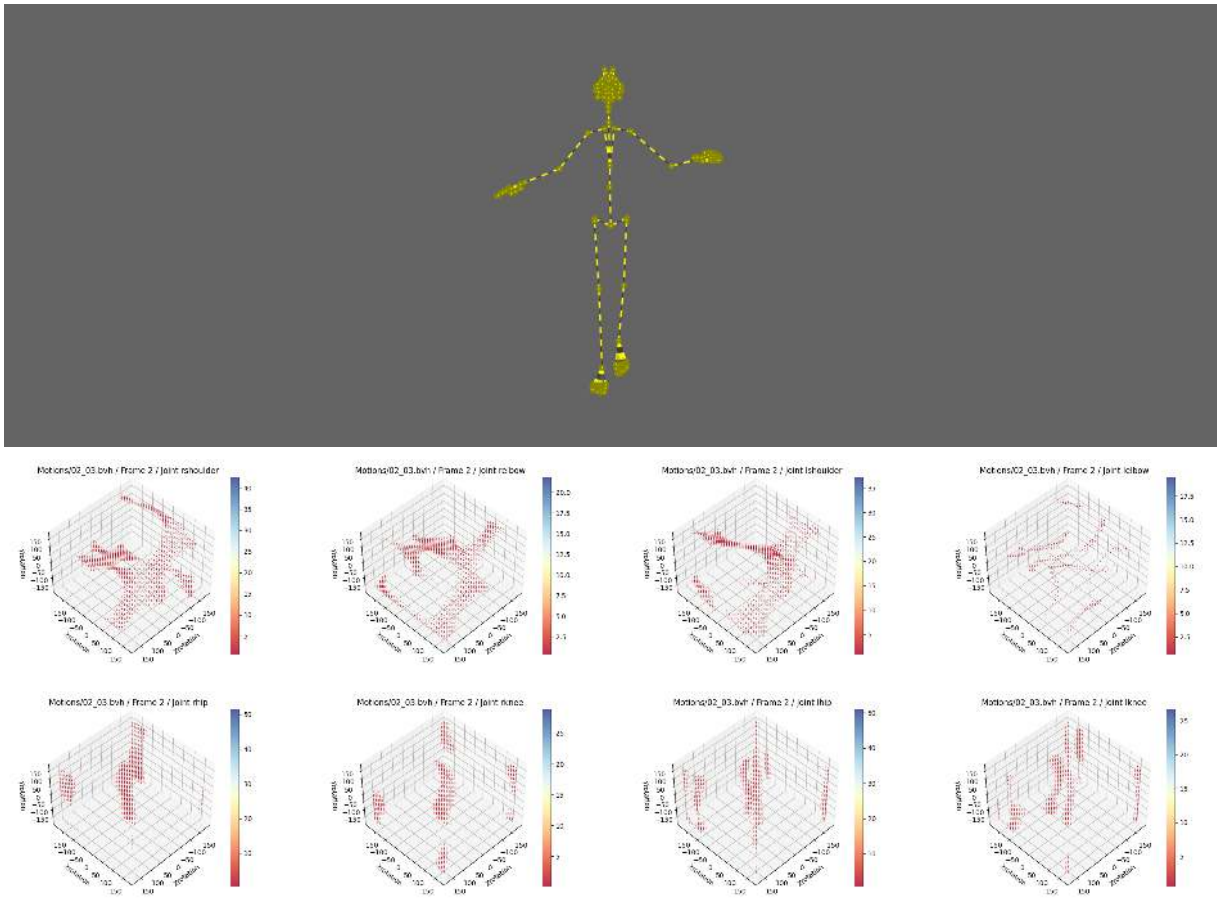


Figure 3.59: 3D loss magnitude plots after loading a BVH ground truth pose (Top) and manipulating RShoulder, RElbow, LShoulder, LElbow (first row of plots) and RHip, RKnee, LHip, LKnee (second row of plots). We observe multiple continuous clusters of values that present symmetries and wrap around the edges of the plot.

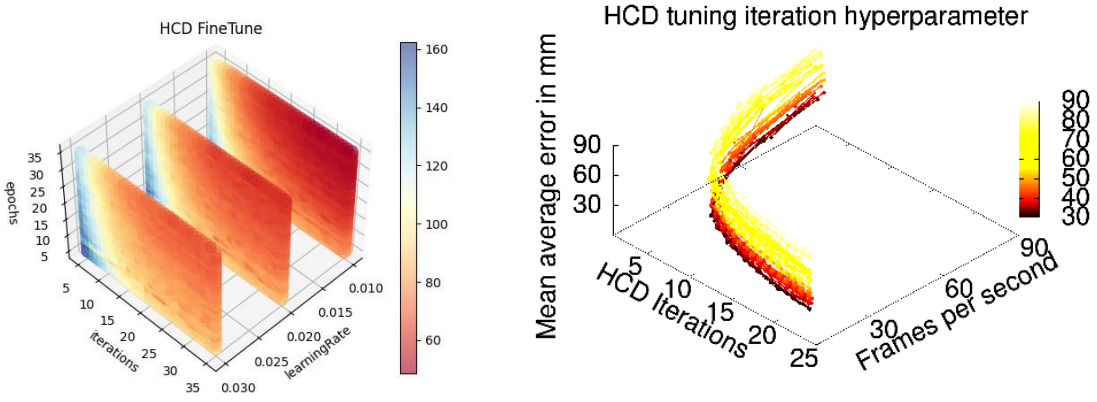


Figure 3.60: Picking a good value for HCD learning rate, epoch number and iteration number.

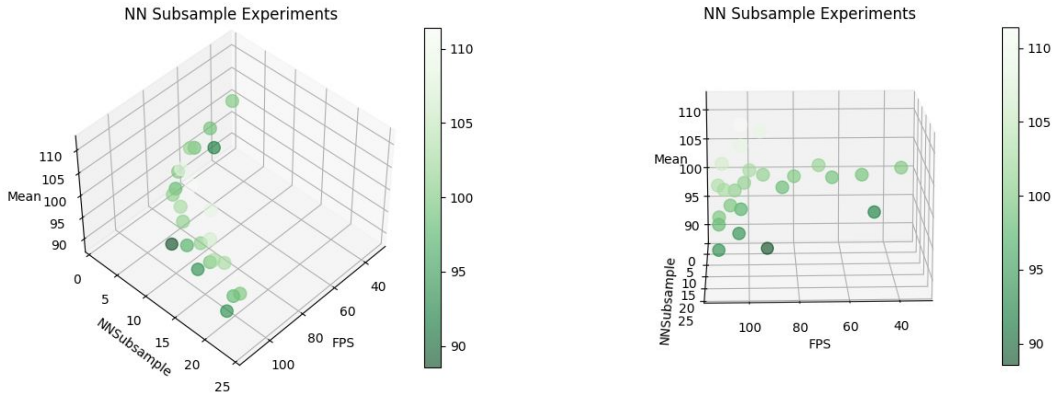


Figure 3.61: Experiments with different NN subsampling schedules. Higher subsampling rates improve framerates at the cost of accuracy.

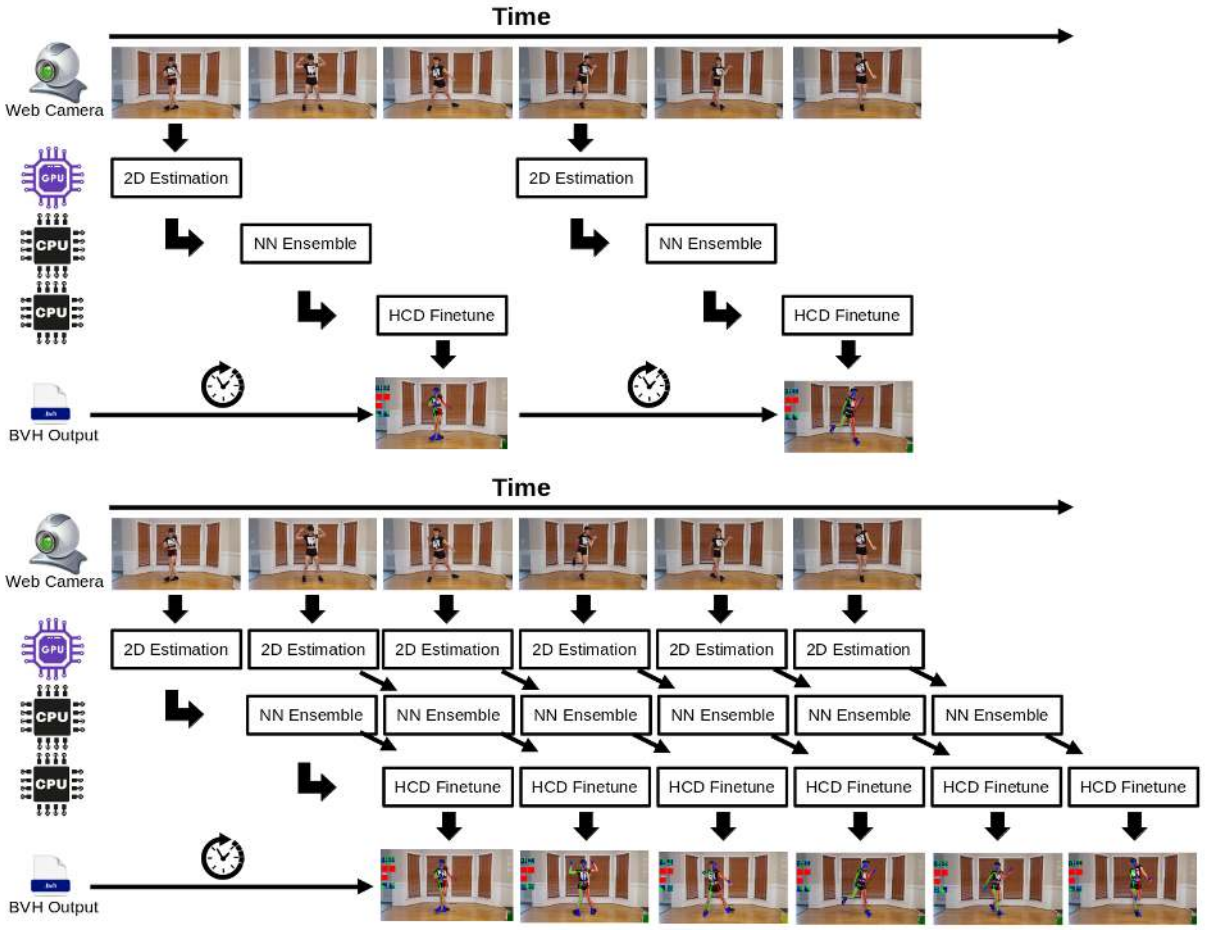


Figure 3.62: Up: Running each proposed module serially leads to resource under-utilization. Down: By taking advantage of the multi-core / multi-thread design of modern CPUs the compositional nature of our method can help pipeline steps, providing higher framerate output that improves user experience.

3.9.6 Running the Neural Network Every Few Frames

In real-time applications such as motion capture and human pose estimation, computational efficiency is a critical factor. While the neural network ensemble provides a rapid and coarse estimation of 3D human poses at an impressive rate of up to 250 frames per second in the case of [8], it's essential to consider the broader computational context in which these estimations are used.

Depending on application requirements using just the discriminative NN component we developed can produce results that exceed accuracy thresholds for use-cases that do not require a great degree of 3D pose regression accuracy. At the same time initializing the generative component from a previously known configuration may be enough in a high-frame rate scenario to perform generative-only optimization without even requiring our application to poll the neural network for a new initialization at every frame.

We call this technique NN subsampling and we experimentally use it to improve regression framerates

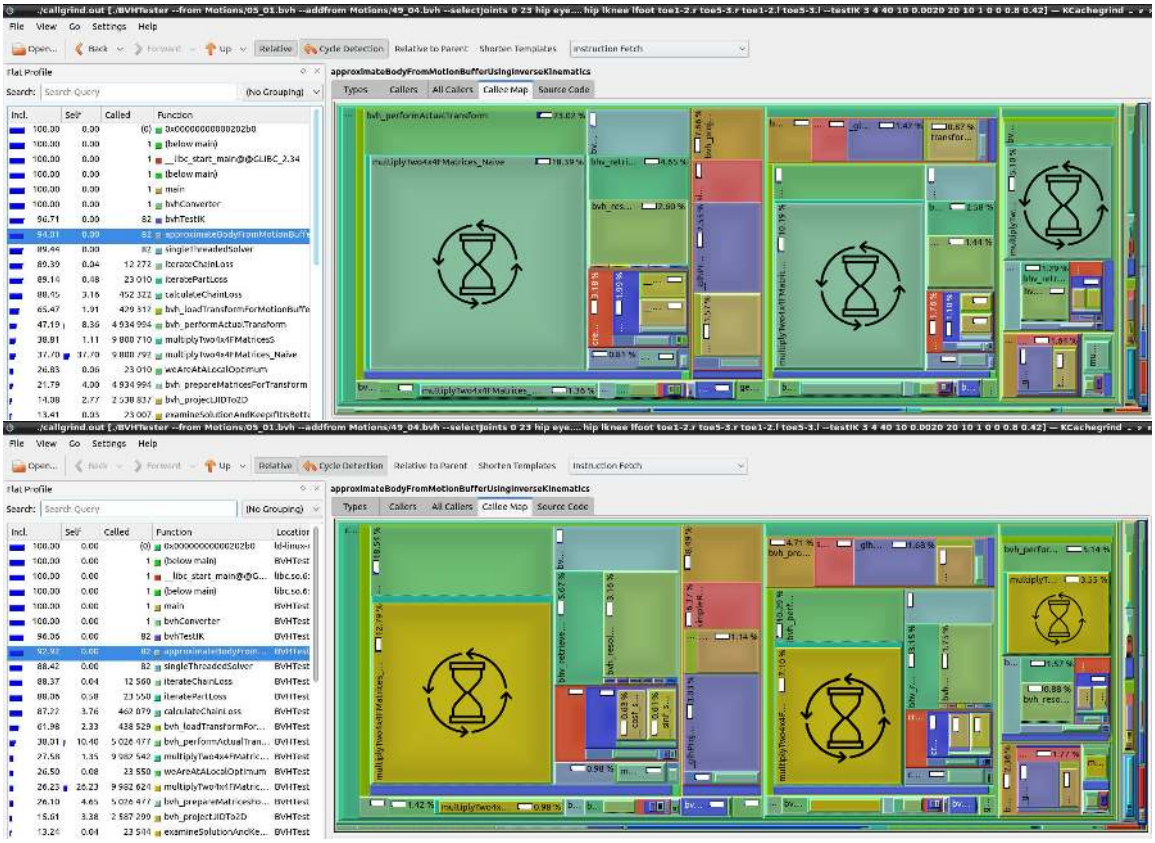


Figure 3.63: Computational time breakdown per function using the kcachegrind utility. Top: With a naïve implementation, Bottom: With an SSE2 implementation. We observe that the bulk of the time is spent on matrix multiplications which we tackle by using SIMD CPU instructions.

when the estimation projections are close to a new frame observation. We will examine such configurations in more detail in Chapter 5.

The Hierarchical Coordinate Descent (HCD) algorithm, (especially at the low epoch/iteration configurations configured to work in conjunction to the NN ensemble) is not able to perform search on the very complex loss landscape of our high-dimensional problems. After initializing it and leaving it to operate in a standalone fashion without NN "hints" when the subject moves abruptly the regressed pose gradually drifts to further and further away configurations from where it is difficult to recover. Exploiting the temporal continuity property of human motion we can improve our optimization strategy by sampling the neural network only when our estimation is further away than a pre-set threshold in terms of MJPJE or a large number of frames have passed from the last NN polling. Since human motion exhibits repetitions and our implementation allows very fast solution evaluation a last measure we can take is to keep a history of previous solutions and quickly check them against the current pose initialization for the HCD algorithm. When a pose from history is closer to the observation compared to the current NN regression it can be used instead. This final characteristic of the fine-tuning module allows improvements over multiple frames to "stack"

Listing 6 Matrix $4x4$ multiplication implementation using unoptimized C code

```
static inline void mul4x4FMatrices_Naive(float * result,
                                         const float * mA,
                                         const float * mB)
{
 //MULTIPLICATION_RESULT FIRST ROW
 result[0]=mA[0]*mB[0]+mA[1]*mB[4]+mA[2]*mB[8]+mA[3]*mB[12];
 result[1]=mA[0]*mB[1]+mA[1]*mB[5]+mA[2]*mB[9]+mA[3]*mB[13];
 result[2]=mA[0]*mB[2]+mA[1]*mB[6]+mA[2]*mB[10]+mA[3]*mB[14];
 result[3]=mA[0]*mB[3]+mA[1]*mB[7]+mA[2]*mB[11]+mA[3]*mB[15];

 //MULTIPLICATION_RESULT SECOND ROW
 result[4]=mA[4]*mB[0]+mA[5]*mB[4]+mA[6]*mB[8]+mA[7]*mB[12];
 result[5]=mA[4]*mB[1]+mA[5]*mB[5]+mA[6]*mB[9]+mA[7]*mB[13];
 result[6]=mA[4]*mB[2]+mA[5]*mB[6]+mA[6]*mB[10]+mA[7]*mB[14];
 result[7]=mA[4]*mB[3]+mA[5]*mB[7]+mA[6]*mB[11]+mA[7]*mB[15];

 //MULTIPLICATION_RESULT THIRD ROW
 result[8] =mA[8]*mB[0]+mA[9]*mB[4]+mA[10]*mB[8]+mA[11]*mB[12];
 result[9] =mA[8]*mB[1]+mA[9]*mB[5]+mA[10]*mB[9]+mA[11]*mB[13];
 result[10]=mA[8]*mB[2]+mA[9]*mB[6]+mA[10]*mB[10]+mA[11]*mB[14];
 result[11]=mA[8]*mB[3]+mA[9]*mB[7]+mA[10]*mB[11]+mA[11]*mB[15];

 //MULTIPLICATION_RESULT FOURTH ROW
 result[12]=mA[12]*mB[0]+mA[13]*mB[4]+mA[14]*mB[8]+mA[15]*mB[12];
 result[13]=mA[12]*mB[1]+mA[13]*mB[5]+mA[14]*mB[9]+mA[15]*mB[13];
 result[14]=mA[12]*mB[2]+mA[13]*mB[6]+mA[14]*mB[10]+mA[15]*mB[14];
 result[15]=mA[12]*mB[3]+mA[13]*mB[7]+mA[14]*mB[11]+mA[15]*mB[15];

 return;
}
```

Listing 7 Matrix $4x4$ multiplication implementation using C code with SSE2 intrinsics.

```
static inline void mul4x4FMatrices_SSE2(float * result,
                                        const float * mA ,
                                        const float * mB)
{
 __m128 row,row1,row2,row3,row4,brod1,brod2,brod3,brod4;
 //Float Arrays packed using __attribute__((aligned(16)))
 //Using the _mm_loadu_ps performs 4x worse..!
 row1 = _mm_load_ps(&mB[0]); row2 = _mm_load_ps(&mB[4]);
 row3 = _mm_load_ps(&mB[8]); row4 = _mm_load_ps(&mB[12]);
 //First Column ------------------------------------------------
 brod1 = _mm_set1_ps(mA[4*0+0]); brod2 = _mm_set1_ps(mA[4*0+1]);
 brod3 = _mm_set1_ps(mA[4*0+2]); brod4 = _mm_set1_ps(mA[4*0+3]);
 row   = _mm_add_ps(_mm_add_ps(_mm_mul_ps(brod1, row1),
                               _mm_mul_ps(brod2, row2)),
                    _mm_add_ps(_mm_mul_ps(brod3, row3),
                               _mm_mul_ps(brod4, row4)));
 _mm_store_ps(&result[4*0], row);
 //Second Column -----------------------------------------------
 brod1 = _mm_set1_ps(mA[4*1+0]); brod2 = _mm_set1_ps(mA[4*1+1]);
 brod3 = _mm_set1_ps(mA[4*1+2]); brod4 = _mm_set1_ps(mA[4*1+3]);
 row   = _mm_add_ps(_mm_add_ps(_mm_mul_ps(brod1, row1),
                               _mm_mul_ps(brod2, row2)),
                    _mm_add_ps(_mm_mul_ps(brod3, row3),
                               _mm_mul_ps(brod4, row4)));
 _mm_store_ps(&result[4*1], row);
 //Third Column ------------------------------------------------
 brod1 = _mm_set1_ps(mA[4*2+0]); brod2 = _mm_set1_ps(mA[4*2+1]);
 brod3 = _mm_set1_ps(mA[4*2+2]); brod4 = _mm_set1_ps(mA[4*2+3]);
 row   = _mm_add_ps(_mm_add_ps(_mm_mul_ps(brod1, row1),
                               _mm_mul_ps(brod2, row2)),
                    _mm_add_ps(_mm_mul_ps(brod3, row3),
                               _mm_mul_ps(brod4, row4)));
 _mm_store_ps(&result[4*2], row);
 //Fourth Column -----------------------------------------------
 brod1 = _mm_set1_ps(mA[4*3+0]); brod2 = _mm_set1_ps(mA[4*3+1]);
 brod3 = _mm_set1_ps(mA[4*3+2]); brod4 = _mm_set1_ps(mA[4*3+3]);
 row   = _mm_add_ps(_mm_add_ps(_mm_mul_ps(brod1, row1),
                               _mm_mul_ps(brod2, row2)),
                    _mm_add_ps(_mm_mul_ps(brod3, row3),
                               _mm_mul_ps(brod4, row4)));
 _mm_store_ps(&result[4*3], row);
}
```

Chapter 4
Explored Method Refinement Ideas

We have thus far described the core ideas of the MocapNET formulation in a detailed and structured manner giving the reader insight for our method and its various design choices. During the course of this PhD various other ideas and applications were explored towards improving the method with various degrees of success. This Chapter attempts to record and document them as extensions to the method.

4.1 Online Hard Example Mining (OHEM)

Neural Network training and by extension their training datasets are extraordinarily important for creating successful deep-learning approaches. The collection, careful processing and preparation of a large corpus of unbiased training samples makes or breaks methods since the same network architecture can perform vastly differently based on its training data.

Similarly important to the training samples is the way they are used to facilitate NN training. The Batch Gradient Descent (BGD), algorithm uses all training data at once and as a single step. The average of the gradients of all the training examples (the mean gradient) is used as the value for the update of all parameters at each step during back propagation as we saw in Section 3.5. While this is great for convex relatively smooth error manifolds this mode of operation is very costly. Although training progresses directly towards the optimum solution, unfortunately this algorithm becomes impractical as datasets become larger. As we saw in the previous Chapter MocapNET relies on millions of samples that complicate the training effort, since at every step we need the calculation of all gradients of several millions of examples.

To improve efficiency Stochastic Gradient Descent (SGD), considers just one example at a time for each step. This however, creates new problems since by considering just one example at a time the loss function fluctuates and this causes it to sometimes diverge, thus never reaching the minima.

The middle-ground between the two Gradient Descent flavours is a mixture of them that is called Mini-Batch Stochastic Gradient Descent. Neither it uses all the dataset all at once, nor it uses a single example at each time. Instead a small (mini) batch of a fixed number of training examples is selected at each update helping us to achieve the advantages of both former variants.

Mini-Batch SGD is the de facto standard algorithm used to train neural networks, and MocapNET is no exception using it with a variety of optimizer functions like RMSProp, ADAM, Adadelta and others during method development.

An important and often overlooked question however is with regards to the role of the order and selection of samples during updates and towards a good NN training session. What is the importance of a particular sample to the training result? What is the best way to select a batch of samples? What effect do

different batch sampling orders have on the training effort? Although in later stages of our work that used Sobol Sequence sampling to ensure good coverage of the high-dimensional pose space, the original work [6] relied on a biased dataset that required considerable effort [8] to be filtered as seen in Figure 3.19. Furthermore although the filtered dataset performed considerably better the filtering procedure can be definitely improved even more, so this always motivated research towards a better solution to this problem.

Studying the problem we came up with the following solution that we later cross-referenced with the literature to be similar to the "Online Hard Example Mining" (OHME).

The algorithm works with the following logic:

- After each training epoch the Mini-Batch SGD has gone over training the neural network with a randomized selection of batches, essentially covering all training samples.
- We know however that our training set is imperfect, thus some poses are over represented and some exotic poses will be seldomly included in a batch.

To remedy this situation in an automated and unsupervised fashion, we split training into regular and hard training epochs and perform the following alternating steps:

- We perform a regular training epoch
- We extract the loss of each of our training samples that when plotted normally follows a "Gaussian" distribution
- We select the right most (high-loss) samples above a fixed percentage (e.g. 30% of the dataset), creating a high-loss group.
- We train on just the high-loss group/subset of the data using mini-batch SGD for a number of hard epochs.
- We return to step 1 until reaching convergence below a loss threshold or a time limit with respect to the number of epochs executed.

Experimental results showed improvements in the range of 1°. However at the same time this method increased training times due to the expensive step of downloading samples from the GPU manually extracting all the losses extracting their statistics and selecting a subgroup. All these extra computations made the training process expensive, inflating training times and making this training method impractical both in terms of the bounded-orientation multi-ensemble paradigm of earlier versions as well as the multi region optimization for hands, eyes, mouth , upper and lowerbody.

We believe that the importance of Online Hard Example Mining (OHME) strategies could also be integrated with very recent works like "Loss of Plasticity in Deep Continual Learning" [441] that studies reinitializing a small fraction of less-used NN units after each example to maintain plasticity indefinitely. The regularization effects of different batch-sizes combined with the effect of different magnitudes of learning rates are also an important field of study that is often overlooked. Contrastive Learning [442], Incremental Learning on out of distribution data, intermediate layer optimization and emergent properties of randomly weighted neural networks are all similar topic, however despite our efforts, and identifying the importance and applicability of such approaches in our problem, in-depth studying of these topics in conjunction with the rest of work for this thesis was not feasible within the given time frame.

4.2 Probabilistic MocapNET

The two fundamental types of neural networks largely fall in the categories of regression or classification. 3D Pose-estimation methods necessarily fall into the regression paradigm since their goal is to predict a continuous numerical output that minimizes the difference between a hypothesis and the unique configuration of the observed human ground truth. This basic design choice informed the design of our networks and guided the development effort of this thesis. However upon reaching a good regression accuracy and specifically during the course of the Healthsign and Signguide projects the ill-posed nature of the 2D to 3D problem started to manifest itself in tracking results with symmetric solutions (Figure 3.13 that while exhibiting low 2D MPJPE errors were highly incorrect in terms of 3D error.

Trying to remedy this fundamental problem without altering the formulation of the method (using 2D landmarks as input and BVH angles as output), and having insight on the underlying loss landscape shown in Figures 3.55, 3.56 3.57, 3.58 and 3.59 we attempted to perform marry regression with a probabilistic approach.

By "sampling from the posterior" and using our rich ground-truth instead of each of our encoders regressing a single value, we attempted to regress a Probability Density Function (PDF). PDFs encode the likelihood of a regressed variable taking on each specific values within the valid motion range. As seen e.g. in Figure 3.57 loss does not follow a Gaussian distribution and there are multiple peaks and valleys that offer "good" and "bad" candidate solutions for a specific pose. PDFs provide a way to quantify the relative likelihood of different outcomes, while also better grounding the result with multiple outputs that force the NN to accomodate and "reply" for the whole landscape of possible solutions instead of just regressing (an often erroneous) value. PDFs have many positive qualities such as only having non-negative values that align better with the activation functions of most MLPs, their total area must be equal to 1 offering a quick way to ensure output quality when dealing with inputs outside the training distribution.

At the same some of the cons of using a PDF output is that in principle there is an infinity of values in the valid range of motion of our BVH encoder values. Directly regressing the output value requires just one output, however regressing a probability for each possible value requires as many outputs as the sampling resolution that we want to achieve. The density of output values is an important trade-off between accuracy and model complexity that is difficult to assess. During our experiments we briefly tested different configurations that produced probability estimations for values $-180°..180°$ sampled every $6°$ or $10°$ degrees with 60 or 36 outputs respectively. Unfortunately all the probabilistic experiments failed in more or less the same way, with MocapNET encoders outputting Gaussian like distributions with a single peak as seen in Figure 4.1. Various changes in model size and connectivity (even testing a convolutional variant) seemed not to affect this behavior leading to the abandonment of this idea.

Although forcing the network to produce an answer for each possible value is a much more complicated task than just regressing a single good value, and this of course means that the network will need more capacity to accommodate a harder task (which also works against the real-time performance goals set by our method) a probabilistic version of MocapNET would be an interesting endeavor especially since it would also shed light on the differences with the single value encoder regression currently used. Furthermore being able to regress PDFs could also potentially allow the HCD algorithm to be reimagined, this time being able to perform much better also using a probabilistic mode of operation allowing multiple values to be admissible as outputs and allowing better exploration of the high-dimensional pose space.

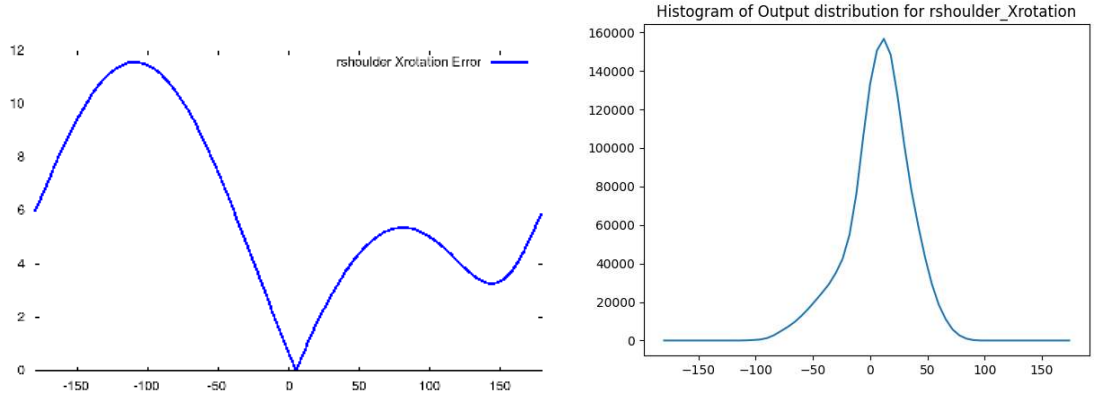


Figure 4.1: Desired behavior of probabilistic MocapNET output (left) versus what the actual probabilistic MocapNET NN we trained regresses.

4.3 EigenPoses

Computer vision research tends to sometimes resemble a fractal with important and novel ideas that resurface after many years and get re-evaluated in the light of new discoveries in the field leading to more detailed methods that exhibit the same formulation patterns. Such an important idea proposed during the 1990s was the concept of Eigenpictures by Kirby et.al. [443] and Eigenfaces by Turk et.al. [35]. Given the very complicated and high-dimensional task of face detection and recognition these methods involve the use of Principal Component Analysis (PCA) to analyze and represent images in a lower-dimensional space and subsequently represent an image as a linear combination of a small number of basis images or faces, known as Eigenpictures or Eigenfaces.

While the importance of these methods cannot be stressed enough in the context of early face recognition technologies, they gradually fell out of favor. Although Neural Networks nowadays allow for effective direct tackling of very hard and high dimensional classification tasks, the concept of Eigenfaces came to mind early during the development of the project while reading the CVPR 2019 "Monocular total capture: Posing face, body, and hands in the wild" [36] that was published shortly after our first BMVC19 publication [6]. The original version of the method was very deficient in terms of accuracy, especially when measured in the Human3.6M dataset and specifically in the sitting tasks that obviously fell out of the distribution of our CMU MOCAP [16] dataset. Browsing through the Figure 7 of [36] (also included in this thesis as Figure 4.3 to assist the reader), the observation of different MPJPE errors for different poses brought to mind the following idea.

Since these poses can be known in advance, what if we helped the regression task by immediately comparing the input eNSRM matrix with the eNSRM matrices of known poses. These "eigenposes" could create support points for the network in difficult poses thus allowing better accuracy at little cost since the task of extracting and comparing an eNSRM matrix is minimal when compared to performing the Multiply-Accumulate (MAC) and Multiply-Add (MAD) operations while doing the forward propagation step of the input through our NN ensemble. Manually selecting 6 eigenposes shown in Figure 4.4 and just adding their eNSRM similarity score as input elements unfortunately had a negligible effect on the regression accuracy.

We believe that this behavior was partly because the poses selected manually are not necessarily the

Figure 4.2: Illustration from the pioneering work of Turk et.al. [35] showcasing a part of the training set of faces and the seven eigenfaces extracted as a basis to facilitate face identification.

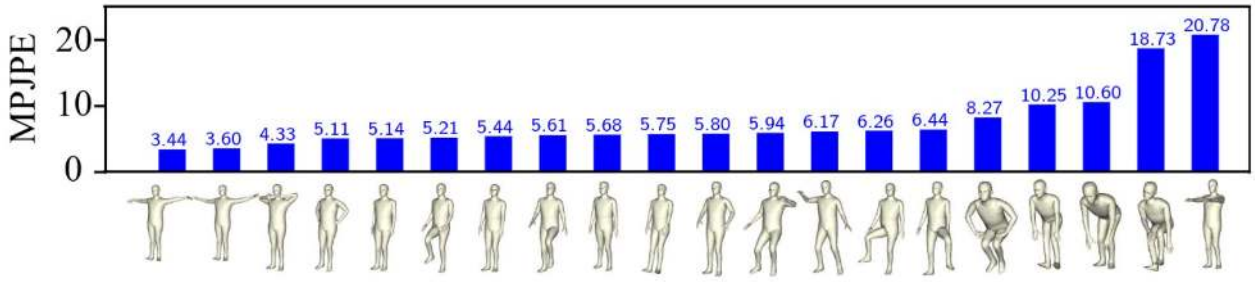


Figure 4.3: Visualization from [36] showcasing the MPJPE error for various poses. Being able to identify "difficult" poses where a network performs worse means that we can encode more representative features in an attempt to increase our accuracy.

ones that are needed, and an optimized way of deciding them (like the PCA basis analysis of eigenfaces [35]) would be needed to properly decide the number and their configuration. A second issue is the over-representation of 2D+eNSRM values which are more numerous (e.g. 323 values) compared to the 6 or 9 eigenposes (just 2% of the input vector). Given the very high degree of dropout during training it could very well be the case that they are not well correlated with the rest of the outputs making the network discard them. Although this optimization attempt was ultimately a failure we continue to believe in it and given enough time it would be an interesting aspect of the problem to study especially as a way to perform better in poses where the ensembles exhibit poor accuracy, like sitting or crouching.

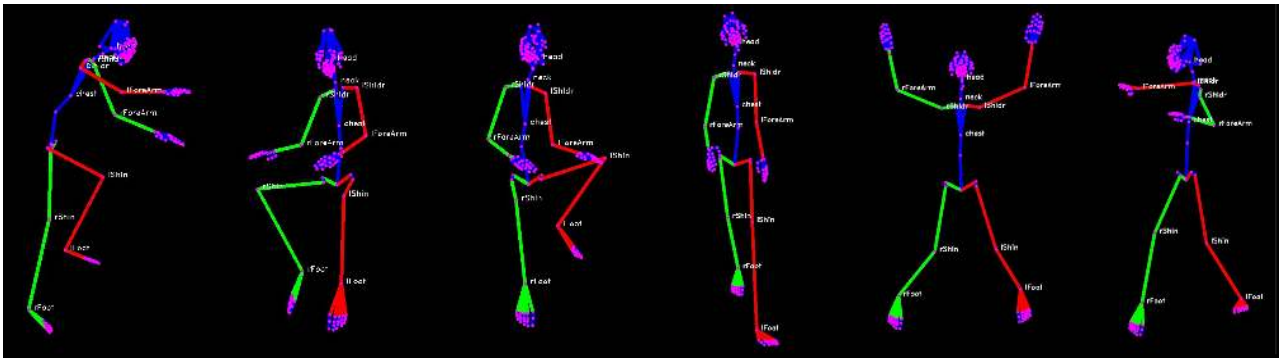


Figure 4.4: A selection of poses chosen during early eigenpose experiments.

4.4 Pose Diffusion

After the first 3 publications of the MocapNET project [6, 8, 9], and in 2022, Generative AI took the world by storm with a series of publications culminating with the release of DALL-E and Stable-Diffusion [56] that where able to successfully train conditionally text-guided de-noisers that where able to generate novel images from pure noise, and perform in-painting, style-transfer and super-resolution tasks using the same formulation. Having the good fortune of participating in the first AI summer school "Archimedes" organized by Christos Papadimitriou, Constantinos Daskalakis and Timos Sellis, and under the guidance of professor Kostas Daniilidis a Diffusion-based addendum to MocapNET was formulated based on the NCSNV2 work of Yang et.al [444, 445].

The derivation of the score-based or diffusion process can be found in detail in [37]. Although going through it in detail falls out of the scope of this thesis, Figure 4.5 offers a schematic outlining the forward (during training) and reverse (during generation) summary of SDE. Using this and with the help of Kostis Tzevelekakis we developed a fork of the NCSNV2 project [446] where we took advantage of the diffusion process in the following way.

Using the same CMU BVH dataset [16] we render and project our 3D point groundtruth in a virtual framebuffer, encoding depth values in millimeters using RGB color. We programmatically draw interpolated depth lines between skeleton joints also recording intermediate depth values along the limbs of the skeleton. We generate one image for each sample (Examples can be seen in Figure 4.6).

Once a diffusion network is trained, the way to leverage it for 3D pose estimation would be the following. Upon receiving a set of 2D observations we would render them in a framebuffer populating them with noisy values close to an initial 3D estimation. After receiving the data and going through the denoising steps, we would expect the network to "denoise" the image in a way consistent with the training dataset thus recovering the "depth map" of the skeleton and thus allowing direct depth sampling at the 2D landmarks of the skeleton, thus enabling 2D to 3D pose estimation.

The generated RGB image dataset is then processed with a modified version of NCSNV2 and is left to train for 300000 steps while we monitor the training effort. Due to NCSNV2 having been tested in different resolution sizes (FFHQ 256px, LSUN bedroom 128px, LSUN tower 128px, LSUN church outdoor 96px, and CelebA 64px.) each of which require specific tuning, we begun training with a very conservative training image size of 32x32 pixels. Although training was stable and relatively not demanding in terms of hardware requirements in this low resolution, it quickly became obvious that this low resolution would not be helpful

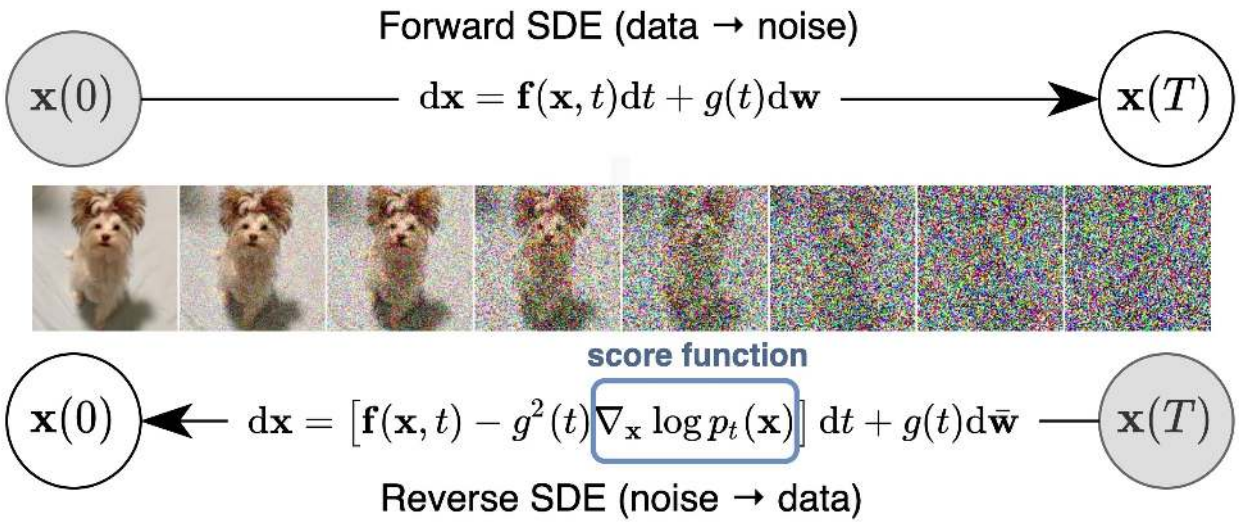


Figure 4.5: Forward and backward SDE outline for score-based generative models. Figure from [37].

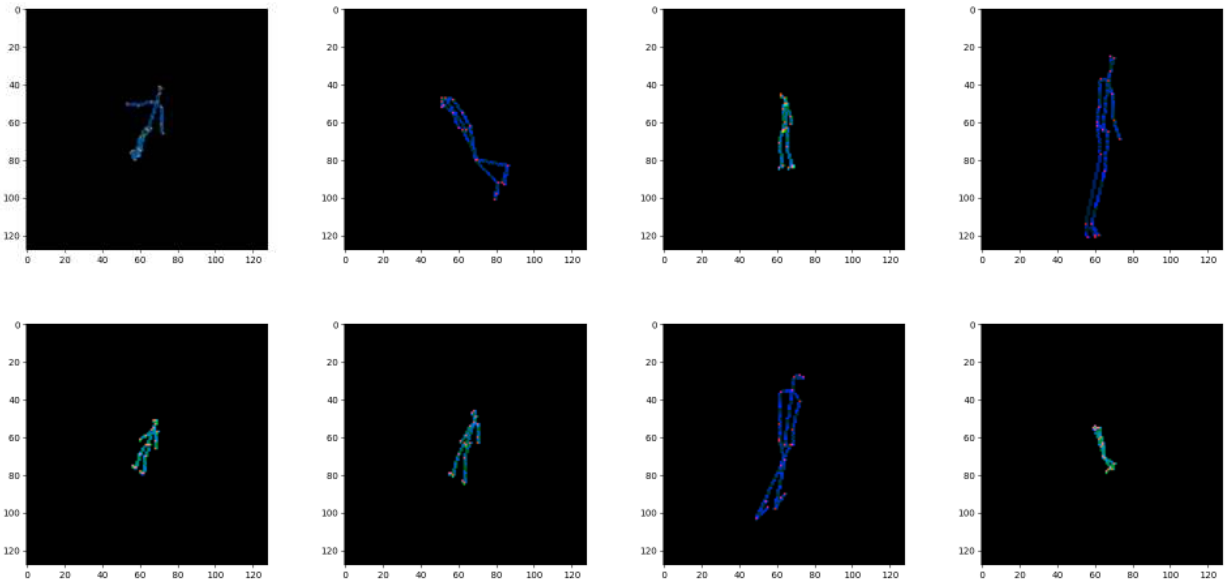


Figure 4.6: Generating score based generative NN training samples from our BVH dataset at a 128x128 resolution.

in achieving any sort of usable accuracy by the resulting network. Scaling up resolution to 64x64 improved the situation while also maintaining stability and the good training properties of the 32x32 configuration.

Moving up to 128x128 resolution, however, the task hardware complexity sharply increased, while we encountered many problems in terms of training stability, as seen in the early steps in Figure4.9. Many

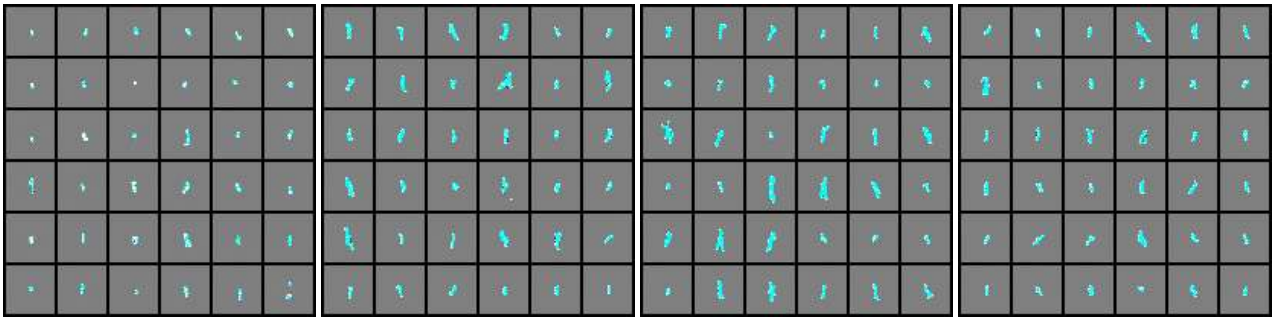


Figure 4.7: Experiments using a 32x32 tile size for 10K, 45K, 234K and 300K steps.

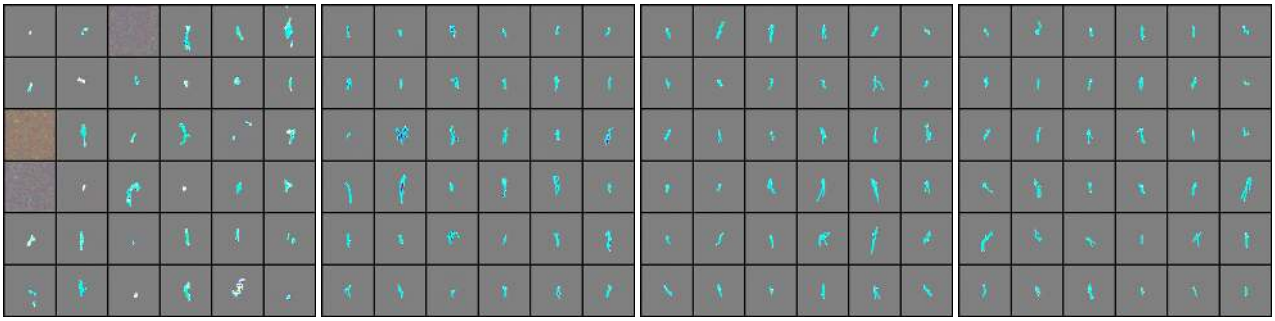


Figure 4.8: Experiments using a 64x64 tile size for 10K, 45K, 234K and 300K steps.

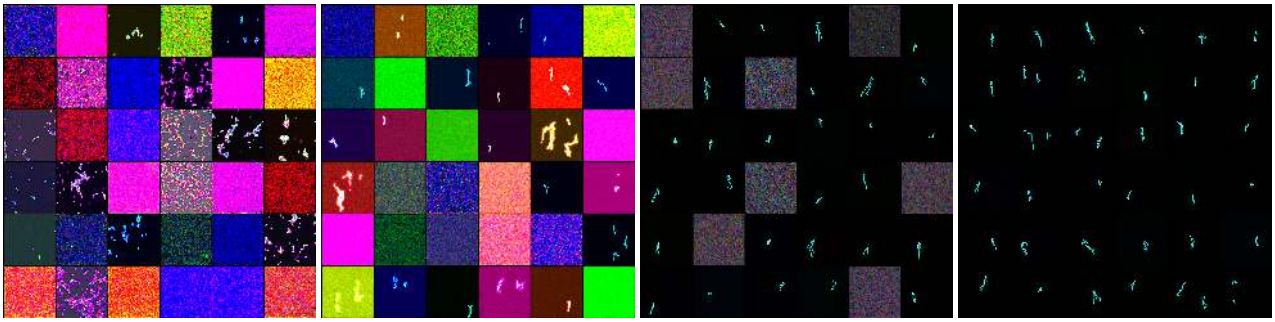


Figure 4.9: Experiments using a 128x128 tile size for 10K, 45K, 234K and 300K steps.

attempts on training in this resolution failed while after many attempts we were able to successfully perform training after switching the background to black instead of the gray color seen in previous attempts. Due to the diffusion process already being very heavy and prohibitive in terms of real-time operation we decided to further increase resolution in an effort to target an even clearer and better process focused on accuracy.

The maximum size we were able to successfully train was using images of 156x156 (Figure 4.10) size, with 200x200 (Figure 4.11) being the resolution limit at which our training code and tuning could no longer tackle the process. Although the exact mechanism that governed this large resolution failure were not known to us, they probably had to do with the relatively small training dataset of a few million samples and the very large surface of the image that most of the time remained empty. As seen in Figure 3.19 the dataset is biased towards specific poses and areas, something which might negatively affect a high resolution diffusion

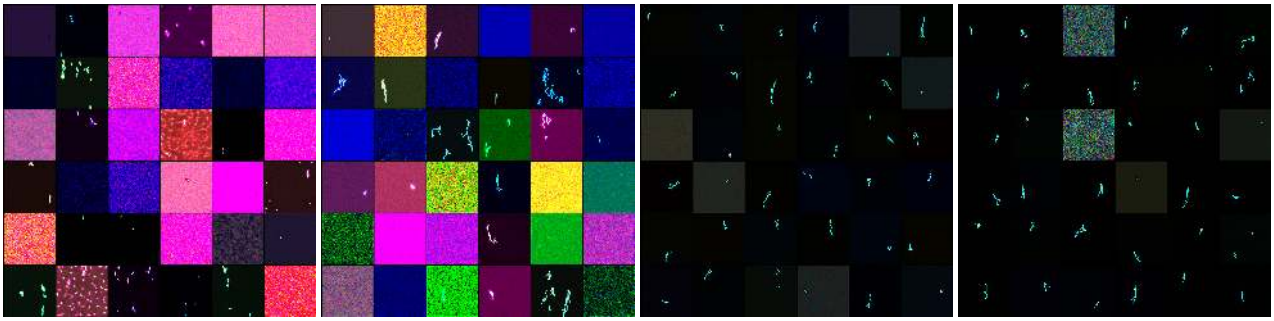


Figure 4.10: Experiments using a 156x156 tile size for 10K, 45K, 234K and 300K steps.

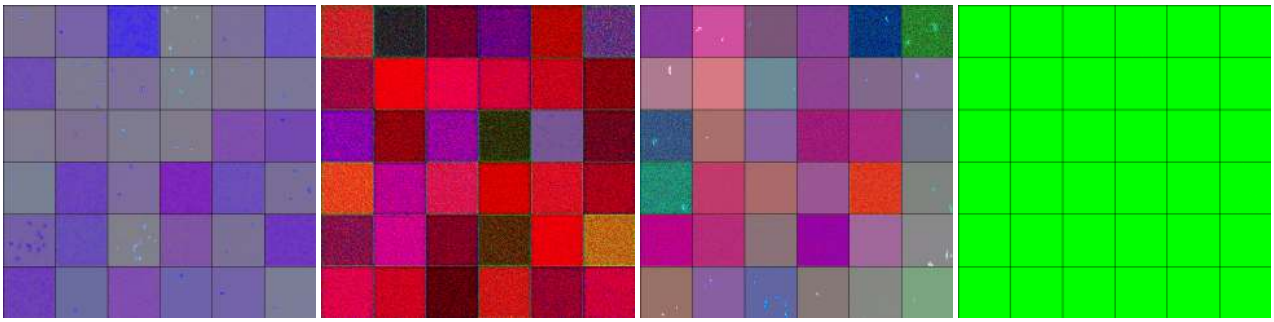


Figure 4.11: Failed experiment using a 200x200 tile size for 5K, 10K, 15K and 20K steps.

process.

At this point and having already dedicated considerable effort towards this research direction we further conducted a thought experiment regarding the maximum theoretical accuracy that we would be able to acquire by this method. These experiments involved getting the ground truth 2D and 3D points, converting them into an image (Figure 4.6) and then the recovery step comparing the recovered values to the ground truth. As it is easily understandable low resolution images (e.g. using a 16x16 frame) do not have enough capacity to record all points that lead to a very high degree of error, unsuitable for 3D human pose estimation. Extensive results of this experiment can be seen in Table 4.1.

Even configurations with 64x64 pixels seem not to be able to correctly reflect important landmarks such as shoulders and neck with accuracies that are much lower than our much faster and versatile MocapNETs as we will see in the Experiments Chapter. Furthermore state of the art results like MotionBERT [199] achieving 16.9mm depth regression scores in the Human3.6M dataset [39] in a monocular setup shows us that a resolution of 256x256 would be required for equivalent accuracy, however we were not able to train even at the substantially lower resolution of 200x200.

Some final attempts at the project included a “digitization” of the pose into image patches as seen in Figure 4.12, however this drastically departed from the original idea being much closer to the distance matrix regression idea of [172], albeit in a much more complicated and computationally heavy formulation. These discouraging results that went counter to the rest of this thesis goals for real-time 3D total capture where also combined with complications due to other work requirements and the departure of Kostis Tzevelekakis from the joint effort due to his MSc. thesis completion and subsequent transfer to ETH Zurich. All these issues led to the stagnation of this research direction leading to its abandonment. It is worth noting however

Joint Name	Mean Average Error for specific resolution					
	16x16	32x32	64x64	128x128	156x156	256x256
Eye.l	82.46	98.38	88.28	24.30	37.36	14.28
Eye.r	82.36	98.34	89.62	48.85	39.95	15.85
Toe1-2.l	205.80	134.41	78.78	51.66	26.11	8.51
Toe1-2.r	206.77	134.92	78.34	36.87	24.70	7.69
Toe5-3.l	201.80	126.78	71.10	35.52	24.78	8.96
Toe5-3.r	202.24	124.80	69.88	33.41	26.44	10.12
Head	66.59	64.81	35.79	34.69	7.47	2.43
Hip	242.54	193.72	78.28	11.79	15.88	5.18
LElbow	180.62	182.58	60.20	23.51	7.64	3.09
LFoot	187.97	88.20	32.24	11.99	6.25	2.52
LHand	200.24	180.84	76.94	9.32	16.80	6.75
LHip	240.67	201.66	91.32	32.74	23.93	10.11
LKnee	235.43	147.04	37.58	8.35	5.42	2.33
LShoulder	110.65	171.29	157.15	50.76	30.41	8.59
Neck	86.44	127.33	134.93	42.71	22.83	5.98
RElbow	180.83	180.59	60.32	12.21	7.81	3.12
RFoot	187.82	86.93	31.42	9.16	6.12	2.49
RHand	198.33	177.32	76.13	24.40	16.81	6.72
RHip	241.95	203.27	92.54	32.94	23.88	10.08
RKnee	236.26	148.07	37.93	8.42	5.47	2.34
RShoulder	113.12	173.85	156.02	49.83	29.98	8.54

Table 4.1: Summary of theoretical accuracy achievable with different resolution diffusion

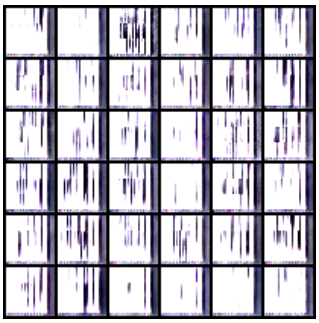


Figure 4.12: Experiments using a 32x32 tile size using “digital” encoding.

that during 2023 there has been a multitude of methods using Diffusion-based formulations for pose estimation, like [447], [448] and [449] showcasing the research interest in this direction, and given new discoveries that could reduce the complexity of such methods (or better hardware acceleration for such tasks [450]) warrants a re-evaluation of this topic.

Chapter 5
Experimental Evaluation

Experimental evaluation for 3D pose estimation is an important consideration that largely dictates the success or failure of methods. Although methods may exhibit a variety of properties and other interesting and novel ideas successfully publishing a method in a major computer vision conference typically entails surpassing the state-of-the-art in some of the available method benchmarks. Although this work attempts bridging the gaps of multiple pose estimation sub-problems with a common solution, until recently benchmark datasets were also divided, with different benchmarks for body, hands, faces and gaze. While this makes sense for historical reasons it requires a substantial effort in the context of a 3D total capture method in order to properly parse all of the different data, formats, perform the necessary conversions and conduct fair experiments.

An important issue that is often overlooked (often not being mentioned at all at publications) is the importance of computational performance. Most methods tend to employ incredibly complicated architectures that while performing better than their counterparts in reality are very impractical for use. Our work stands in stark contrast to methods that ignore computational cost. This is evident from our relatively stable (even relatively decreasing) model size (Figure 3.39). Although offline methods achieve incredible accuracy, we believe that 3D human pose estimation systems design will eventually prioritize computational efficiency. The main reason for that is that energy consumption plays a key role both in terms of battery life in most modern mobile computing devices, but also in data-centers that require substantial amounts of energy to train and run the pose estimation algorithms.

Another important issue is that due to the unique design of our method that regresses BVH to 2D, unfortunately since we are the first to attempt this in a Total Capture scenario there are no datasets that test a similar use-case and thus we need to measure our accuracy using proxies like 3D or even 2D (in the case of the face) reprojections in order to systematically assess our performance. Although these metrics still reflect output quality methods that directly regress depth (and thus having a loss explicitly tied to their target goal) will architecturally score better than methods like MocapNET where the output is not just a depth estimation but needs to comply with armature dimensions (that always maintains the same limb distances) and where even a relatively small regression error of 3° on a single degree of freedom will stack on all consecutive joints accumulating over time and preventing us to achieve 3D accuracy that can compete with the state of the art.

Finally another often overlooked issue is avoiding data leakage by ensuring that the test dataset remains completely isolated from model training and hyperparameter tuning. Due to our method regression output being higher-level than 3D points in our case the separation of Training, Validation and Test Datasets was always ensured with Test datasets measurements only being performed shortly before publications. During

the development of the MocapNET modules only Training and Validation sets were used, while in many experiments (like the case of Hands and Faces) due to not having an actual real groundtruth dataset to work with, and to maximize the capacity of our hardware to fit as many training samples as possible, we also worked with completely synthetically generated data.

Having provided a rough overview of some of the unique considerations of our experimental evaluation we will begin to examine it on a per-task basis that closely tracks the publication history of our method. Since the incremental versions of the method gained gradual deeper networks with a higher degree of connectivity. We mark the subsequent versions as **MocapNET 1**, for the BMVC2019 implementation [6], **MocapNET 2** for the ICPR2020 implementation [8], **MocapNET 3** for the BMVC2021 version [9], **MocapNET 3.5** for the PCA compressed version (Extensively analysed in Section 3.8) of our method [20] and **MocapNET 4** for the final version of the method that also accommodates 3D Head Pose Estimation, 3D Gaze and Facial Capture [31].

5.1 Body Pose Estimation

As shown in Table 2.6 is the de-facto standard for 3D human pose estimation benchmarking that allows comparison with the largest number of methods is Human3.6M [38]. Using it we are able to establishing a comparison with a wide variety of monocular and multi camera methods. To maintain a richer comparison basis we also create a validation split for datasets 01-10 CMU BVH [16]. We include results from all the intermediate stages of the project as an ablation study. For the final accuracy results of the resulting **MocapNET 4** method see Table 5.10. M.A.E. refers to mean absolute error for regressed joints after Procrustes Analysis. Serial is the serial number of the experiment. Intermediate missing serial numbers are ones that where dedicated to training hand,mouth,reye,facial ensembles (since all of them where updated in a round-robin way). Column labeled PCA refers to using PCA as an input dimensionality decomposition. Tr.Samples refers to the number of training samples used. Lambda is the scaling value of the NN controlling its width. Tr.Model Size refers to the trained number of parameters it should be interpreted taking into consideration the number of Enc.Outputs (Encoded outputs by each encoder). For 30 Enc.Outputs Tr.Model Size is larger since all of the network is trained at once. Depth refers to the NN layer depth. Loss scale is a multiplier for each sample loss (that otherwise is in the range of [0..1]). Loss Exp. refers to the loss exponent. A value of 1 means MAE, a value of 2 means MSE, a value of 4 means MQE. Batch size records the used batch size during training. Cont. is a shorthand for Continuous Training and controls the inheritance of trained networks across the hierarchical changes (i.e. not starting from a random set of weights for each encoder but inheriting the previous one). EDM can also append a Euclidean Distance Matrix after the eNSRM to provide even more scale features. Dropout controls the dropout amount for NN layers. Please note that these results have to only do with the MocapNET 4 neural network and do not use the Hierachical Coordinate Descent (HCD) refinement module.

5.1.1 MocapNET 1

We evaluate MocapNET using the Human 3.6M (H36M) [39] dataset which it is the de facto standard [98, 171, 173, 186, 188, 191, 192, 200, 201, 204, 210, 217, 223, 225, 451–453]. Evaluation is performed using the mean per joint estimation error (**MPJPE**) for all joints after Procrustes alignment [42] of a method's estimation to the ground truth. The datasets used for training and evaluation are defined through clearly specified protocols. Protocol 1 dictates training on subjects 1, 4, 6, 7, 8 and performing tests on subjects 9, 11 on 2D

Figure 5.1: Snapshots from the Human 3.6M dataset [38] depicting various subjects performing different actions on a VICON MOCAP system.

MocapNET λ = 1, Trained on CMU, tested using H36M Blind Protocol 1																
Input	Dir	Dis	Eat	Gre	Pho	Pos	Pur	Sit	Smo	Pho	Wai	Wal	Dog	WaT	Sit.	Avg
GT	135	140	145	143	153	137	174	215	1565	150	151	156	166	134	246	**160**
GT + $\mathcal{N}$(0,5)	141	147	150	148	158	141	178	219	161	155	156	159	169	139	248	164
GT + $\mathcal{N}$(0,10)	154	159	162	160	169	152	188	228	174	167	168	170	180	154	256	176
GT + $\mathcal{N}$(0,15)	172	178	180	178	186	170	202	241	190	183	186	187	194	174	267	191
GT + $\mathcal{N}$(0,20)	195	199	198	199	204	189	218	256	208	201	204	207	213	197	280	211

Table 5.1: Results of the original **MocapNET 1** for $\lambda = 1.0$ trained on CMU data and tested on Human3.6M using Blind Protocol 1. All numbers are MPJPE in millimeters. We test using Ground Truth (GT) plus different settings of gaussian pixel noise $\mathcal{N}(\mu,\sigma^2)$ with mean μ and variance σ^2. The average error for Protocol 1 is marked with bold.

points originating from all available cameras. Since we do not train on any data from H36M but otherwise adhere to this protocol, we label it *Blind P1 (BP1)*. Protocol 2 uses the same training and test sets as P1 but only on the frontal camera. We again perform experiments without training on H36M, so we label this as *Blind P2 (BP2)* protocol. There are inconsistencies in the literature about the alignment performed during experiments. For example in [171], P1 uses Procrustes alignment and P2 root alignment, whereas in [175], P1 is root and P2 Procrustes. We perform both P1 and P2 using Procrustes alignment.

Tables 5.7 and 5.2 reveal medium accuracy across all actions from all viewing angles with the exception of the two sitting datasets where the topology of the body is more challenging and not well represented in the CMU training examples. This is contrary to other methods which have larger relative fluctuations across actions with the most accurate typically being the walk action. Another interesting result is that on BP2 (Table 5.2) which reflects frontal view accuracy, and thus an easier case, MocapNET achieves the same average error as in BP1. We attribute this to the employed class structure that effectively decouples orientations. An important result is that the method performs well with noisy input (Fig 5.2), with an average impact of less than 2cm for a relatively large noise margin (σ = 10 pixels).

Our network has never seen people performing many of the H36M actions, nor received any input using similar camera intrinsics, angles etc. Experiments from Yasin et al's Dual-Source 3D Human Estimation [453] hint at accuracy penalties of $\approx$ 30mm for our case. The limitations of [18] are very well documented and also one of the main premises of [302] that adds a greater variety of poses with a MOCAP suit. Comparison to other methods is summarized in Table 5.3. MocapNET is not as accurate as the state of the art but this is natural as it deals with rotation regression and not 3D position regression. Even small errors in each of the joint angles quickly stacks across the kinematic chain with a negative impact on accuracy.

MocapNET λ = 1, Trained on CMU, tested using H36M Blind Protocol 2																
Input	Dir	Dis	Eat	Gre	Pho	Pos	Pur	Sit	Smo	Pho	Wai	Wal	Dog	WaT	Sit	Avg.
GT	138	145	142	144	151	146	165	194	152	154	150	160	175	138	219	**143**
GT + $\mathcal{N}$(0,5)	143	150	145	149	155	149	170	197	155	157	154	162	177	142	223	156
GT + $\mathcal{N}$(0,10)	156	163	155	159	164	157	183	205	165	167	167	170	186	154	230	175
GT + $\mathcal{N}$(0,15)	175	181	169	175	180	172	197	219	180	182	182	185	196	170	245	194
GT + $\mathcal{N}$(0,20)	194	202	184	194	196	192	216	233	194	199	197	200	215	191	259	204

Table 5.2: Same as in Table 5.1 for Blind Protocol 2.

Comparison of methods tested on H36M Protocol 1 (Method / MPJPE)																
[39]	**MNET1**	[201]	[451]	[452]	[210]	[453]	[186]	[225]	[200]	[191]	[192]	[204]	[171]	[217]	[173]	[178]
162	160	119	118	116	113	108	107	101	93	88	88	88	82	80	72	40

Table 5.3: Comparison of **MocapNET 1 (MNET1)** to other methods (errors in mm). MocapNET is only trained in the CMU dataset [18] so P1 accuracy is negatively biased.

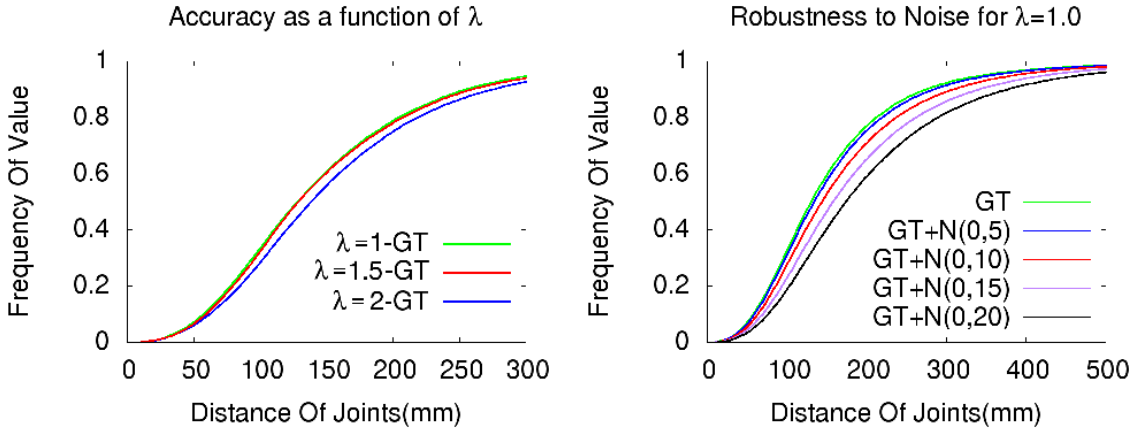


Figure 5.2: **MocapNET 1** accuracy for different λ values (left), and for various levels of Gaussian noise on the 2D input for λ = 1.0 (right).

5.1.2 MocapNET 2

We base the quantitative evaluation of the proposed method on the Human 3.6M (H36M) [39] dataset which is used to compare a variety of methods [6, 36, 98, 171, 173, 186, 188, 191, 192, 200, 204, 210, 217, 223, 225, 452, 453]. Evaluation on H36M is performed through specified protocols that use mean per joint estimation error (MPJPE) after Procrustes alignment [42] of the output of a method compared to the ground truth. The H36M protocol 1 dictates training on subjects 1, 4, 6, 7, 8 and testing on subjects 9 and 11 on 2D points originating from all available cameras. An important consideration is that the baseline method we are improving upon [6] does not train on any subjects provided by H36M. Therefore, in order to assure a fair comparison, we also performed experiments without training on H36M samples. Hence, following [6], we also label the protocol for our quantitative experiments as *Blind P1 (BP1)*.

Comparison to the MocapNET 1 baseline: The obtained results are summarized in Table 5.7. The compared methods are (a) *NN+HCD*, the full proposed solution, (b) *NN*, the estimation provided by the neural network component of our method (no HCD) and (c) *MocapNET*, the baseline approach [6]. Table 5.7 reveals that the

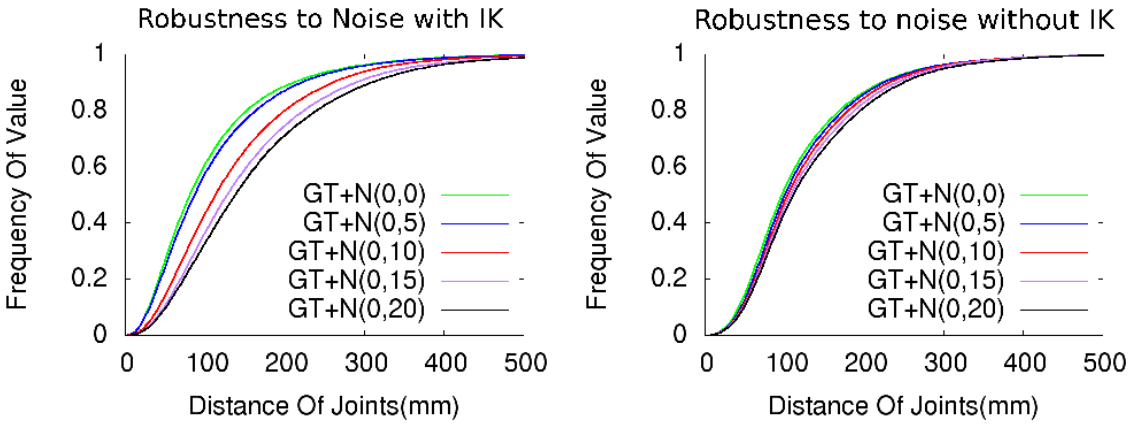


Figure 5.3: **MocapNET 2** accuracy **with** (left) and **without** (right) the HCD IK module for various levels of Gaussian noise on H36M [39] 2D input.

proposed solution (*NN+HCD*) is superior compared both to the network-only solution and to the baseline, by a great margin. Importantly, the network-only solution of our approach, is also superior to the baseline, a fact that we attribute to our superior 2D joints encoding, the twice as many orientation categories and the more elaborate orientation classifier. H3.6M [39] quantitative tests do not feature scene occluders so occlusions are tested using in-the-wild videos with cases of severe occlusions (last 3 examples on bottom row of Fig 5.11) the baseline method predictably breaks down providing incoherent results, since its single NSDM [6] matrices are architecturaly not designed to deal with this scenario. Each missing joint eliminates one line and one column of the matrix and even with a few occlusions matrices become extremely sparse causing neural network convolutions to only produce corrupted poses. The proposed approach however is much more robust to these scenarios.

Evaluation with respect to noise tolerance: In order to assess the robustness of our approach to noise, we repeated the above evaluation assuming different levels of noise contamination of the input 2D joints. Specifically, we considered ground truth 2D joint positions contaminated errors following the normal distribution $\mathcal{N}(\mu, \sigma^2)$. Figure 5.3 illustrates the relevant results by showing the percentage of joints that are estimated within a certain distance from their ground truth positions for different noise levels. We observe that when 2D joints are accurate, the inverse kinematics provides a consistent improvement. As noise becomes more intense, the performance are degraded due to the effort of the Inverse Kinematics module to strictly comply to the corrupted input data. This is in contrast to the neural network which responds in a more timid way performing some kind of internal pose filtering.

Comparison to SoTA: Comparison to other methods is summarized in Table 5.5. Keeping in mind that the neural network ensemble is evaluated at sustained rates of over 250fps on CPU execution when executed on a relatively dated i7-4790 CPU and over 70fps when using both the neural network and the unoptimized HCD module, we believe that we achieve a very good balance in the performance/accuracy trade-off. Moreover, we stress that, contrary to the rest of the evaluated methods, our method has not been trained with *any* of the H3.6M data, therefore, the comparison is disadvantageous to our method. Last but not least, our method always outputs anatomically valid results that comply to the same joint dimensions, compared to

Input	Dir	Dis	Eat	Gre	Pho	Pos	Pur	Sit	Smo	Pho	Wai	Wal	Dog	WaT	Sit.	**Avg**
MocapNET 2 (NN+HCD)	69	78	92	78	100	79	134	141	97	89	84	85	102	81	165	**108**
MocapNET 2 (NN only)	88	105	116	99	120	102	152	165	127	116	114	112	146	98	180	**122**
MocapNET 1 [6]	135	140	145	143	153	137	174	215	156	150	151	156	166	134	246	**160**

Table 5.4: Comparison of the **MocapNET 2** [8] method versus the original MocapNET 1 approach [6] with respect to the MPJPE error metric. Methods are trained on CMU and tested using H36M Blind Protocol 1.

MNET1	**MNET2**	[451]	[452]	[210]	[453]	**MNET2+HCD**	[186]	[225]	[200]	[191]	[192]	[204]	[171]	[217]	[173]	[178]
160	122	118	116	113	108	**108**	107	101	93	88	88	88	82	80	72	40
2.5	2.0	<0.1	<0.1	<0.1	<0.1	**0.6**	N/A	N/A	0.01	1.8	0.28	N/A	0.46	0.38	N/A	N/A

Table 5.5: Comparison of **MocapNET 2 (MNET2)** vs **MocapNET 1 (MNET1)** method computational performance tested on H36M Protocol 1. 1st row: MPJPE in mm (the smaller, the better), 2nd row: ratio of achieved frame rate over MPJPE (the larger, the better).

Method	MPJPE
MocapNET 2	122mm
MocapNET 2 + HCD	108mm
Standalone Levmar [433]	89mm
MocapNET 2 + Levmar [433]	78mm

Table 5.6: Comparison of MocapNET 2 using Levenberg-Marquardt as a fine tuning algorithm. Although accuracy substantially improved, Levmar added an additional 30ms of processing time per frame, an order of magnitude more processing time compared to HCD.

methods that ignore joint dimensions and relevant constraints.

5.1.3 MocapNET 3

For the quantitative assessment of our method we relied on well-established standalone hand and body pose datasets. This is possible due to our formulation that can addresses them in isolation. For qualitative evaluation we use a number of diverse use cases. We record VR sessions in retail applications to ensure realistic scenarios, we showcase our method on the SIGNUM dataset [454] that has a variety of distinct sign language gestures. We attempt body+hand pose estimation at the Leeds Sports Dataset [44] since hand pose has not been considered at it. Finally, we capture challenging performances "in-the-wild" from YouTube.

Quantitative body pose estimation experiments: We "back propagate" our methodological improvements for hand pose estimation to the body estimation task. Evaluation is performed on the H36M dataset [39] through mean per joint estimation error (MPJPE) after Procrustes alignment [42] compared to ground truth. We use the Blind P1 [6, 8] protocol for directly comparable results to [6, 8]. Table 5.7 offers an accuracy breakdown for H36M exhibiting improvements across most tasks. Results reveal an error reduction of 9% relative to the baseline [8] despite the 51% frame-rate increase from 70Hz [8] to 106Hz. Table 5.8 shows our accuracy compared to competing body pose estimation methods.

Latency measurement experiments: An important aspect to experimentally test in order to validate our claim for the very latency sensitive VR use-case is latency across the various processing steps of our pipeline.

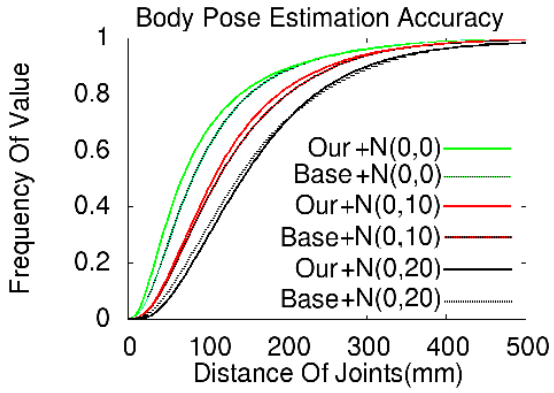


Figure 5.4: **MocapNET 3** body pose method accuracy on **H36M [39]** for increments of synthetic Gaussian noise (labeled Our) [9] vs **MocapNET 2** baseline (labeled Base) [8]. We observe consistent accuracy improvements despite the lack of an orientation classifier.

Input	Dir	Dis	Eat	Gre	Pho	Pos	Pur	Sit	Smo	Pho	Wai	Wal	Dog	WaT	Sit.	**Avg**
MocapNET 3+HCD (2021) [9]	71	78	107	86	98	79	113	121	99	98	84	100	122	104	138	**99**
MocapNET 2+HCD (2020) [8]	76	93	107	88	102	86	143	147	104	111	97	96	143	88	162	**108**
MocapNET 1 (2019) [6]	135	140	145	143	153	137	174	215	156	150	151	156	166	134	246	**160**

Table 5.7: Comparison of the first three major releases of our method with the baseline on body pose estimation. Methods are trained on CMU and tested using H36M Blind Protocol 1. All numbers are in millimeters.

MNET1	[201]	[451]	[210]	[453]	MNET2+HCD	[186]	[225]	**MNET3+HCD**	[200]	[191]	[204]	[171]	[217]	[173]	[178]
160	119	118	113	108	108	107	101	99	93	88	88	82	80	72	40

Table 5.8: 3D body estimator comparison of MocapNET 1-3 (MNET1-3) on H36M Protocol 1 (Method / MPJPE in mm).

	[41]	**MocapNET 3**	[250]	[455]	[456]	[251]	[457]	[252]	[253]
RHD	30.42	25.37	20.89	19.95	19.73	17.11	15.77	15.61	13.88
STB	8.68	9.93	7.95	8.66	8.56	7.27	-	6.93	6.71

Table 5.9: Comparison of **MocapNET 3** 3D hand estimators tested on RHD/STB (Method / MPJPE in mm).

One important deployment consideration is the camera system that often can be a bottleneck since many off-the-shelf webcams have a limited framerate of around 20fps. The sensor resolution and field of view is another important consideration since the Mediapipe Holistic 2D tracker we utilize uses a fixed 256x256 window for body pose estimation. Since at least a VGA resolution is needed in order to accommodate the tiling windows for hand pose estimation task a relatively cheap 13 year old camera system based on the OV7720/OV7221 sensor can achieve 70Hz @ VGA resolution on a well lit environment.

We have tested a variety of 2D joint estimators with most fitting candidates being the OpenPose [4] and Mediapipe Holistic [78] networks. We use OpenPose for experiments, however Mediapipe offers superior estimator performance since even in very thin clients such as mobile phones it achieves 20fps 2D joint

M.A.E.	Serial	PCA	Tr.Samples	Lambda	Tr.Model Size	Enc.Outputs	Depth	Loss Scale	Loss Exp.	Batch Size	Cont.	EDM	Dropout
86.8	262	180	3	1	62K	1	8	30	2	192	1	0	0.3
86.8	266	-	3M	1.2	105K	1	8	4	4	192	1	0	0.3
85.1	268	-	3M	1	150K	1	9	5	4	64	0	0	0.3
85.2	270	-	3M	1	150K	1	9	5	4	64	0	0	0.3
79.2	272	160	3M	0.6	131K	1	9	5	4	64	0	0	0.3
104.6	274	160	3M	0.6	138K	1	12	5	4	64	0	0	0.2
83.8	277	210	3M	0.4	446K	3	12	5	4	64	1	1	0.2
87.5	279	220	3M	0.6	234K	3	12	5	4	64	1	1	0.1
78.2	282	220	1.6M	0.6	231K	3	12	5	4	64	1	1	0.1
101.1	285	230	1.6M	0.6	253K	3	12	2	4	256	1	1	0.1
115.4	287	230	1.6M	0.5	348K	3	12	2	4	48	1	1	0.1
87.1	289	-	1.6M	1.2	117K	3	12	1.5	4	32	1	0	0.1
97.9	292	-	1.6M	0.9	188K	3	12	6.5	4	14	1	0	0.1
82.8	294	-	1.6M	1.1	134K	3	12	3.5	4	48	1	0	0.1
81.6	295	-	1.6M	1.1	134K	3	12	3.5	4	64	1	0	0.1
70.4	297	-	1.6M	1.1	134K	3	12	3.5	2	64	1	0	0.2
76.2	298	-	1.6M	1	157K	3	12	3.5	2	64	1	0	0.3
74.4	300	-	1.6M	1	157K	3	12	3.5	2	64	1	0	0.3
73.5	301	-	1.6M	1	157K	3	12	3.5	2	64	1	0	0.25
96.7	308	-	1.6M	0.2	2800K	30	12	3.5	2	64	1	0	0.22
66.5	310	-	1.6M	0.22	5700K	30	12	3.5	2	64	1	0	0.22
68.3	311	-	1.6M	0.34	2000K	30	12	3.5	2	64	1	0	0.12
69.7	313	-	3.3M	0.34	2000K	30	12	1.5	2	64	1	0	0.22
68.1	314	-	3.3M	0.18	7000K	30	12	1.5	2	64	1	0	0.18
64.8	315	-	3.3M	0.1	20000K	30	12	2.5	2	96	1	0	0.25
71.4	316	-	3.3M	0.4	1600K	30	12	3.5	2	96	1	0	0.25
64.9	317	-	3.3M	0.2	5600K	30	12	3.5	2	96	1	0	0.1
66.4	318	-	3.3M	0.25	5600K	30	12	1.5	2	64	1	0	0.1
64.6	**319**	**-**	**3.3M**	**0.19**	**6300K**	**30**	**12**	**4.5**	**2**	**64**	**1**	**0**	**0.1**

Table 5.10: Final **MocapNET 4** experiments (19/5/23 to 19/9/23) for 3D Body Pose Estimation using the CMU [16] Train/Test split.

estimation rates and in PCs it matches most webcam acquisition rates.

The method we propose achieves an aggregate average rate of 60Hz for the 2D joint to 3D pose task on a i7-4790 CPU. This includes execution time for the 2D joint to 3D output for both hands and upper and lower body ensembles as well as the hierarchical coordinate descent post-processing step.

For VR use-cases, a PC connected to a regular 802.11 g/n wireless access point via an ethernet cable introduces 0.5ms of additional latency. The aggregate framerate of MediaPipe + our method can thus theoretically be adequate for providing VR user avatars.

5.1.4 MocapNET 4

During the final months of this PhD and while having our focus on the Facial task published in [31], we navigate the very high dimensional space of possible **MocapNET 4** configurations in the course of 4 months. We manage to reach a 3D body pose estimation error of **64.6 mm** in our CMU MOCAP [16] test set. These M.A.E. measurements are more indicative of the method performance when operating with known camera intrinsics and skeleton dimensions, a.k.a. with a known setup and not in-the-wild. The high computational cost of training given the available resources is also very evident, since with very little downtime 4 months of time allowed only 28 3D human body pose training sessions. Not having implemented formulation improvements like not needing a 4x networks for each orientation, would only allow 7 training sessions in the same time using the same computing resources!

5.2 Hand Pose Estimation

Quantitative hand pose estimation experiments: Our method's compositionality allows hand tracking execution isolated from the rest of the pipeline. This allows experiments that can compare our method with

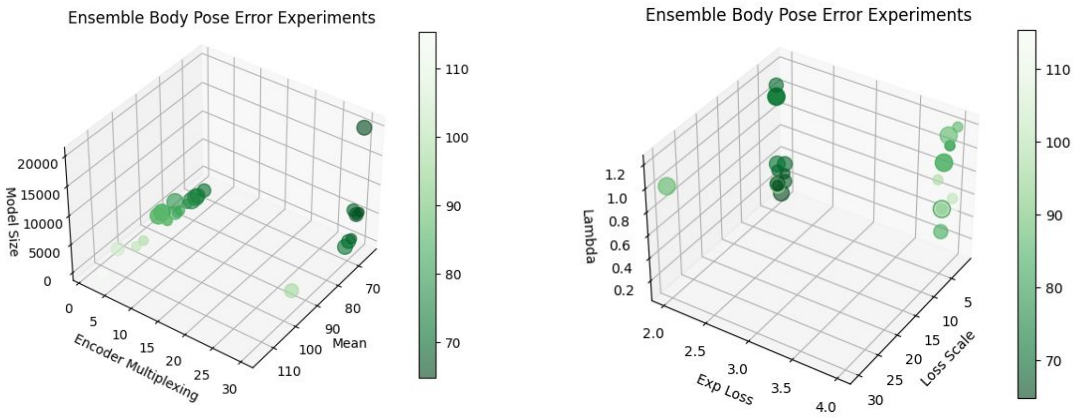


Figure 5.5: Table 5.10 3D plotted for better clarity.

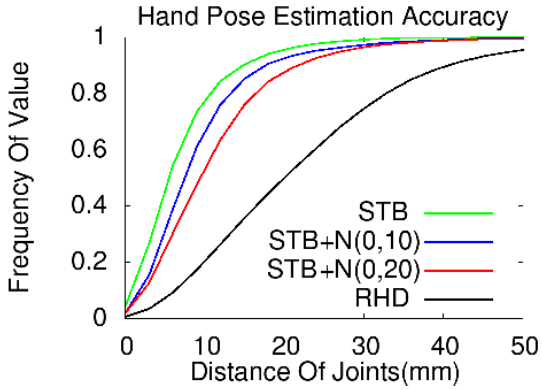


Figure 5.6: Hand pose estimation accuracy of MocapNET 3 [9] on **STB [40]** and **RHD [41]** for increments of synthetic Gaussian noise. We observe that the impact of 2D noise is less pronounced compared to the impact of non-temporally cohesive poses.

standalone 3D hand pose estimators. The STB [40] dataset features a variety of hand poses recorded using a depth camera with annotated ground truth. Position of the root joint and global scale is aligned to ground truth in the literature [36, 41] a practice we also follow for comparable results. A dataset issue is that it features a palm joint instead of a wrist. Most hand models, including our own, feature wrist joints since armatures mimic the natural bone structure of hands. Following the literature, we handle this palm/wrist dislocation via linear approximation between the wrist and metacarpal [36, 312] and do not consider the wrist joint in error calculations. We achieve a mean average error (MAE) in the sub 10mm zone. STB dataset poses are consecutive and use only the left hand. We test on the RHD dataset [41] that has no temporal continuity and features both hands to gain more insight on our method accuracy. Our method manages to retain a MAE of 25.37mm indicative of the expected accuracy and rate of recovery in cases of abrupt multi-frame hand occlusions. The Percentage of Correct Keypoints (PCK) curves are reported in Figure 5.6 and comparison with other methods is summarized in Table 5.9. Despite an average 3.22 mm higher STB error than SOTA we manage an excellent accuracy / performance ratio since the only other real-time method of

	1 Output	2 Outputs	3 Outputs	5 Outputs
RHD	26.48	25.03	24.28	27.35
STB	18.28	19.80	25.58	27.03

Table 5.11: Multiple **MocapNET 3** output multiplexed regression experiments via a single encoder on STB [40] (Method / MPJPE in mm). We do not perform the HCD step to measure the unskewed NN result. We observe deteriorating accuracy as more outputs are regressed using the same encoder.

Sample Size	Dropout	D.o.f.	HCD	Descriptor	Activation	STB avg. acc.	RHD avg. acc.
727.591	0.3	1	✓	NSRM	SWISH	10.2 mm	24.9 mm
727.591	0.3	1	✓	eNSRM	SWISH	11.8 mm	25.0 mm
727.591	0.3	1	✓	eNSRM	SELU	10.1 mm	25.2 mm
727.591	0.3	1	✓	NSRM	SELU	10.6 mm	25.1 mm
727.591	0.3	1	✗	NSRM	SWISH	15.6 mm	26.1 mm
727.591	0.3	1	✗	eNSRM	SWISH	18.2 mm	26.4 mm
727.591	0.3	1	✗	eNSRM	SELU	18.0 mm	27.1 mm
727.591	0.3	1	✗	NSRM	SELU	15.9 mm	26.5 mm

Table 5.12: Ablation study initial experiments

Table 5.9 is [457], it uses GPGPUs, and we achieve similar frame-rates to it on CPU-only execution while also computing the body pose.

Ablation study: We perform experiments to examine the 3D hand pose estimation accuracy effects of HCD, training sample size, eNSRM vs NSRM [8] descriptors, SELU [28] vs SWISH [458] activation functions and multiple d.o.f. regression via the same encoder. We observe (a) the importance of HCD, particularly in temporarily cohesive input like STB [40], (b) that larger randomized sample sizes improve resulting trained encoders especially using the more complex eNSRM descriptor, (c) that SWISH and SELU perform similarly and (d) that while regressing multiple d.o.f. from each encoder is feasible, it quickly degrades output accuracy as more d.o.f. share the capacity of a single encoder.

We conducted all the experiments listed here by acquiring the STB and RHD datasets from the VAE 3D hands repository. We used the HandOnlyTest utility to compute standalone 3D hand poses and this Python utility to compare the 3D output of the proposed method to ground truth and extract results.

Due to RAM constraints, the largest generated data samples consisted of 7.276.716 M samples. A fully trained ensemble for the left hand took approximately 6 days to train for 7M samples. Using a tenth of the sample size (727K) we where able to conduct faster experiments to cover a larger variety of combinations, but we also used an intermediate 1/2 sample ratio with 3,638,568 samples also comparing the possible variance that sample size introduces to the training procedure. It should be noted that we generate each of the 3 differently sized datasets once using this tool in order for all the experiments across a sample size to be directly comparable. The tensorflow randomized seed is also fixed to 0 for each training to be deterministic and reproducible.

The Tables below are all variations on the final proposed setup of the MocapNET3 ensemble that uses Hierarchical Coordinate Descent (HCD) optimization as a fine tuning step, eNSRM descriptors to encode 2D input data, SWISH activations, a dropout of 0.3 or 30

The first set of "small" sample size experiments using 1/10 of the final sample size for different combinations of a 2D Descriptor, comparing the proposed eNSRM with the rich diagonal with the MocapNET2 NSRM and the Self-normalizing (SELU) activations of MocapNET 1 and MocapNET 2. Finally, all runs are tested with and without the Hierarchical Coordinate Descent fine-tuning step introduced in MocapNET 2.

One of the first observations is that the HCD step plays an important role in improving accuracy, espe-

Sample Size	Dropout	D.o.f.	HCD	Descriptor	Activation	STB avg. acc.	RHD avg. acc.
727.591	0.3	2	✗	eNSRM	SWISH	19.8 mm	25.0 mm
727.591	0.3	3	✗	eNSRM	SWISH	25.5 mm	24.2 mm
727.591	0.3	5	✗	eNSRM	SWISH	27.0 mm	27.3 mm

Table 5.13: Ablation study multiplexing encoder outputs

Sample Size	Dropout	D.o.f.	HCD	Descriptor	Activation	STB avg. acc.	RHD avg. acc.
727.591	0.1	1	✓	eNSRM	SWISH	10.1 mm	25.3 mm
727.591	0.1	1	✗	eNSRM	SWISH	19.1 mm	27.2 mm
727.591	0.2	1	✓	eNSRM	SWISH	10.9 mm	24.9 mm
727.591	0.2	1	✗	eNSRM	SWISH	18.2 mm	26.7 mm
727.591	0.25	1	✓	eNSRM	SWISH	10.4 mm	25.0 mm
727.591	0.25	1	✗	eNSRM	SWISH	19.3 mm	26.8 mm

Table 5.14: Ablation study with various dropout levels

cially on temporarily cohesive input data such as the STB data. The results also show that since the HCD improvements improve the results of the neural network, in order to properly compare neural network configurations this is best done when HCD is turned off. We also observe that for this small sample size although the proposed SWISH activation performs better than the MocapNET 2 [SELU]((https://arxiv.org/abs/1706.02515) the richer diagonal actually makes the task more difficult for the neural network although the slightly increased error is still recoverable via the HCD step.

Another idea we tested is grouping together encoder outputs. Since the internal state of each of the encoders could theoretically be reused across more than one degrees of freedom, especially since many joints feature more than one degrees of freedoms, we attempted to regress 2,3 and 5 degrees of freedom using the same encoder. Testing these configurations without the HCD step reveals a gradually worse accuracy. This is to be expected since this was one of the reasons for the ensemble architecture of encoders with 1 d.o.f. . Our explanation for this behavior is that sharing the capacity of the network between more than one degrees of freedom means less network computations resources dedicated to each of the outputs. It also means that during training the loss function becomes less indicative of each of the outputs making the training procedure gradually harder.

In order to successfully deepen a densely connected network we have employed residual connections and relatively intense dropout rates. Concerned about whether the high drop-out rate could cause the network encoders (seen in Figure 2 of the paper) to over-rely on the skip connections and thus effectively really utilize only a lower total number of layers, we performed experiments varying the dropout rate to study the effects. We observe that increased dropout improves results especially in the RHD dataset, however the accuracy variance is not indicative of such a side-effect.

We also performed some ablation experiments using the full sample size proposed in the paper. The following experiments that took approximately 3 weeks to complete, show less variance between the different combinations of design characteristics when HCD is enabled.

Finally, using a dataset 1/2 the size of our final size we once again perform variations on the proposed configuration and we manage to slightly improve the best results of our method achieving 9.7 mm for the STB and 25.0 mm for the RHD. However, since the BMVC publication policies prohibit updating experimental results during the rebuttal period we did not include these slightly improved Figures on the paper.

To summarize our ablation study results, the hierarchical coordinate descent (HCD) algorithm provides significant pose estimation improvements regardless of the configuration and training of the neural network providing the initial 3D pose estimation. That being said, a network that can arrive closer to the correct so-

Sample Size	Dropout	D.o.f.	HCD	Descriptor	Activation	STB avg. acc.	RHD avg. acc.
7.276.716	0.3	1	✓	eNSRM	SWISH	9.9 mm	25.3 mm
7.276.716	0.3	1	✗	eNSRM	SWISH	20.5 mm	26.8 mm
7.276.716	0.3	1	✓	NSRM	SWISH	9.8 mm	25.0 mm
7.276.716	0.3	1	✗	NSRM	SELU	19.5 mm	26.3 mm
7.276.716	0.3	1	✓	NSRM	SELU	9.9 mm	25.2 mm
7.276.716	0.3	1	✗	NSRM	SELU	17.6 mm	27.5 mm

Table 5.15: Ablation study with different descriptors and activation functions on a very high number of training samples

Sample Size	Dropout	D.o.f.	HCD	Descriptor	Activation	STB avg. acc.	RHD avg. acc.
3.638.568	0.3	1	✓	eNSRM	SWISH	9.7 mm	25.0 mm
3.638.568	0.3	1	✗	eNSRM	SWISH	18.6 mm	26.7 mm
3.638.568	0.3	1	✓	NSRM	SWISH	10.0 mm	24.9 mm
3.638.568	0.3	1	✗	NSRM	SWISH	17.6 mm	26.1 mm
3.638.568	0.3	1	✓	NSRM	SELU	11.5 mm	25.1 mm
3.638.568	0.3	1	✗	NSRM	SELU	18.9 mm	26.3 mm
3.638.568	0.3	1	✓	eNSRM	SELU	9.9 mm	25.2 mm
3.638.568	0.3	1	✗	eNSRM	SELU	17.6 mm	27.5 mm
3.638.568	0.1	1	✓	eNSRM	SWISH	10.2 mm	25.2 mm
3.638.568	0.1	1	✗	eNSRM	SWISH	17.2 mm	26.9 mm
3.638.568	0.1	1	✓	eNSRM	SELU	10.0 mm	25.4 mm
3.638.568	0.1	1	✗	eNSRM	SELU	16.3 mm	27.2 mm
3.638.568	0.1	1	✓	NSRM	SELU	10.8 mm	25.2 mm
3.638.568	0.1	1	✗	NSRM	SELU	19.8 mm	26.6 mm

Table 5.16: Ablation study on a high number of samples varying various options

Method	**Gaze Error**
Deepwarp [459]	15.3°
He et al. [460]	8.7°
Xia et al. [461]	6.3°
Ours [−30°..15°]	14.8°
Ours [−30°..30°]	18.7°

Table 5.17: **MocapNET 4** 3D gaze estimation accuracy comparison in degrees, when measured against the Columbia Gaze Dataset [43].

lution makes HCD perform even better. A second conclusion from our experimental observations is that a reduced dropout rate does not seem to have a dramatic effect on error thus suggesting that the utilized capacity of the network remains similar under all tested dropout ratios. We observe that a larger number of samples improves network accuracy especially in the case of eNSRM descriptors that feature more complex features and also perform 2D alignment of 2D shapes that are more difficult to ”learn”. Finally, SELU and SWISH activation functions seem to perform similarly, however due to the slight edge of SWISH experiments and their improved computational performance they seem to be a good choice for the MocapNET 3 ensemble.

5.3 Head Pose, Gaze and Facial Capture

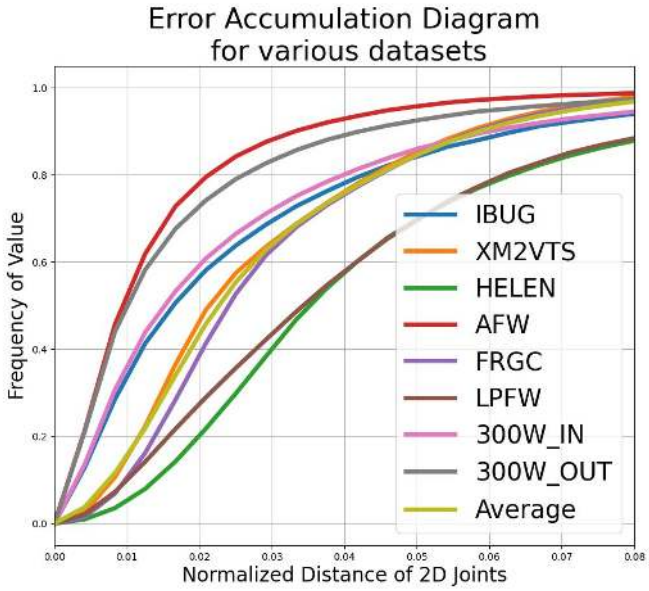


Figure 5.7: Error accumulation diagram for **MocapNET 4** 3D Face Capture. We use the 68 joint 2D reprojections of the 3D output of our method compared to ground truth using Procrustes analysis [42] across various different facial 2D landmark datasets. X-Axis uses normalized coordinates w.r.t the input image dimensions.

5.4 Dimensionality Reduction Experiments

We perform dimensionality reduction to input 2D coordinates concatenated by NSRM [9] descriptors. An interesting experiment is to only apply PCA to raw 2D data as done by [76]. Significantly different Scree and clustering plots are observed with NSRMs, as seen in Figure 3.47, giving rise to a more structured sample order. We show how translation/rotation invariant NSRM descriptors [6, 9] affect samples, highlighting them with color, based on ground-truth d.o.f. and observe that different areas gain spatial coherence compared to raw 2D data. This explains why descriptors simplify the hard task of sample separation making direct NN IK regression feasible.

To provide a broad view of the PCA input compression effect, we vary dimensionality with 210, 180, 150, 120, 90, 50, 30 and 15 PCA dimensions as seen in the first row of Table 5.20. Column pairs correspond to a fixed number of PCA dimensions. We maintain a specific network parameter size (#Par., in thousands of parameters) with each column depicting a specific parameter count. For each PCA dimension/parameter count combination, error values presented are in millimeters for rows M.A.E. and St.D. (3D mean average error and standard deviation for 3D joint locations of neck, shoulders, elbows, wrists, hips, knees, ankles) after Procrustes analysis [42] for validation sets 01-10 CMU BVH [16]. The rest of the values showcase training errors and different observations we make. Position X, Y, Z errors show the raw encoder translation output error in centimeters. We observe that maintaining less than 150 PCA dimensions causes networks to no longer accurately regress skeleton position, despite an overall better relative pose regression than the baseline method. The rest of the rows depict encoder rotation errors for each joint degree of freedom in Euler

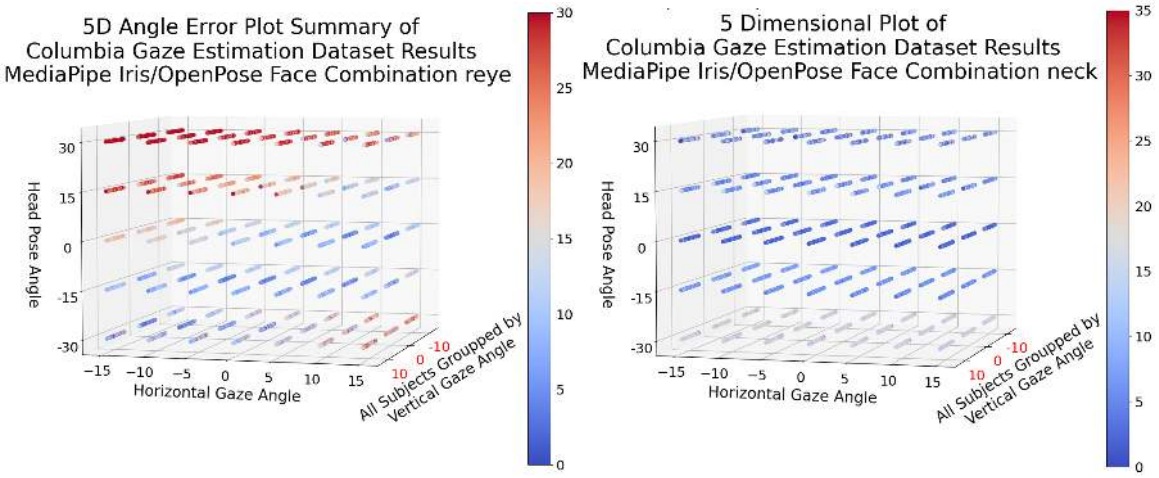


Figure 5.8: Left: Quantitative 3D eye gaze accuracy results for **MocapNET 4** measured against the Columbia Gaze Dataset [43]. The dataset contains RGB images from 56 subjects with gazes fixed at specific intervals. We regress and plot the angular error of the right eye using color, plotting all subjects adjacent one to the other. Each 3D line depicts results for all subjects. The X, Y and Z axes depict the horizontal/vertical gaze angle and head pose angle in relation to the camera. Right: Quantitative 3D neck/head pose accuracy for **MocapNET 4** seems to be uniformly good across all subjects with slightly elevated errors around the -30° limit.

degrees. Rot X, Y, Z is the absolute skeleton rotation. The rest of the abbreviations are: RS: Right Shoulder, RE: Right Elbow, LT: left thigh, LKn: Left Knee. Columns 323 use the full input without PCA in an attempt to compress using MocapNET λ values [6] to the same size as PCA experiments (32K/105K). The baseline (153K/$\lambda = 1$) is also included for a complete comparison. Despite the configuration of 210 PCA dimensions / 105K parameters achieving better training errors in all degrees of freedom its validation Procrustes M.A.E. is higher than the best configuration. Aggressive 60 PCA dim/105K param compression achieves better M.A.E. if an application is not concerned about positional components of the recovered pose, this is the case with VR applications.

Baseline method results can be found on columns labeled 323, the non compressed input element number. We observe that 210 PCA dimensions maintain virtually all input variance despite amounting to $\sim 35\%$ less input dimensions so we use this as our minimum compression setting. Since networks that have 153K parameters are created by baseline 323 input configurations and they can be reduced by $\approx 35\%$ while being certain that the culled inputs are redundant, we are motivated to experiment with networks that have a similar reduction ratio in parameter count (105K parameters). Since naïvely reducing the input leads to aggressive parameter reduction, another experiment consideration is to attempt similarly intense NN compression. Using the λ of the baseline formulation [6] with $\lambda < 1.0$ allows us to maintain a larger number of NN parameters and thus facilitate parametric control for our study. We use 32K parameters which is 80% less than the baseline although still bigger than a naïvely compressed network. We enforce the 32K/105K constant network sizes so that accuracy fluctuations can be clearly attributed to the dimensionality reduc-

Dataset	Median	Mean	St.Dev.
AFW [331]	0.90%	1.44%	0.02
300W Outdoor [45]	0.95%	1.73%	0.02
300W Indoor [45]	1.46%	2.50%	0.03
IBUG [333]	1.57%	2.69%	0.03
XM2VTS [329]	2.01%	2.52%	0.02
FRGC [330]	2.26%	2.78%	0.09
HELEN [334]	3.42%	4.33%	0.03
LPFW [332]	3.36%	4.13%	0.03

Table 5.18: **MocapNET 4** quantitative results on 3D Facial Capture, when taking ground truth 2D data, feeding it through our ensemble, extracting BVH output, rendering the skinned model shown on Figure 5.16, getting corresponding 2D joints out of the 3D model and comparing it to the input after Procrustes analysis [42]. Results use Normalized Mean Error (NME) with respect to input image resolution.

Method	m.a.d.	Ex. time (sec)
Martinez et al. [462]	0.0514	42.5
Čech et al. [463]	0.1047	4.05
Uřičář et al. [464]	0.0970	3.46
Deng et al. [465]	0.0226	1.97
Fan et al. [466]	0.0309	1.29
Ours	0.1623	0.03

Table 5.19: **MocapNET 4** quantitative on 3D Facial Capture. Comparison of the mean absolute deviation (m.a.d.) of 2D fitting results for 68 facial landmarks with the mean computational cost required. Our method uses 2D ground truth which is regressed to a BVH facial configuration and reprojected back to 2D points and compared to ground truth using Procrustes analysis [42]. Percentages reflect normalized pixel distance w.r.t. to the d_{outter} metric described in Figure 6 of [45].

tion technique and not diminished network capacity. We resort to λ variable [6] scaling to achieve this. For example 15 PCA inputs with $\lambda = 0.048$ create networks with 105K parameters, while the same input and $\lambda = 0.086$, 32K parameters.

3D human pose estimation algorithms are typically benchmarked by calculating mean per joint position error (MPJPE) after Procrustes analysis, which is translation and rotation invariant. Despite regressing higher-order BVH output instead of just 3D points, we perform this experiment since it indicates overall aggregate pose accuracy. We observe that despite the "lossy" compression, the 105K encoder network with 60 PCA input dimensions performs better in 3D Minimum Average Error (M.A.E.) compared to the baseline. This is explained (a) by the fact that the network capacity is directed towards fewer inputs with more explanatory power and (b) because input variance is not spread across more variables. We also see that supplying many redundant dimensions to a relatively small neural network has an adverse impact on learning to derive poses, despite training still having access to the same more useful inputs. An intriguing discovery is that

PCA	15		30		60		90		120		150		180		210		323		
#Par.	32	105	32	105	32	105	32	105	32	105	32	105	32	105	32	105	32	105	153
M.A.E.	86.3	85.8	83.9	81.8	82.9	**80.1**	86.2	82.6	88.4	84.7	91.3	83.6	103.2	100.8	100.8	100.9	89.0	91.9	89.0
St.D	35.3	35.7	35.2	**34.9**	37.1	37.0	39.1	38.5	40.4	43.8	47.5	43.7	57.3	58.2	52.3	52.1	46.7	46.7	46.7
PosX	158.2	154.9	152.9	147.0	139.3	132.2	130.0	119.7	37.6	31.1	19.3	15.8	18.7	14.6	16.8	**13.1**	20.3	15.9	15.0
PosY	59.6	58.9	58.5	57.0	55.2	53.2	53.7	51.1	48.2	44.9	7.7	6.1	7.2	5.5	6.7	**5.2**	7.6	6.2	5.9
PosZ	48.7	45.4	47.4	43.9	27.3	25.1	26.5	23.9	24.1	20.8	19.2	16.3	17.3	14.6	15.7	**12.4**	18.4	15.3	14.7
RotZ	12.9	12.7	12.6	12.3	12.1	12.0	12.0	11.8	11.1	10.6	10.3	9.4	10.3	9.4	10.2	**9.2**	10.7	9.7	9.5
RotY	54.6	53.3	51.7	49.8	46.0	43.9	43.1	41.0	26.4	24.5	23.5	21.5	23.1	21.4	22.0	**20.4**	25.8	23.6	23.4
RotX	15.5	15.1	14.9	14.6	14.4	14.1	14.2	13.8	13.4	12.8	12.1	11.3	12.0	11.1	11.5	**10.6**	12.9	11.7	11.4
Z/RS	16.6	15.9	15.4	14.6	14.7	13.8	14.5	13.6	14.2	13.1	14.0	12.8	13.7	12.5	13.4	**12.4**	14.4	13.4	13.1
X/RS	16.9	16.2	15.8	14.9	15.2	13.9	14.8	13.7	14.2	13.1	13.8	12.6	13.6	12.4	13.5	**12.3**	14.6	13.4	13.0
Y/RS	17.1	16.1	15.9	14.7	15.0	13.5	14.3	12.9	13.5	12.1	13.2	11.7	13.0	11.2	12.7	**11.1**	14.2	12.5	12.0
Z/RE	5.1	5.0	4.7	4.5	4.5	4.3	4.4	4.2	4.3	4.1	4.3	4.0	4.2	4.0	4.2	**4.0**	4.4	4.2	4.1
X/RE	7.9	7.6	6.9	6.6	6.6	6.3	6.5	6.2	6.3	6.0	6.2	5.9	6.2	5.8	6.2	**5.8**	6.4	6.1	6.0
Y/RE	13.4	12.8	11.2	10.6	10.6	9.9	10.3	9.6	9.9	9.2	9.7	8.9	9.5	8.7	9.6	**8.7**	10.2	9.1	9.2
Z/LT	12.9	12.6	12.5	12.1	12.1	11.7	11.7	11.3	12.1	10.9	11.1	10.7	10.9	10.5	10.7	**10.2**	11.6	11.1	11.1
X/LT	13.3	12.9	12.5	12.0	11.8	11.4	10.9	10.1	12.0	7.1	7.5	6.6	7.3	6.4	7.0	**6.0**	9.7	8.4	8.2
Y/LT	15.6	15.3	15.2	14.9	14.8	14.3	14.1	13.7	14.9	13.1	13.3	12.9	13.2	12.8	13.1	**12.6**	14.0	13.4	13.4
Z/LK	8.3	7.9	7.2	6.9	6.8	6.1	6.6	6.1	6.6	5.6	5.6	5.4	5.6	5.3	5.4	**5.3**	6.2	6.1	5.7
X/LK	11.1	10.5	9.3	8.7	8.5	7.8	8.0	7.6	8.5	6.4	6.7	6.1	6.9	6.0	6.4	**5.9**	7.3	6.8	6.7
Y/LK	8.0	7.9	7.3	7.2	7.0	6.6	6.6	6.6	6.7	6.2	6.1	6.1	5.9	5.9	5.9	**5.8**	6.5	6.5	6.3

Table 5.20: Ensemble/encoder accuracy analysis, when comparing regression results against ground truth.

Dimensionality Reduction Method	Dims	Min	Max	Mean	Median	Variance	Std. Dev.
PCA with full SVD [424]	150	14.0	300.4	83.6	76.0	1906	47.5
PCA with full SVD [424]	210	11.2	361.8	100.9	93.0	2717	52.1
PCA randomized [425, 426]	150	**9.7**	345.9	112.4	107.3	2249	47.4
PCA randomized [425, 426]	210	10.9	390.3	126.3	124.8	4561	67.5
Fast ICA [428]	150	9.8	364.4	137.7	138.6	2988	54.7
Fast ICA [428]	210	14.1	438.4	141.9	131.3	4854	81.0
Factor Analysis [427]	150	10.7	414.5	155.6	158.1	4742	68.9
Factor Analysis [427]	210	10.8	405.4	153.4	161.9	5323	73.0
PCA with full SVD [424]	60	16.2	314.9	**80.1**	**72.3**	**1370**	**37.0**
Baseline/No Dim. Reduction	323	14.8	**298.7**	91.9	80.7	2185	46.7

Table 5.21: Analysis of the effect of different dimensionality reduction techniques in terms to rotation/translation invariant procrustes M.A.E. on the problem when fixing the number of kept input dimensions to 150 or 210 (from the original 323), the number of network parameters to 105K and training until early stopping is activated to ensure a maximum optimisation budget. Values represent M.A.E. in millimeters after 3D Procrustes Analysis [42] on the test set. We observe that SVD/PCA behaves best compared to other methods and that the same method with 60 input dimensions surpasses the baseline that however is still better than all 210 dim alternatives.

these less significant inputs are not entirely disregarded by back-propagation. Finally, despite managing to recover the 3D pose we observe that the absolute 3D position is not correctly recovered with less than 150 input dimensions (Pos X,Y,Z rows in Table 5.21). This is an important consideration for applications requiring accurate translation data, since they will need to use a combination of different PCA dimensionalities for different degrees of freedom of the problem something that is permissible by the extensible ensemble formulation of [9].

Another interesting aspect is training randomization performed in [9]. Both position and orientation

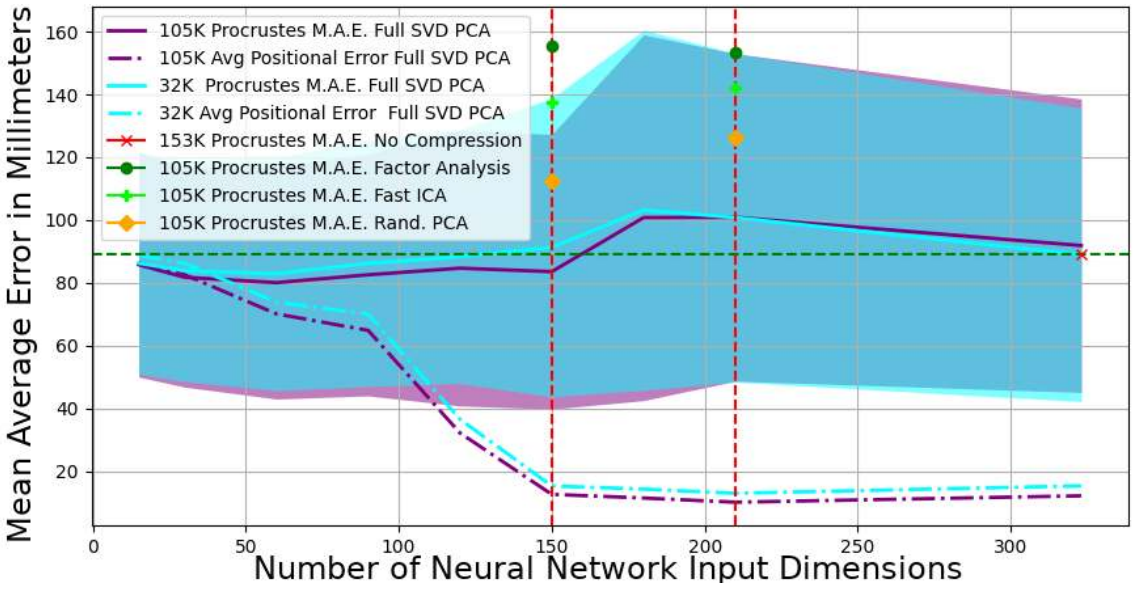


Figure 5.9: M.A.E. plot for experiments summarized in Tables 1 and 2. Colored area represents standard deviation w.r.t mean error for 105K and 32K SVD/PCA networks of various input dimensionalities. The 210 dim. (vertical red line) configuration is used as the non SVD/PCA experiment basis in Table 2. We observe a complex landscape where although smaller (32K) networks consistently perform worse than larger (105K) ones for ¡ 150 PCA dimensions, as less essential PCA dimensions are added they increase the difficulty of the regression task. We manage to perform better than the baseline for configurations under the green horizontal line despite the reduced parameter counts.

are picked from a uniform random distribution during data augmentation. Their entropy is thus much higher than the MOCAP recorded relative joint poses that exhibit regularities due to the constrained range of motion of the human body. We believe that this is the reason why a 35% smaller NN with 60 PCA dimensions surpasses the baseline in M.A.E. since PCA only maintains frequent and important data. On the other hand the high-entropy positional (Pos X,Y,Z) and orientation (Rot Y) d.o.f. need more than 150 PCA dimensions to be effectively regressed as seen in Table 5.20 and Fig 5.9.

Since 210 dimensions seem to be enough in terms of representational power and different network sizes (32K/105K) exhibit similar behavior in terms of M.A.E. and St.D., this prompts use of this configuration to benchmark different decomposition methods and study their behaviour at this setting. Table 5.21 and Figure 5.9 show that the NN error is adversely affected by other decomposition algorithms compared to SVD/PCA. It should be noted however, that this should not necessarily be interpreted as that the algorithms themselves are not efficient at the dimensionality reduction task, but rather that the selected dimensions are somehow ill-fitted for the problem we examine. Finally, we study dimensionality reduction impact to specific degrees of freedom of the problem since BVH files mainly contain rotational output. Results are presented in Table 5.20 in rows 6-20. Although maintaining more input variance with bigger NNs predictably performs better, experiments reveal that for applications that do not require a lot of accuracy even aggressively compressed NNs can accommodate the task.

A final finding is that the original λ variable compression proposed for MocapNETs [6] performs worse compared to λ compressed networks with intelligent input dimensionality reduction. For example comparing 323 input parameters that for $\lambda = 1.95/\lambda = 2.7$ create 32K/105K encoders with a PCA basis of 120 input elements we see less error in terms of M.A.E., and can be more accurate in terms of standalone encoder accuracy for 90 dimensions and more.

5.5 Qualitative Body Pose Estimation

Qualitative 3D pose estimation experiments are important, since they provide visual information and cues to evaluate the discrepancies of regressed data compared to input observations. Although the detailed quantitative experiments we conducted allow for rigorous systematic analysis and comparison of the method, visualizing our BVH MOCAP output using a skinned model next to the RGB observation is important to confirm the interpretability of our output when also taking into account all of the other intricate details introduced by 3D graphic engines and 3D models. Visualizing results on challenging poses and inspecting the robustness, consistency and accuracy of the recovered poses can provide a deeper understanding of our proposed system in real-world scenarios.

5.5.1 MocapNET 1

Results from MocapNET 1 can be seen in Figure 5.10. During the first iteration of the method, we collected samples from youtube building a small library of videos that where used until the completion of the project to monitor method improvements. An OpenGL based renderer was also written from scratch in C to animate our BVH results. We exported the polygons and bone weights from a 3D human model we acquired from Makehuman. After performing the necessary joint associations of the model joints to our output, using the BVH output of our method we animated the 3D model without textures resulting in Figure 5.10. Despite limbs not aligning perfectly with the observations the overall performance of the method was good and the yielded BVH poses where consistent with the observed RGB frames.

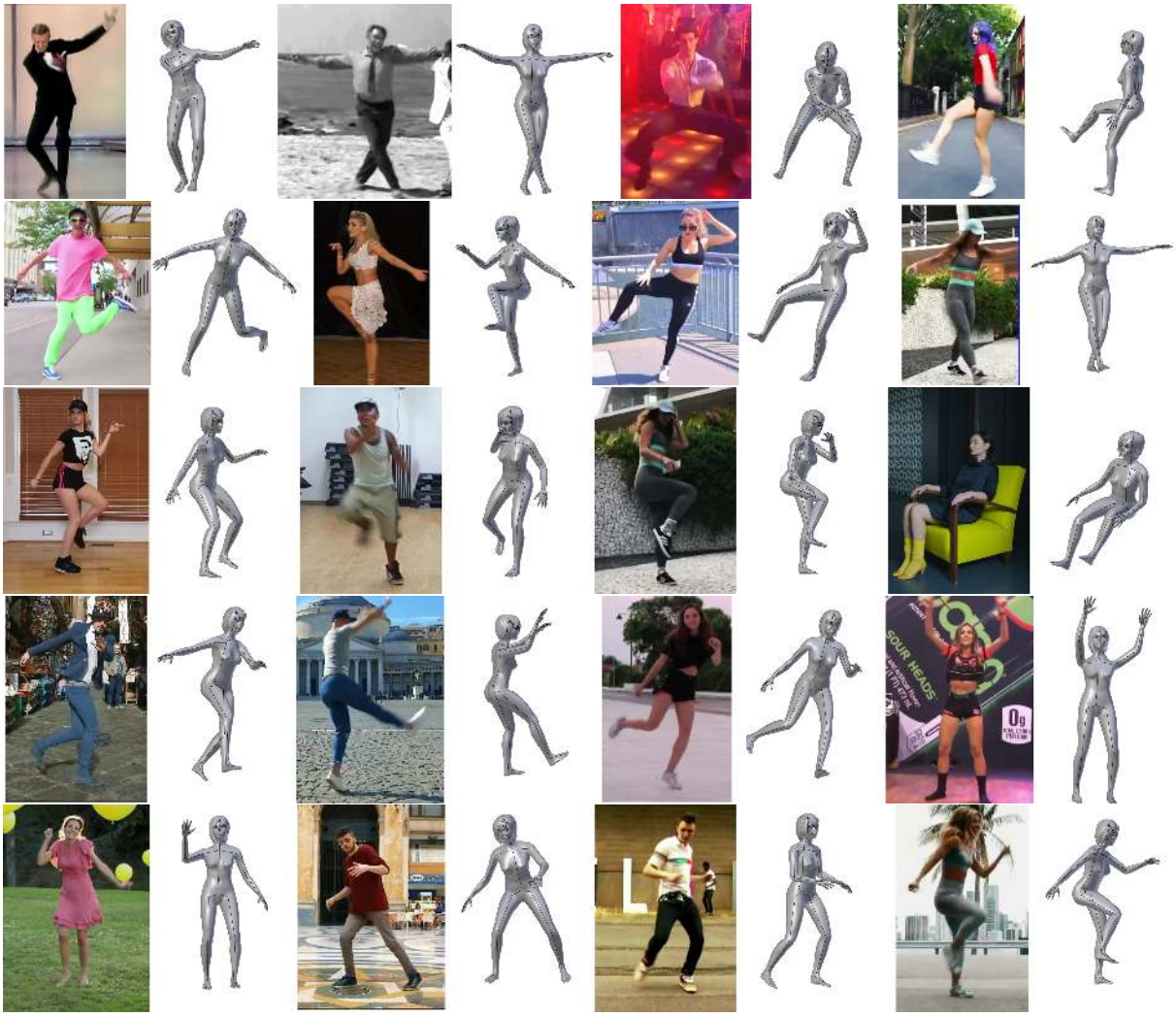


Figure 5.10: Qualitative results from **MocapNET 1** [6].

5.5.2 MocapNET 2

MocapNET 2 focused more in improving the accuracy of the initial version of the method. This prompted qualitative experiments where their results could be directly compared both in terms of their individual improvements as well as to the input RGB observation. Figure 5.11 summarizes these experiments showing the expected improvements originating from the occlusion tolerant properties of the second iteration method as well as the more fine grained orientation classification and larger NN capacity.

Figure 5.11: Qualitative difference between the first version of **MocapNET 1** [6] (with red) compared to its **MocapNET 2** iteration [8] (with green) when tested on "in-the-wild" YouTube videos. We observe improved accuracy, robust orientation classification and better occlusion tolerance.

In order to more properly depict the output of the method, instead of only relying on Youtube videos we used another popular data source from our literature review (Table 2.6), namely Leeds Sport Dataset [44]. Results, summarized in Figure 5.12 revealed good performance over a very large variety of exotic poses from various sports, views, and subjects.

Figure 5.12: Qualitative results of **MocapNET 2** on the Leeds Sport Dataset (LSP) [44] using the second generation of MocapNET [8].

5.5.3 MocapNET 3

MocapNET 3 combined 3D boby pose estimation with 3D hand pose estimation. Having confirmed the interpretability of the BVH output with a skinned model such as makehuman, qualitative visualization shifted to a stick model in order to provide higher resolution on the hands that where now a part of the regressed output. Qualitative results from MocapNET 3 are summarized in Figure 5.13.

Qualitative results in VR scenarios: We employ two retail Oculus Quest VR applications, Elixir and Hand Physics Lab (HPL), for qualitative assessment of our method. They are both state of the art in terms of the provided onboard VR hand tracking experience and feature a variety of interesting to track in-game tasks and interactions. Elixir takes place in a room-sized space requiring lateral movement while HPL doesn't since all in-game objects are within arms reach. Our camera faces the user in the initial orientation of the application when it first loads. Testing both scenarios allows us to study our method while controlling for body pose variations. We observe high fidelity 3D BVH skeletons throughout the activities.

Some artifacts are observed in the head pose orientation due to the VR-headset obscuring facial features (eyes, nose and ears) that cause 2D joint estimation and subsequently 3D regression to sometimes be inaccurate. This is however a non-issue for VR applications since the head tracking accuracy provided by the headset sensors is incredibly accurate. Very often during the trials we are able to retrieve hand poses that are not visible by the headset due to the hands going out of its view frustrum.

Qualitative results in sports: Leeds Sport Dataset [44] consists of standalone frames depicting various sport activities with no temporal, subject or activity cohesion. It bears similarities to RHD [41] albeit targeting bodies with non synthetic images. Our method accommodates this challenging task extracting BVH skeletons including hands for the first time.

Qualitative results in sign language: The SIGNUM dataset [454] is a multi person monocular RGB dataset

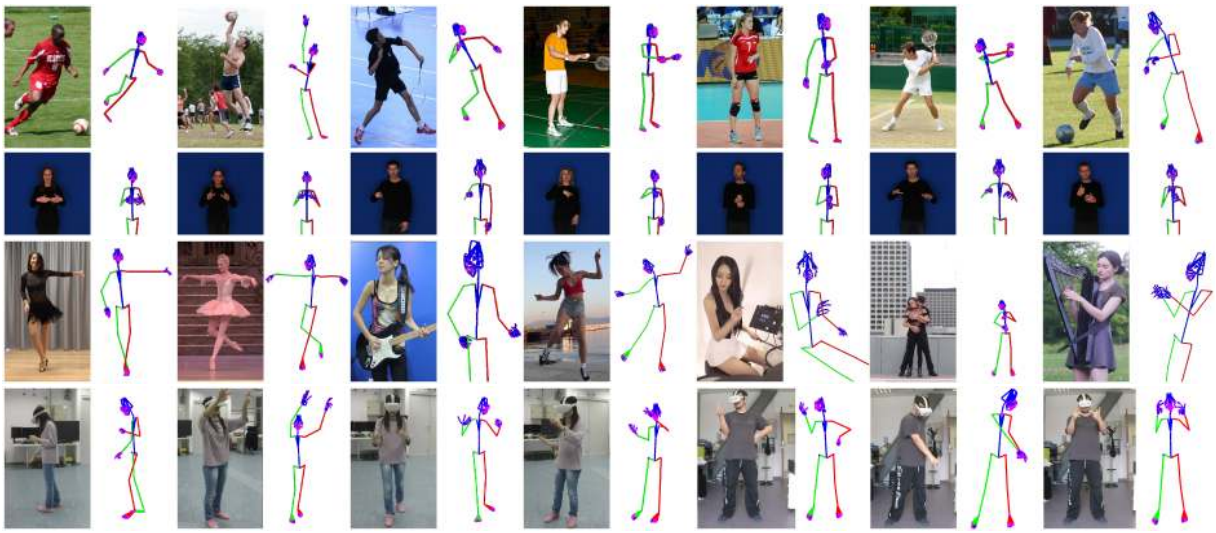


Figure 5.13: Qualitative results of **MocapNET 3** in LSP(first row), SIGNUM(second row), Youtube (third row) and VR (fourth row).

designed for sign language system training. We use it to qualitatively test our hand+body pose estimation ensemble with its fast, complex and challenging sign language gestures. Our method manages to generalize across multiple users of different genders and body types and we observe high fidelity 3D pose capture. The lower-body is constantly cropped out of the input feed, testing the ensemble occlusion tolerance properties [8].

Qualitative results in in-the-wild Youtube videos: We browse YouTube and collect videos that contain dancing and instrument performances that show interesting body and hand motions. We proceed to successfully track them despite the videos exhibiting unknown camera setups, rapid camera changes, rapid motions, self-occlusions and motion blur. Sample qualitative results are illustrated in Figure 5.13.

5.6 Qualitative Hand Pose Experiments

5.6.1 MocapNET 3

In a manner similar to body pose estimation, in order to properly assess the hand pose estimation capabilities of our presented method we conducted qualitative experiments focusing on hands. We used the RHD and STB datasets from our Literature Review (Table 2.7). Hand pose estimation debuted as a capability of the method during the MocapNET 3 iteration of our formulation. Results of experiments are summarized in Figure 5.14 for STB and Figure 5.15 for RHD. We observe the STB dataset that consists of continuous frames being very accurately tracked by the method, while the more demanding RHD dataset that consists of single frame synthetic poses being also successfully tackled, albeit with larger discrepanies compared to the input.

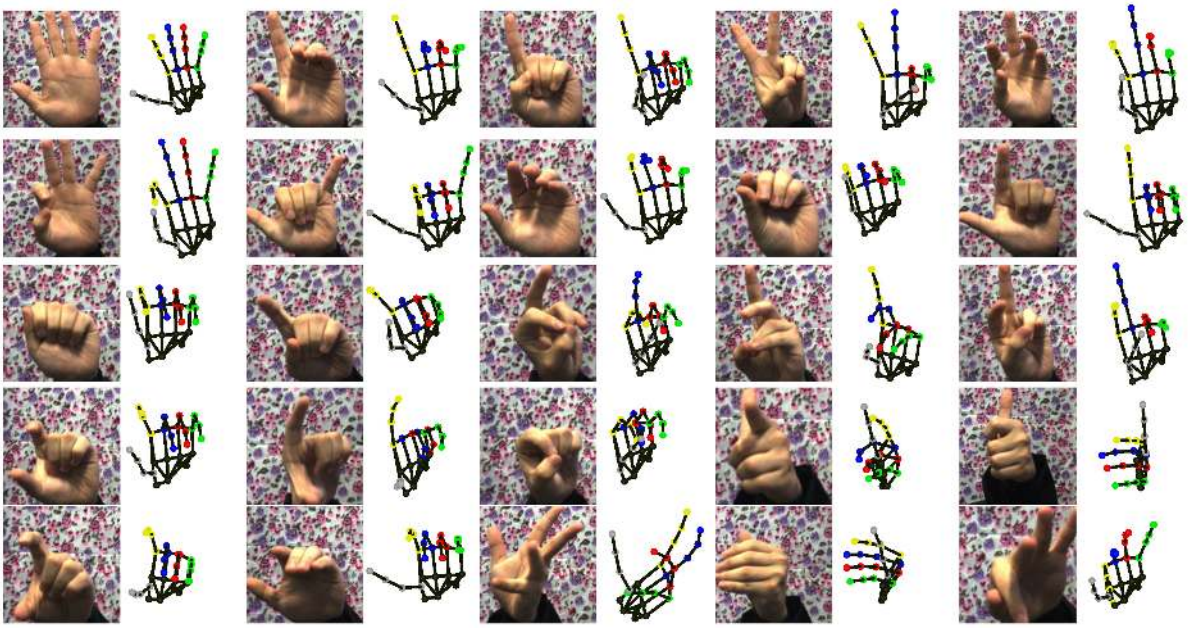


Figure 5.14: Qualitative results of **MocapNET 3** 3D hand tracking on the STB dataset [40].

Figure 5.15: Qualitative results of **MocapNET 3** 3D hand tracking on the RHD dataset [41].

5.7 Qualitative Head / Gaze / Facial Experiments

5.7.1 MocapNET 4

During the final iteration of our method and in order to complete our work on Total Human Capture, we integrated a 3D Head/Gaze and Facial recovery ensemble to our method. Due to the BVH skeleton being very sparse and not indicative of the quality of results we once again resorted to a makehuman skinned model for our visualization. Indicative results can be seen in Figure 5.16.

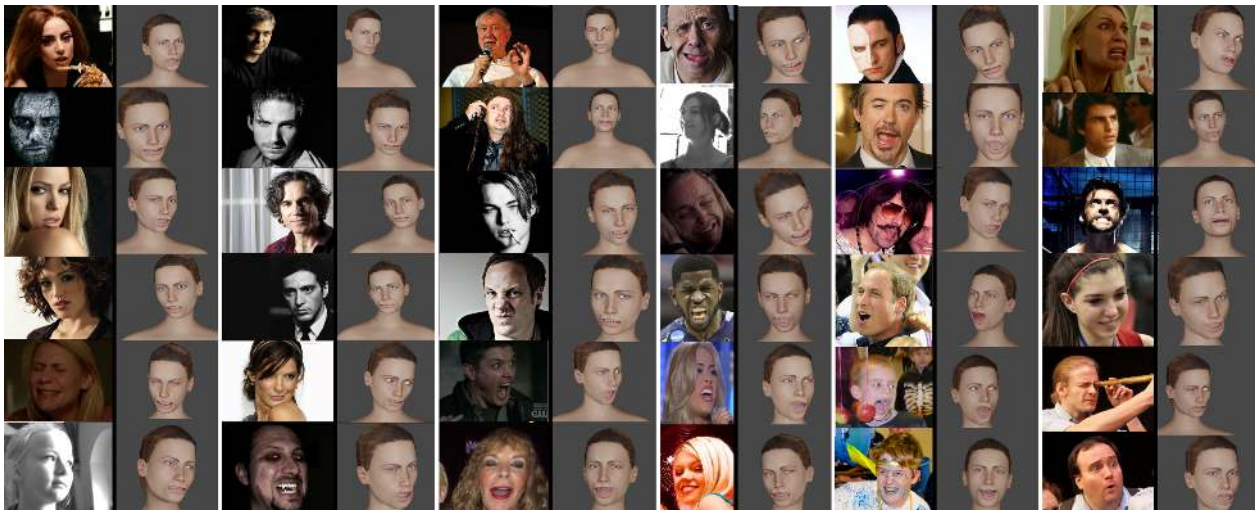


Figure 5.16: Qualitative results of **MocapNET 4** from the 300W dataset [45] adding iris data extracted with MediaPipe Iris [46] to dataset 2D landmarks. We render the BVH skeleton acquired by our NN ensemble using the same MakeHuman skinned model [23] and Blender [21]. The six rows of illustrations contain RGB input images (left) and retrieved renderings (right). We observe that the configuration of the input face, gaze and pose is respected by our method. We also observe that the sparse 2D input in conjunction with the BVH container means that the 3D rendered output is less expressive despite managing to convey facial expressions like being puzzled, smiling, anger, fear and talking.

Chapter 6
Conclusions

We presented MocapNET, a markerless 2-stage real-time RGB to 3D BVH Total Human Capture solution. Its formulation begun from a purely Body Pose Estimation method and gradually successfully generalized to the neighboring problems of hand, gaze, head and facial capture. The method was designed using first principles and was the first to regress 2D joint estimations to BVH angles in an end-to-end fashion. The research focused on developing an efficient and practical method that can be easily deployed using current off-the-shelf hardware in real-world scenarios for applications such as virtual reality, human-computer interaction, robotics, sign language and computer graphics. The proposed research also considered the compatibility of the output format with existing software and open standards ensuring the accessibility of our results to a wider audience. Implementation-wise its design was frugal relying only on the Tensorflow back-end and the OpenCV framework, implementing all other required code from scratch to reduce dependencies. Due to requirements introduced by the BONSAPPS project, the final version was re-implemented from scratch in python providing a slower to execute but easier to modify MocapNET run-time, that can be deployed with a single click in Google Colab [405].

The disadvantages of the method mainly stem from its 2-stage architecture. Although working using sparse 2D points allows for great regression speed, not encoding appearance makes it impossible to recover from 2D estimation error, while the ambiguity of the mathematically ill-posed 2D to 3D task can lead to erroneous symmetric solutions that share a common 2D appearance while having different 3D values (Figure fig:symmetries). All these design fundamentals lead to lower accuracy that can be partly recovered by knowing the dimensions of the observed human and intrinsics of the camera.

Our research included data collection, data preprocessing, NN model development, model integration, model evaluation, calibration, optimization and deployment, result analysis and interpretation, and dissemination of results. Through extensive experiments, evaluations, and result analysis, the strengths, weaknesses, and limitations of the proposed approach where systematically identified and recorded. The resulting insights to guide future research directions will be examined in Section 6.3. The findings of our research are expected to contribute to the field of human behavior understanding from visual data and advance the state-of-the-art in concurrent real-time 3D body pose estimation, 3D hand pose estimation, 3D face estimation and 3D gaze estimation. The developed algorithms, software, and datasets where disseminated through a series of publications [6,8,9,20,31], presentations in various events like the European Researcher Night, and open-source repository releases to share the knowledge and promote further research in the field. Our work awarded the author a BONSAPPS AI Talent Award (Figure 6.1), the Maria Manassaki bequest, funding from VMware University Research Fund (VMURF) and participation in tha Archimedes AI summer school, various EU projects like Co4Robots, Mingei, Greek Research projects like I.C Humans from

Figure 6.1: Top: AI Talent Award (Winner No. Bons_1OC_20) by the BONSAPPS EU H2020 (Grant no.101015848) project. Bottom: snapshot from interactive demos of MocapNET during the Foundation for Research and Technology "Researcher Night 2022" among other dissemination events.

HFRI and other projects like Signguide and Healthsign.

6.1 Synopsis of Thesis Contributions

The presented work lead to a number of novel contributions to the field that are briefly described on the following synopsis:

- **NSDM 2D joint descriptor**: Normalized Signed Distance Matrices (NSDMs) where developed as a novel descriptor part of the original MocapNET 1 [6] implementation of this work.
- **NSRM/eNSRM 2D joint descriptor**: (enhanced) Normalized Signed Rotations Matrices (e/NSRMs) where developed as novel descriptors and parts of the subsuequent MocapNET 2-4 iterations of the method.
- **MocapNET**: We proposed a novel ensemble of MLP encoders that can directly regress 3D pose+IK from RGB/2D estimations.
- **MLP Capacity Analysis**: Using the literature and knowledge about our high-dimensional problem we suggested an activation path MLP capacity metric, while also studying the effects of independent vs jointly regressed loss.
- **Hierarchical Coordinate Descent**: We proposed a novel inverse kinematics fine-tuning algorithm built specifically to complement our independent NN encoder ensemble design.
- **End-to-end BVH (Biovision Hierarchy) regression**: To the best of our knowledge, this was the first work to directly regress 3D Motion Capture output directly compatible with a widely used open-standard MOCAP format, both individually for the body,hands and face as well as for the total capture problem.
- **Adding Gaze to Total Capture**: To the best of our knowledge, our work was the first to include 3D gaze estimation to the total capture problem.
- **BVH standard extension**: Our work extends the BVH specification by allowing the definition of quaternions, using the .qbvh filename extension.
- **Robot Operating System (ROS) TF-Compatible 3D estimations from RGB**: To the best of our knowledge, our node was the first to be able to provide accurate articulated 3D human output using the ROS TF-Tree from RGB sensors.
- **Real-time 3D Total Capture**: Our work is 1 out of 7 globally that has achieved concurrent Total 3D Human Capture of hands, face and body. To the best of our knowledge our method is the only one to achieve this in real-time while also retrieving 3D gaze and being directly compatible with open-standards without relying on proprietary 3D Models like SMPL [207].
- **Dimensionality Reduction via input PCA**: Our work manages to be executable at interactive rates on low-resource embedded devices like the Raspberry Pi 4 using a lower dimensionality NN powered by PCA reduction.
- **Examination of different dataset generation schemas** : Our work explores the use of supervised [16], semi-supervised [9] and unsupervised [31] (using quasi-random sobol sequences) dataset generation.

- **Novel explored ideas** : During the course of this work we explored the use of a diffusion-based formulation along with its theoretical limits on the accuracy based on image resolution. A probabilistic MocapNET output, and came up with novel concepts like EigenPoses and Online Hard Example Mining.

- **Novel codebase** : During this work we developed publicly available C/Python code that is readily available to the research community that can be used to perform 3D total capture. Moreover we also provided plugins for the Blender/Makehuman toolchain to allow for realistic ray-traced Avatar rendering.

6.2 Development Timeline and Statistics

The development timeline and statistics of the presented thesis are summarized in Figures 6.2, 6.3 and 6.4. The development of the code-base was a massive undertaking and took part in three different repositories. The initial version of the project used Tensorflow 1.10 on Ubuntu 18.04 while the final MocapNET 4 works on machines up to Ubuntu 22.04 with Tensorflow 2.14 while also supporting the Open Neural Network eXchange (ONNX), TF-Lite and TensorRT runtimes. All of the code of the project was internally tracked committed and developed using FORTH/ICS CVRL internal GIT repository with automatic statistics recording that are shown in Figure 6.2. After our successful publications, evaluation snapshots of the code (without containing the implementation for training new MocapNETs) where submitted to the MocapNET github repository. At the time of writing this thesis, the public repository has been starred 728 times and has been forked 125 times. The statistics of the development of the Github repository can be shown in Figure 6.3. Finally the BONSAPPS project which awarded the AI Talent grant that provided funding towards the latter part of this PhD required a third repository hosted on Gitlab statistics of which can be found on Figure 6.4. Maintaining internal consistency between the different repositories while also supporting different branches for the different publications and back-porting code-improvements required a lot of effort. The original implementation of the runtime for the project was in C/C++, however during the BONSAPPS project and due to its requirements almost all of the MocapNET run-time code had to be rewritten from scratch in Python.

Between the first iteration of MocapNET v4 that officially started at 28/4/2022 with serial number #77 and the last recorded training at the time of writing these lines at 29/9/2023 (serial #321) the MocapNET database archiving system logged 483 experiments for all the involved hierarchies (upperbody, lowerbody, lhand, reye, mouth). Upperbody ensembles where trained 140 times, lowerbody 143 times, left hands 37 times, whole faces 93 times, right eyes 49 times and mouths 21 times. Training sessions took place in three different machines. The three Operating Systems used during these training sessions where Ubuntu 18.04 up to Ubuntu 22.04. 89 training sessions took place on an Ubuntu 18.04 environment, 284 training sessions took place on Ubuntu 20.04 and 118 training sessions on Ubuntu 22.04. This is important as a testament to the compatibility and compliance of our implementation to various architectural API changes that happened during the course of development. Of all the training sessions, 91 took place on the author's personal computer equipped an AMD FX(tm)-6300 CPU 8GB RAM and a GeForce GTX 1050 GPU with 2GB VRAM. These training sessions where smaller in size (due to the available resources on this system) and where used to quickly iterate through various improvements. Of all the rest, 306 experiments where carried out on the workstation provided to me by FORTH hosting an i7-4790 CPU with 16GB RAM with a GeForce GTX 1070 GPU with 8GB VRAM. 114 experiments where carried out on a machine reserved for demos during its downtime, and 63 experiments where carried out on a powerful super-computer with a AMD Ryzen Threadripper PRO 3955WX 16-Cores CPU with 503GB RAM hosting 2x NVIDIA RTXA6000 GPUs with 98GBVRAM

that became available at 20/04/2023. This latter system hosted mainly large Sobol training sessions with 16M samples that could not be previously realized due to hardware limitations in the other systems.

Assuming an average consumption of 400W, 600W, 800W and 1400W for the above mentioned workstations and an average training time of 24 hours, approximately 9585kWh of energy were consumed during this stage of the project for the MocapNET training effort. Assuming a Carbon Intensity for Greece of 385 $gCO2/kWh$, this means that close to 3.6 tons of CO_2 where released to the atmosphere as a direct result of our work. Assuming that a tree can absorb 0.5t of CO_2 from the atmosphere each year we estimate that it would take planting 16+ trees to offset the cumulative emissions of this project and make similar training efforts sustainable in the future. Assuming a price of 0.204 Euros per kWh for Greece in March 2023 the electricity cost for the project was 1955 euros. Although these metrics might not seem relevant they are recorded as a token of recognition of the wider impact of our work, and the seriousness of the effects in climate change. Not being wasteful and conducting redundant experiments was a core consideration during this thesis. A testament of that is that treating the problem using symmetries would have consumed at least an additional 86 experiments (17.8%) or 0.64 tons of CO_2 while at the same time being counter-productive for development due to decreasing the available time for the rest of the training effort. Of course the actual carbon footprint including the actual computers and infrastructure is far greater, however both our training effort and the resulting method had energy efficiency as a serious consideration.

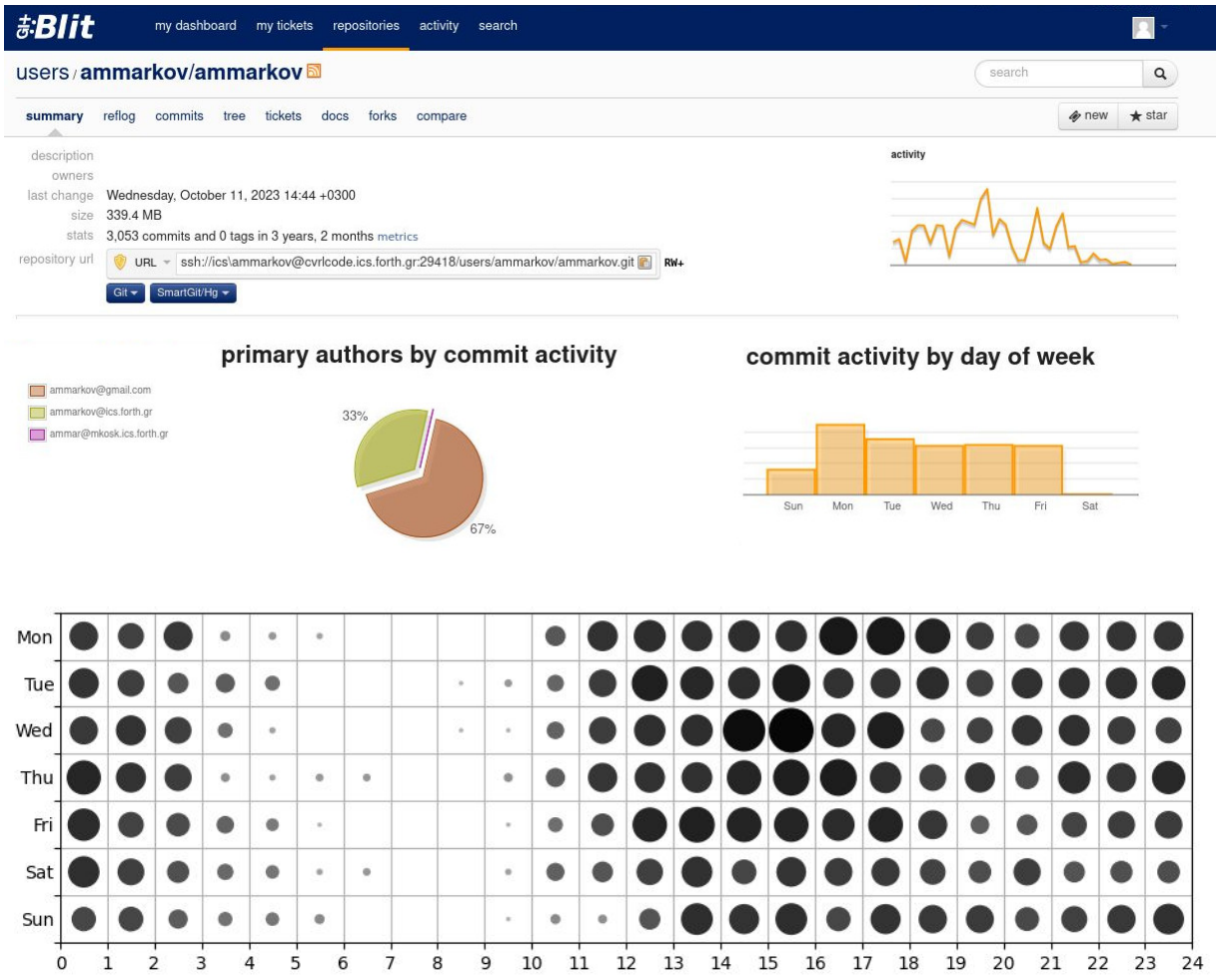


Figure 6.2: Snapshot of the GitBlit front-end for the internal GIT repository of FORTH-ICS CVRL. 2038 commits (67%) where done from the ammarkov@gmail.com identity and 1014 (33%) from ammarkov@ics.forth.gr and 1 commit from Maria Koskinopoulou's workstation during MocapNET 2 clustered training to correct a distribution specific bug.

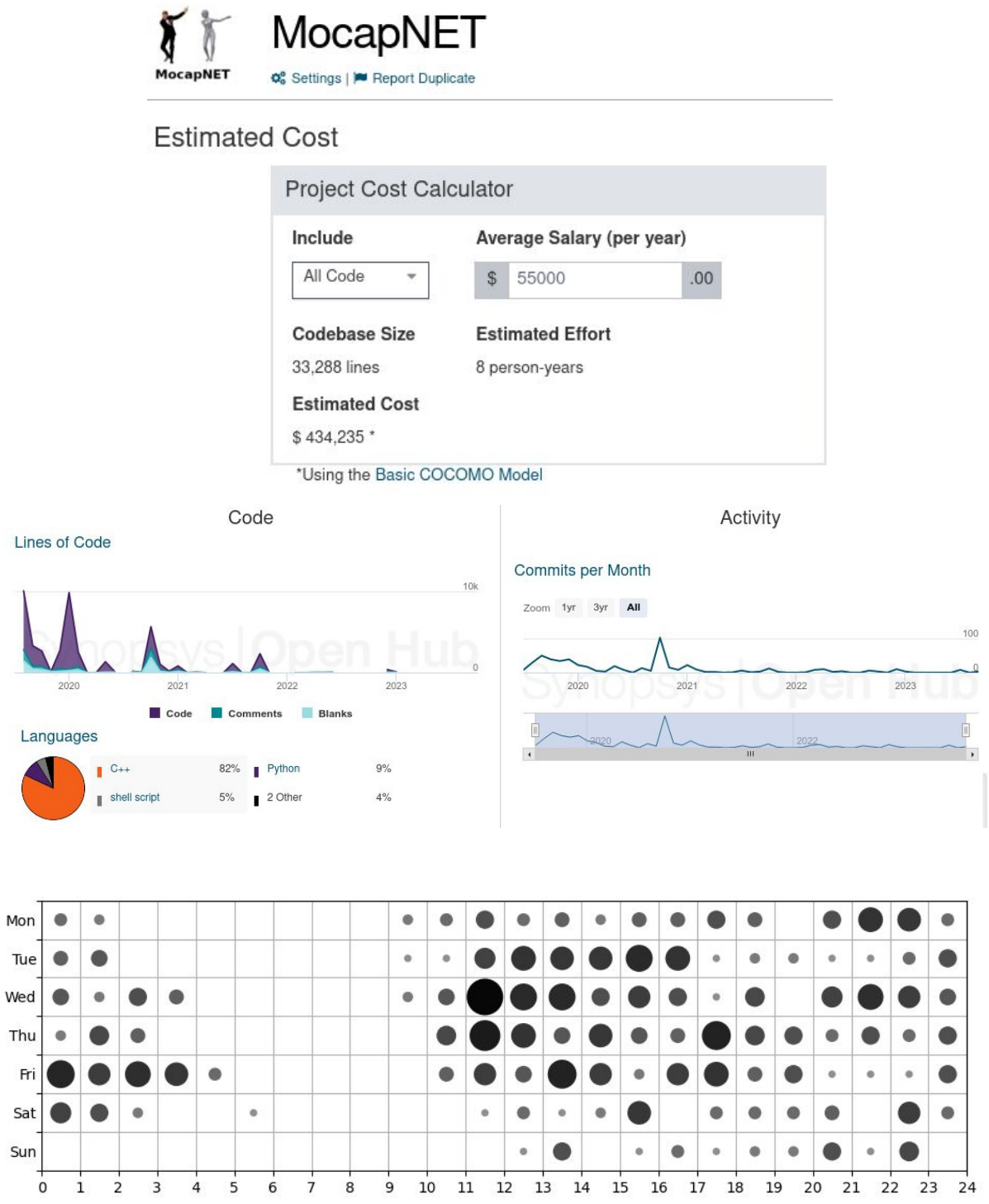


Figure 6.3: Snapshot of the Github repository metrics using Open Hub. This repository was used for hosting public evaluation snapshots after the major publications during this thesis. We corrected and answered 107 issues and questions reported by the research community after 536 commits, throughout the project development course.

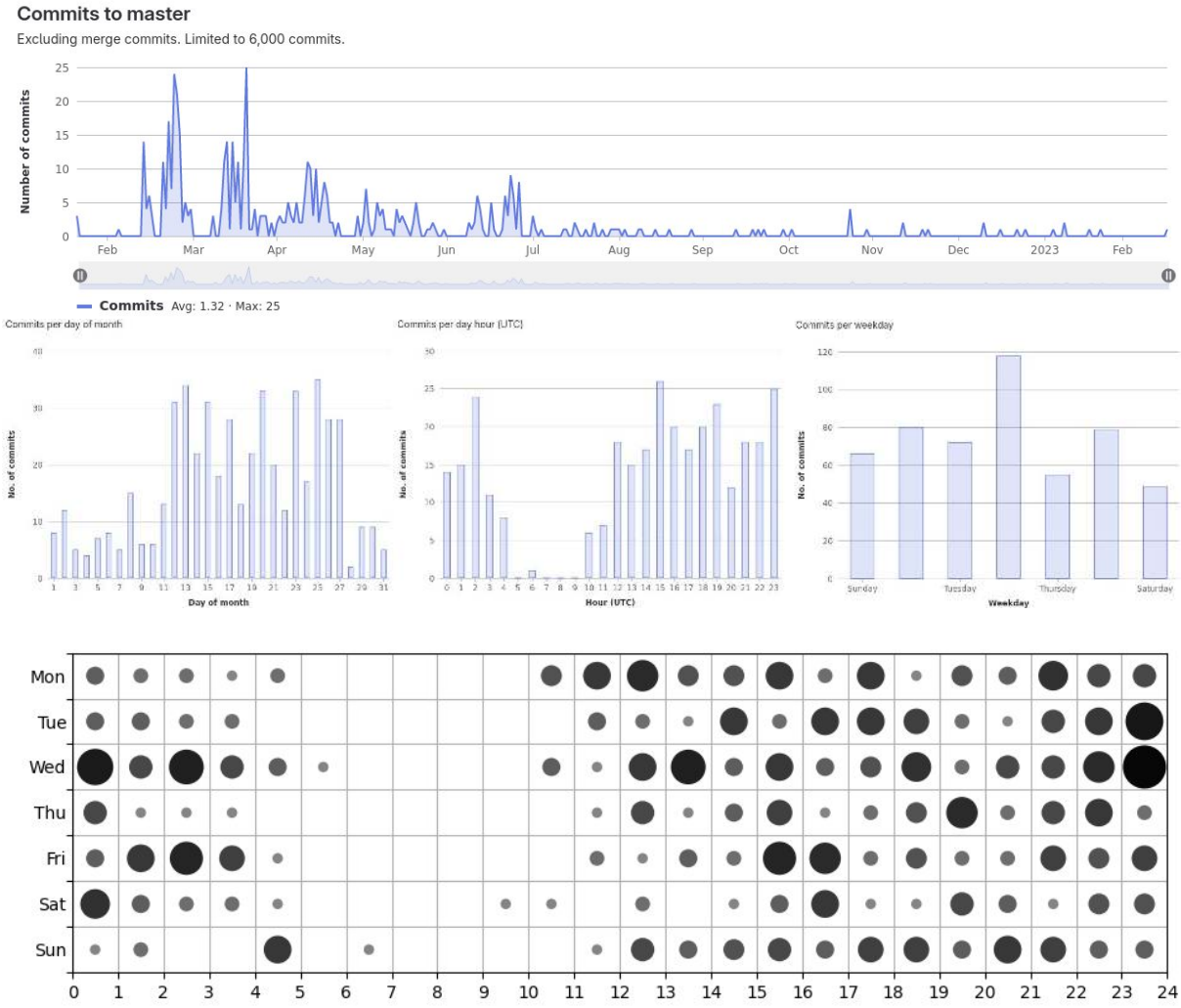


Figure 6.4: Snapshot of the Gitlab AUTO-MNET [47] repository metrics during development dedicated to the BONSAPPS project. The BonseyesAIAssetAutoMNET repository was used for the development that started at 18-2-2022. The project had 527 commits done during the course of development. Supporting this repository two other minor repositories BonseyesDataToolAutoMNET and BonseyesCMUBVH-Dataset where created hosting another 101 commits.

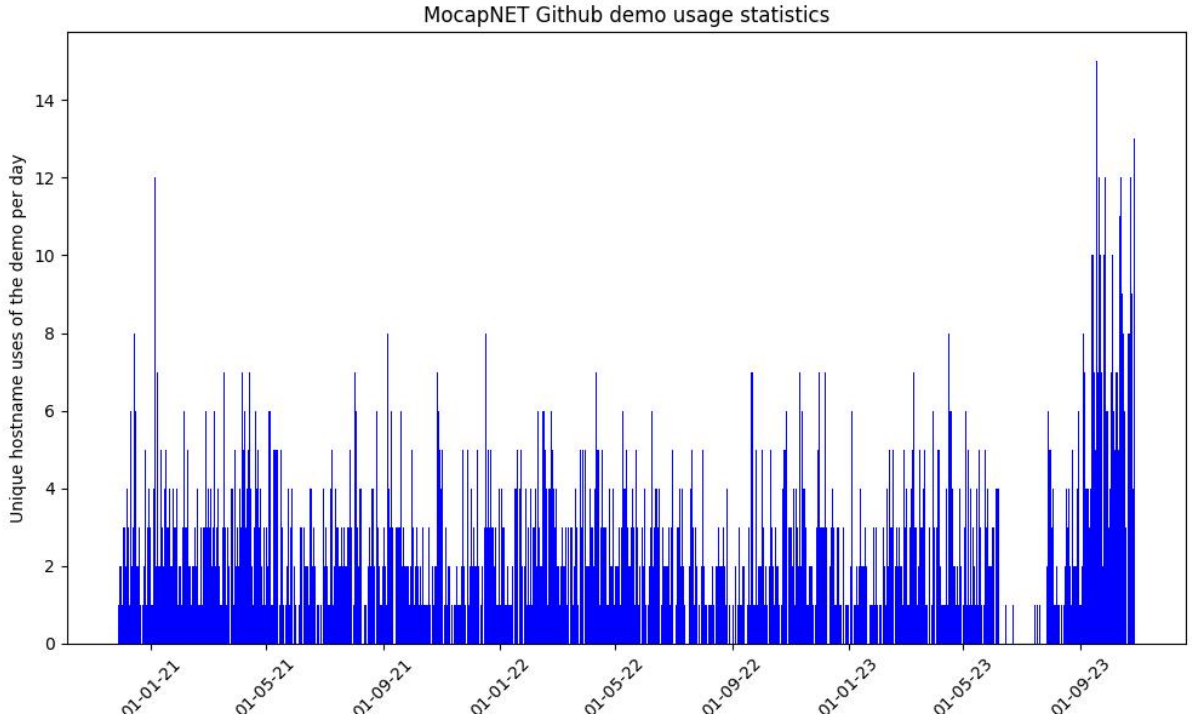


Figure 6.5: After the first MocapNET publication due to the fragmentation of different MocapNET versions, an initialization check was implemented that after a web-request prompted users to update to the latest version. Plotting the version check logs we can get a rough estimation of the usage of the project from unique hosts/IPs. The short duration with no data during the summer of 2023 coincides with network outage on the hosting machine due to a change on the firewall policy of FORTH.

6.3 Directions for Future Work and Research

Given the available time and resources this thesis investigated a variety of aspects of the 3D Total Capture problem from RGB. It also highlighted some directions for future work that could extend and improve it.

6.3.1 Appearance Encoding / Monolithic RGB to BVH MocapNETs

During the Archimedes summer school professor Katerina Fragkiadaki suggested the use of an intermediate layer embeddings from the RGB to 2D stage as additional input to the MocapNET ensemble in an MLPMixer arrangement. This however was not possible at the time due to the lack of RGB training data with BVH ground truth. Although encoding appearance cues would definitely be beneficial, especially for ambiguous 2D clouds with multiple 3D candidate solutions, the unique end-to-end architecture of MocapNETs did not allow for such a development. Given the new VICON MOCAP system available at FORTH in 2023, such a project could be possible by combining RGB, 2D, 3D and BVH angles in the same dataset. The same idea could also be facilitated using synthetic rendered data that however typically need extensive processing to translate to images acquired by real camera sensors. This was previously not easily feasible, however with

the advent of high quality generative diffusion methods [56] and proof of concepts like ControlNet [467] we can now use generative AI to render 3D skeletons from BVH ground truth and then generate high quality RGB training images thus bootstrapping a method that will target single stage regression from RGB to BVH.

A secondary, intermediate way to combine the existing architecture with appearance encoding would be to create a supplementary encoder ensemble to regress important cues like, Subject dimensions, Camera/Floor Position and Camera intrinsics from RGB once and then supply these information to the ensemble for the rest of the session at no additional computation cost. In a similar manner also having 3D depth estimations for the end-effectors of limbs would drastically improve the accuracy by better grounding the solution and removing ambivalent solutions that while having a low 2D score are not good in terms of 3D accuracy.

6.3.2 Probabilistic Regression

The MocapNET formulation at the moment is a neural network ensemble that is tasked with the regression of a single solution on a very ill-posed problem where with the same input one can define many solutions. A loss function that scores a single output value for an input that can in reality have many is architecturally flawed. Even if the value regressed is "one" of the correct ones, the Hierarchical Coordinate Descent algorithm on the fine tuning module will be probably trapped on a local minima not being able to recover the correct one. A solution to this issue would be to "Sample from the posterior" during training and have a PDF output that can encode multiple solutions propagating them to the HCD fine tuner that could also work probabilistically. This mode of operation would also allow postprocessing of Motion Capture sessions built after monitoring a stream of poses for a substantial time thus resolving ambiguity as a secondary optimization task over the global maximum probability of the tracked pose, instead on a per-frame basis like the current implementation.

6.3.3 Motion Series, Gesture / Action Recognition

The current version of the MocapNET NN ensemble performs single frame regression. Leveraging temporal cues is currently mitigated to the HCD algorithm that also examines a history of previous solutions and applies a Butterworth signal processing for temporal smoothing. However, most current state of the art in monocular 3D pose estimation methods [183,184,199] use transformer based architectures with 243 frames of input. Although these techniques are offline they offer insights on how to improve 3D accuracy by relying on multiple series of images that provide more data that reduce pose ambiguity. As seen in papers like "Hierarchical recurrent neural network for skeleton based action recognition" [468], considering that even recurrent neural networks (RNN) can model the long-term contextual information of temporal sequences they can be used for improved skeleton detection. In a similar course of work grounding the estimation task using action recognition labels or motion "sumarization" could also serve as a means to both perform gesture/action recognition task, while also providing grounding for subsequent frames possibly improving regression accuracy. Assuming that a frame has successfully been labeled as "sitting" propagating this to the next frame regression could potentially help with accuracy improvements.

6.3.4 Physics Integration, Floor and Foot Stabilization

Works like "3D Human Pose Estimation via Intuitive Physics" [469] correct implausible 3D regression results that lean, float, or penetrate the floor. Similarly works like DECO [470], InterDiff [471], PhysDiff [472],

Physics augmented auto-encoders for gait recognition [473] all show that Physics informed regression is both an important research direction and can vastly improve output results compared to methods that ignore the physics aspect of the scene. Given the sparsity of our formulation and the extensible 3D BVH armature, including 3D planes on contact surfaces to physically bound regression during HCD could be implemented in a straight forward way, while encoders for contact points could be also added in the ensemble architecture without severely altering the current architecture.

6.3.5 Extension to Garments, Fine Grained Human Pose

Although the current version of the method tackles all of the main subgroups of 3D human pose estimation, namely face, gaze, body and hands, its resolution could be increased. Increasing the dimensionality of our BVH skeleton we could provide more fine tuned output while also providing output for toes, hair groups, or even accessories like garments or jewlery, glasses etc. Works like neural haircut [474] already push the state of the art by dynamically tracking and simulating individual strands of hair, while a multitude of methods deal with motion and image synthesis with emphasis on clothes [475–478], while 3D cloth datasets and recording protocols are becoming available [479, 480].

6.3.6 MocapNET Network Architecture Improvements

MocapNET uses a relatively simple MLP architecture. In terms of neural network architectures however a variety of models have been proposed with the most prominent currently being Transformer networks [388] that have the capacity for "introspection" using their attention heads. While a transformer based back-end would be undoubtedly slower than the current one, possibly sacrificing the real-time aspect of this work, having the capacity to generate large numbers of samples attempting to use a transformer to perform regression would be an obvious direction for future research. Other promising network architectures include sum-of-product neural networks (SOPNN) [481], Assembly Calculus [482] Spiking Neural networks [483] that could be also combined with Event Cameras to build on the strengths of our low complexity model speeding up computations even more. Other improvements could include dimensiontality reduction using PCA.

6.3.7 Hierarchical Coordinate Descent

The HCD algorithm warrants deeper analysis and could be improved in multiple ways, for example by implementing mirror descent, the fast gradient method [484] or Annealed Gradient Descent with Langevin Dynamics [485].

6.3.8 Principle Component Trees and Variance-Based Tree Splitting

A part of the conducted work during this PhD research involved using dimensionality reduction techniques to reduce our neural network size and make our formulation applicable to embedded devices [20]. However as elaborated in Section 3.8 recent findings such as "any NN with any activation function can be represented as a decision tree" [26] combined with publications such as [346] and [347] provide insight into how "Fourier Features can allow networks to learn high frequency functions in low dimensional domains" [346]. Studying the representational impact of altering high frequency PCA dimensions [486] was recently even performed in the context of generative diffusion models. This spurred a pivotal question: What if, instead

of relying on traditional Multi-Layer Perceptrons (MLPs) or standard decision trees (and by extension, random forests) where splits hinge on metrics like information gain or Gini impurity, we adopted an alternative strategy? First of all apply a PCA fitting step to our data, performing dimensionality reduction keeping only a few dominant dimensions. Then as a subsequent step conduct splits starting from the most important (variance wise) dimension with a decreasing number of splits as we progress towards the final tree layer. We call this idea PCTree or Variance based tree splitting (VaTS). Preliminary indications suggest that this idea is novel and absent in existing literature, marking it as a promising area for future research.

6.3.9 Application in Other Articulated Object Domains

Our method deals with articulated armatures of torsos, arms, feet, fingers and faces of adult humans. The problem we tackle is a very high-dimensional one. Having successfully tackled it we can attempt to also perform tracking in articulated objects for infants and babies [487, 488]. 3D articulated object pose estimation in conjunction with 3D human pose tracking for Hand Object Interaction (HOI). 3D animal tracking for animals like primates, dogs, cats, birds or horses. Finally an attempt to learn Molecule configurations given their handedness constraints and public datasets with billions of molecules like QM9 [489].

Bibliography

[1] E. Britannica, "The human body." https://www.britannica.com/science/human-body, 2020.

[2] Jonothan Hui, "Understanding matrix capsules with em routing (based on hinton's capsule networks)," 2017. [Online; accessed 1-August-2023].

[3] CMU Perceptual Computing Lab, "Openpose output format specifications." https://github.com/CMU-Perceptual-Computing-Lab/openpose/blob/master/doc/output.md, 2019. [Online; accessed 9-July-2019].

[4] Z. Cao, G. Hidalgo, T. Simon, S.-E. Wei, and Y. Sheikh, "OpenPose: realtime multi-person 2D pose estimation using Part Affinity Fields," in *arXiv preprint arXiv:1812.08008*, 2018.

[5] R. Gross, I. Matthews, J. Cohn, T. Kanade, and S. Baker, "Multi-pie," *Image and vision computing*, vol. 28, no. 5, pp. 807–813, 2010.

[6] A. e. a. Qammaz, "Mocapnet: Ensemble of snn encoders for 3d human pose estimation in rgb images," in *British Machine Vision Conference (BMVC 2019)*, (Cardiff, UK), BMVA, September 2019.

[7] Wikipedia contributors, "Euclidean distance matrix — Wikipedia, the free encyclopedia," 2018. [Online; accessed 8-April-2019].

[8] A. e. a. Qammaz, "Occlusion-tolerant and personalized 3d human pose estimation in rgb images," in *IEEE International Conference on Pattern Recognition (ICPR 2020), (to appear)*, January 2021.

[9] A. e. a. Qammaz, "Towards holistic real-time human 3d pose estimation using mocapnets," in *BMVC 2021*, BMVA, November 2021.

[10] W. Chen and K. Shi, "A deep learning framework for time series classification using relative position matrix and convolutional neural network," *Neurocomputing*, vol. 359, pp. 384–394, 2019.

[11] I. Mitiche, G. Morison, A. Nesbitt, M. Hughes-Narborough, B. G. Stewart, and P. Boreham, "Imaging time series for the classification of emi discharge sources," *Sensors*, vol. 18, no. 9, p. 3098, 2018.

[12] F. Xiao, Y. Chen, and Y. Zhu, "Gadf/gasf-hog: feature extraction methods for hand movement classification from surface electromyography," *Journal of Neural Engineering*, vol. 17, no. 4, p. 046016, 2020.

[13] D. G. Lowe *et al.*, "Object recognition from local scale-invariant features.," in *iccv*, vol. 99, pp. 1150–1157, 1999.

[14] P. Fischer, A. Dosovitskiy, and T. Brox, "Descriptor matching with convolutional neural networks: a comparison to sift," *arXiv preprint arXiv:1405.5769*, 2014.

[15] D. Tsourounis, D. Kastaniotis, C. Theoharatos, A. Kazantzidis, and G. Economou, "Sift-cnn: When convolutional neural networks meet dense sift descriptors for image and sequence classification," *Journal of Imaging*, vol. 8, no. 10, p. 256, 2022.

[16] A. Q. B. Hahne, "The mocapnet bvh training dataset, based on the daz-friendly bvh release of cmu motion capture database." http://cvrlcode.ics.forth.gr/web_share/mocapnet/CMUPlusHeadMotionCapture.zip, 2020. Accessed: 2023-08-23.

[17] B. Hahne, "The daz-friendly bvh release of cmu motion capture database." https://sites.google.com/a/cgspeed.com/cgspeed/motion-capture/daz-friendly-release, 2010. Accessed: 2018-10-05.

[18] C. M. University, "Cmu graphics lab motion capture database." http://mocap.cs.cmu.edu/, 2003. Accessed: 2017-06-01.

[19] I. M. Sobol', "On the distribution of points in a cube and the approximate evaluation of integrals," *Zhurnal Vychislitel'noi Matematiki i Matematicheskoi Fiziki*, vol. 7, no. 4, pp. 784–802, 1967.

[20] A. Qammaz and A. Argyros, "Compacting mocapnet-based 3d human pose estimation via dimensionality reduction," in *International Conference on Pervasive Technologies Related to Assistive Environments (PETRA 2023)*, (Corfu, Greece), ACM, July 2023.

[21] Blender Online Community, "Blender - a 3d modelling and rendering package." http://www.blender.org, 2019.

[22] A. Qammaz, "Blender plugin for mocapnet/makehuman facial control using bvh armatures." https://github.com/FORTH-ModelBasedTracker/MocapNET/blob/mnet4/src/python/blender/blender_face.py, 2023. [Online; accessed 9-August-2023].

[23] MakeHuman Community, "Makehuman." http://www.makehumancommunity.org/, 2019. [Online; accessed 8-April-2019].

[24] M. I. Jordan and R. A. Jacobs, "Hierarchical mixtures of experts and the em algorithm," *Neural Computation*, vol. 6, no. 2, pp. 181–214, 1994.

[25] S. Theodoridis, *Machine learning: a Bayesian and optimization perspective*. Academic press, 2015.

[26] C. Aytekin, "Neural networks are decision trees," 2022.

[27] D. Misra, "Mish: A self regularized non-monotonic activation function," *arXiv preprint arXiv:1908.08681*, 2019.

[28] G. Klambauer, T. Unterthiner, A. Mayr, and S. Hochreiter, "Self-normalizing neural networks," in *Advances in neural information processing systems*, pp. 971–980, 2017.

[29] P. Ramachandran, B. Zoph, and Q. V. Le, "Searching for activation functions," *arXiv preprint arXiv:1710.05941*, 2017.

[30] S. Bianco, R. Cadene, L. Celona, and P. Napoletano, "Benchmark analysis of representative deep neural network architectures," *IEEE access*, vol. 6, pp. 64270–64277, 2018.

[31] A. Qammaz and A. Argyros, "A unified approach for occlusion tolerant 3d facial pose capture and gaze estimation using mocapnets," in *In Proceedings, Analysis and Modeling of Faces and Gestures (AMFG) in conjunction with International Conference on Computer Vision (ICCV 2023)*, (Paris, France), IEEE, October 2023.

[32] J. Simon, "Large language models: A new moore's law?." `https://huggingface.co/blog/large-language-models`, 2021.

[33] L. Van der Maaten and G. Hinton, "Visualizing data using t-sne.," *Journal of machine learning research*, vol. 9, no. 11, 2008.

[34] V. Bazarevsky, I. Grishchenko, K. Raveendran, T. Zhu, F. Zhang, and M. Grundmann, "Blazepose: On-device real-time body pose tracking," *arXiv preprint arXiv:2006.10204*, 2020.

[35] M. Turk and A. Pentland, "Eigenfaces for recognition," *Journal of cognitive neuroscience*, vol. 3, no. 1, pp. 71–86, 1991.

[36] D. Xiang, H. Joo, and Y. Sheikh, "Monocular total capture: Posing face, body, and hands in the wild," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10965–10974, 2019.

[37] Yang Song, "Generative modeling by estimating gradients of the data distribution." `https://yang-song.net/blog/2021/score/`, 2021. [Online; accessed 9-September-2023].

[38] C. Ionescu, D. Papava, V. Olaru, and C. Sminchisescu, "Human3.6m: Large scale datasets and predictive methods for 3d human sensing in natural environments," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2014.

[39] C. Ionescu, D. Papava, V. Olaru, and C. Sminchisescu, "Human3.6m: Large scale datasets and predictive methods for 3d human sensing in natural environments," *IEEE transactions on pattern analysis and machine intelligence*, vol. 36, no. 7, pp. 1325–1339, 2014.

[40] Y. Cai, L. Ge, J. Cai, and J. Yuan, "Weakly-supervised 3d hand pose estimation from monocular rgb images," in *Proceedings of the European conference on computer vision (ECCV)*, pp. 666–682, 2018.

[41] C. Zimmermann and T. Brox, "Learning to estimate 3d hand pose from single rgb images," in *Proceedings of the IEEE ICCV*, pp. 4903–4911, 2017.

[42] J. C. Gower, "Generalized procrustes analysis," *Psychometrika*, vol. 40, no. 1, pp. 33–51, 1975.

[43] B. Smith, Q. Yin, S. Feiner, and S. Nayar, "Gaze Locking: Passive Eye Contact Detection for Human?Object Interaction," in *ACM Symposium on User Interface Software and Technology (UIST)*, pp. 271–280, Oct 2013.

[44] S. Johnson and M. Everingham, "Clustered pose and nonlinear appearance models for human pose estimation.," in *bmvc*, vol. 2, p. 5, Aberystwyth, UK, 2010.

[45] C. Sagonas, E. Antonakos, G. Tzimiropoulos, S. Zafeiriou, and M. Pantic, "300 faces in-the-wild challenge: Database and results," *Image and vision computing*, vol. 47, pp. 3–18, 2016.

[46] Google, "Mediapipe iris detection." `https://solutions.mediapipe.dev/iris`, 2023. [Online; accessed 13-June-2023].

[47] A. Qammaz, "Gitlab repository for auto-mnet, during the development of the bonsapps h2020 project." `https://gitlab.com/bonseyes/bonsapps/1st-support-programme/automotive/body-part-tracking/auto-mnet/bonseyesaiassetautomnet`, 2022.

[48] A. Qammaz, "Omni-mocapnet smart home demo video." `https://www.youtube.com/watch?v=zQsOfPBX8RM`, 2020.

[49] J. Lee, J. Hwangbo, and M. Hutter, "Robust recovery controller for a quadrupedal robot using deep reinforcement learning," *arXiv preprint arXiv:1901.07517*, 2019.

[50] J. Hwangbo, J. Lee, A. Dosovitskiy, D. Bellicoso, V. Tsounis, V. Koltun, and M. Hutter, "Learning agile and dynamic motor skills for legged robots," *Science Robotics*, vol. 4, no. 26, p. eaau5872, 2019.

[51] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning quadrupedal locomotion over challenging terrain," *Science robotics*, vol. 5, no. 47, p. eabc5986, 2020.

[52] V. Tsounis, M. Alge, J. Lee, F. Farshidian, and M. Hutter, "Deepgait: Planning and control of quadrupedal gaits using deep reinforcement learning," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3699–3706, 2020.

[53] F. Shi, T. Homberger, J. Lee, T. Miki, M. Zhao, F. Farshidian, K. Okada, M. Inaba, and M. Hutter, "Circus anymal: A quadruped learning dexterous manipulation with its limbs," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2316–2323, IEEE, 2021.

[54] P. W. Code, "Human3.6m leaderboard." `https://www.paperswithcode.com/sota/3d-human-pose-estimation-on-human36m`, 2020.

[55] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, *et al.*, "Llama: Open and efficient foundation language models," *arXiv preprint arXiv:2302.13971*, 2023.

[56] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-resolution image synthesis with latent diffusion models," 2021.

[57] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, *et al.*, "Learning transferable visual models from natural language supervision," in *International conference on machine learning*, pp. 8748–8763, PMLR, 2021.

[58] M. Caron, H. Touvron, I. Misra, H. Jégou, J. Mairal, P. Bojanowski, and A. Joulin, "Emerging properties in self-supervised vision transformers," in *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 9650–9660, 2021.

[59] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, "Nerf: Representing scenes as neural radiance fields for view synthesis," *Communications of the ACM*, vol. 65, no. 1, pp. 99–106, 2021.

[60] G. Pavlakos, V. Choutas, N. Ghorbani, T. Bolkart, A. A. Osman, D. Tzionas, and M. J. Black, "Expressive body capture: 3d hands, face, and body from a single image," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10975–10985, 2019.

[61] Y. Zhou, M. Habermann, I. Habibie, A. Tewari, C. Theobalt, and F. Xu, "Monocular real-time full body capture with inter-part correlations," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4811–4822, 2021.

[62] Y. Rong, T. Shiratori, and H. Joo, "Frankmocap: Fast monocular 3d hand and body motion capture by regression and integration," *arXiv preprint arXiv:2008.08324*, 2020.

[63] L. Xu, S. Jin, W. Liu, C. Qian, W. Ouyang, P. Luo, and X. Wang, "Zoomnas: searching for whole-body human pose estimation in the wild," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

[64] M.-P. Forte, P. Kulits, C.-H. P. Huang, V. Choutas, D. Tzionas, K. J. Kuchenbecker, and M. J. Black, "Reconstructing signing avatars from video using linguistic priors," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12791–12801, 2023.

[65] A. Qammaz, "Mocapnet github repository." `https://github.com/FORTH-ModelBasedTracker/MocapNET`, 2019. [Online; accessed 11-July-2019].

[66] Deep Motion, "Deep motion, create 3d animations from video using ai." `https://www.deepmotion.com/`, 2023. [Online; accessed 1-August-2023].

[67] RADiCAL Solutions, Inc., "Radical motion," 2023. [Online; accessed 1-August-2023].

[68] PLASK, "Plask ai-powered mocap animation tool," 2023. [Online; accessed 1-August-2023].

[69] RUSH, "Rush motion capture," 2023. [Online; accessed 1-August-2023].

[70] Adobe, "Mixamo, get animated," 2023. [Online; accessed 1-August-2023].

[71] G. D. e. a. Anthony Brohan, "Rt-2: Vision-language-action models transfer web knowledge to robotic control," 2023.

[72] Wikipedia contributors, "Chatgpt — Wikipedia, the free encyclopedia," 2023. [Online; accessed 31-July-2023].

[73] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, pp. 1097–1105, 2012.

[74] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 779–788, 2016.

[75] X. He and L. Deng, "Deep learning for image-to-text generation: A technical overview," *IEEE Signal Processing Magazine*, vol. 34, no. 6, pp. 109–116, 2017.

[76] B. e. a. Wandt, "Elepose: Unsupervised 3d human pose estimation by predicting camera elevation and learning normalizing flows on 2d poses," 2021.

[77] M. Meredith, S. Maddock, *et al.*, "Motion capture file formats explained," *Department of Computer Science, University of Sheffield*, vol. 211, pp. 241–244, 2001.

[78] Google, "Mediapipe holistic detection." `https://github.com/google/mediapipe/blob/master/docs/solutions/holistic.md`, 2023. [Online; accessed 13-June-2023].

[79] J. M. Rehg and T. Kanade, "Visual tracking of high dof articulated structures: an application to human hand tracking," in *Computer Vision—ECCV'94: Third European Conference on Computer Vision Stockholm, Sweden, May 2–6 1994 Proceedings, Volume II 3*, pp. 35–46, Springer, 1994.

[80] P. J. Besl and N. D. McKay, "Method for registration of 3-d shapes," in *Sensor fusion IV: control paradigms and data structures*, vol. 1611, pp. 586–606, International Society for Optics and Photonics, 1992.

[81] D. Terzopoulos and D. Metaxas, "Dynamic 3 d models with local and global deformations: deformable superquadrics," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 13, no. 7, pp. 703–714, 1991.

[82] M. J. Black and Y. Yacoob, "Tracking and recognizing rigid and non-rigid facial motions using local parametric models of image motion," in *Proceedings of IEEE international conference on computer vision*, pp. 374–381, IEEE, 1995.

[83] Y. Zhang, Z. Li, L. An, M. Li, T. Yu, and Y. Liu, "Lightweight multi-person total motion capture using sparse multi-view cameras," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 5560–5569, 2021.

[84] PixCap Pte Ltd, "Pixcap ai mocap," 2023. [Online; accessed 1-August-2023].

[85] Move.AI, "Move.ai mocap," 2023. [Online; accessed 1-August-2023].

[86] J. K. Aggarwal and Q. Cai, "Human motion analysis: A review," *Computer vision and image understanding*, vol. 73, no. 3, pp. 428–440, 1999.

[87] D. M. Gavrila, "The visual analysis of human movement: A survey," *Computer vision and image understanding*, vol. 73, no. 1, pp. 82–98, 1999.

[88] T. B. Moeslund and E. Granum, "A survey of computer vision-based human motion capture," *Computer vision and image understanding*, vol. 81, no. 3, pp. 231–268, 2001.

[89] W. Hu, T. Tan, L. Wang, and S. Maybank, "A survey on visual surveillance of object motion and behaviors," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 34, no. 3, pp. 334–352, 2004.

[90] T. B. Moeslund, A. Hilton, and V. Krüger, "A survey of advances in vision-based human motion capture and analysis," *Computer vision and image understanding*, vol. 104, no. 2-3, pp. 90–126, 2006.

[91] C. Sminchisescu, "3d human motion analysis in monocular video: techniques and challenges," in *Human Motion*, pp. 185–211, Springer, 2008.

[92] X. Ji and H. Liu, "Advances in view-invariant human motion analysis: a review," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 40, no. 1, pp. 13–24, 2009.

[93] T. B. Moeslund, A. Hilton, V. Krüger, and L. Sigal, *Visual analysis of humans*. Springer, 2011.

[94] M. B. Holte, C. Tran, M. M. Trivedi, and T. B. Moeslund, "Human pose estimation and activity recognition from multi-view videos: Comparative explorations of recent developments," *IEEE Journal of selected topics in signal processing*, vol. 6, no. 5, pp. 538–552, 2012.

[95] L. Chen, H. Wei, and J. Ferryman, "A survey of human motion analysis using depth imagery," *Pattern Recognition Letters*, vol. 34, no. 15, pp. 1995–2006, 2013.

[96] X. Perez-Sala, S. Escalera, C. Angulo, and J. Gonzalez, "A survey on model based approaches for 2d and 3d visual human pose recovery," *Sensors*, vol. 14, no. 3, pp. 4189–4210, 2014.

[97] W. Gong, X. Zhang, J. Gonzàlez, A. Sobral, T. Bouwmans, C. Tu, and E.-h. Zahzah, "Human pose estimation from monocular images: A comprehensive survey," *Sensors*, vol. 16, no. 12, p. 1966, 2016.

[98] N. Sarafianos, B. Boteanu, B. Ionescu, and I. A. Kakadiaris, "3d human pose estimation: A review of the literature and analysis of covariates," *Computer Vision and Image Understanding*, vol. 152, pp. 1–20, 2016.

[99] P. Wang, W. Li, P. Ogunbona, J. Wan, and S. Escalera, "Rgb-d-based human motion recognition with deep learning: A survey," *Computer Vision and Image Understanding*, vol. 171, pp. 118–139, 2018.

[100] Y. Chen, Y. Tian, and M. He, "Monocular human pose estimation: A survey of deep learning-based methods," *Computer Vision and Image Understanding*, p. 102897, 2020.

[101] T. L. Munea, Y. Z. Jembre, H. T. Weldegebriel, L. Chen, C. Huang, and C. Yang, "The progress of human pose estimation: a survey and taxonomy of models applied in 2d human pose estimation," *IEEE Access*, vol. 8, pp. 133330–133348, 2020.

[102] C. Wang, F. Zhang, and S. S. Ge, "A comprehensive survey on 2d multi-person pose estimation methods," *Engineering Applications of Artificial Intelligence*, vol. 102, p. 104260, 2021.

[103] M. Dantone, J. Gall, C. Leistner, and L. Van Gool, "Human pose estimation using body parts dependent joint regressors," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3041–3048, 2013.

[104] G. Fanelli, M. Dantone, J. Gall, A. Fossati, and L. Van Gool, "Random forests for real time 3d face analysis," *International journal of computer vision*, vol. 101, no. 3, pp. 437–458, 2013.

[105] X. Chen and A. L. Yuille, "Articulated pose estimation by a graphical model with image dependent pairwise relations," in *Advances in neural information processing systems*, pp. 1736–1744, 2014.

[106] G. Gkioxari, B. Hariharan, R. Girshick, and J. Malik, "R-cnns for pose estimation and action detection," *arXiv preprint arXiv:1406.5212*, 2014.

[107] A. Toshev and C. Szegedy, "Deeppose: Human pose estimation via deep neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1653–1660, 2014.

[108] A. Newell, K. Yang, and J. Deng, "Stacked hourglass networks for human pose estimation," in *European Conference on Computer Vision*, pp. 483–499, Springer, 2016.

[109] S.-E. Wei, V. Ramakrishna, T. Kanade, and Y. Sheikh, "Convolutional pose machines," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4724–4732, 2016.

[110] Z. Cao, T. Simon, S.-E. Wei, and Y. Sheikh, "Realtime multi-person 2d pose estimation using part affinity fields," *arXiv preprint arXiv:1611.08050*, 2016.

[111] L. Pishchulin, E. Insafutdinov, S. Tang, B. Andres, M. Andriluka, P. V. Gehler, and B. Schiele, "Deepcut: Joint subset partition and labeling for multi person pose estimation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4929–4937, 2016.

[112] A. Bulat and G. Tzimiropoulos, "Human pose estimation via convolutional part heatmap regression," in *European Conference on Computer Vision*, pp. 717–732, Springer, 2016.

[113] J. Carreira, P. Agrawal, K. Fragkiadaki, and J. Malik, "Human pose estimation with iterative error feedback," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4733–4742, 2016.

[114] X. Yu, F. Zhou, and M. Chandraker, "Deep deformation network for object landmark localization," in *European Conference on Computer Vision*, pp. 52–70, Springer, 2016.

[115] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural computation*, vol. 1, no. 4, pp. 541–551, 1989.

[116] J. J. Tompson, A. Jain, Y. LeCun, and C. Bregler, "Joint training of a convolutional network and a graphical model for human pose estimation," in *Advances in neural information processing systems*, pp. 1799–1807, 2014.

[117] C. Sun, A. Shrivastava, S. Singh, and A. Gupta, "Revisiting unreasonable effectiveness of data in deep learning era," in *Proceedings of the IEEE international conference on computer vision*, pp. 843–852, 2017.

[118] D. C. Luvizon, H. Tabia, and D. Picard, "Learning features combination for human action recognition from skeleton sequences," *Pattern Recognition Letters*, vol. 99, pp. 13–20, 2017.

[119] W. Yang, W. Ouyang, H. Li, and X. Wang, "End-to-end learning of deformable mixture of parts and deep convolutional neural networks for human pose estimation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3073–3082, 2016.

[120] X. Chu, W. Yang, W. Ouyang, C. Ma, A. L. Yuille, and X. Wang, "Multi-context attention for human pose estimation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1831–1840, 2017.

[121] W. Yang, S. Li, W. Ouyang, H. Li, and X. Wang, "Learning feature pyramids for human pose estimation," in *proceedings of the IEEE international conference on computer vision*, pp. 1281–1290, 2017.

[122] Y. Chen, C. Shen, X.-S. Wei, L. Liu, and J. Yang, "Adversarial posenet: A structure-aware convolutional network for human pose estimation," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1212–1221, 2017.

[123] X. Peng, Z. Tang, F. Yang, R. S. Feris, and D. Metaxas, "Jointly optimize data augmentation and network training: Adversarial data augmentation in human pose estimation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2226–2234, 2018.

[124] L. Ke, M.-C. Chang, H. Qi, and S. Lyu, "Multi-scale structure-aware network for human pose estimation," in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 713–728, 2018.

[125] Q. Wang, Z. Teng, J. Xing, J. Gao, W. Hu, and S. Maybank, "Learning attentions: residual attentional siamese network for high performance online visual tracking," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4854–4863, 2018.

[126] K. Sun, B. Xiao, D. Liu, and J. Wang, "Deep high-resolution representation learning for human pose estimation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5693–5703, 2019.

[127] W. Tang and Y. Wu, "Does learning specific features for related parts help human pose estimation?," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1107–1116, 2019.

[128] B. Cheng, B. Xiao, J. Wang, H. Shi, T. S. Huang, and L. Zhang, "Higherhrnet: Scale-aware representation learning for bottom-up human pose estimation," 2019.

[129] Y. Cai, Z. Wang, Z. Luo, B. Yin, A. Du, H. Wang, X. Zhang, X. Zhou, E. Zhou, and J. Sun, "Learning delicate local representations for multi-person pose estimation," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part III 16*, pp. 455–472, Springer, 2020.

[130] C. Yu, B. Xiao, C. Gao, L. Yuan, L. Zhang, N. Sang, and J. Wang, "Lite-hrnet: A lightweight high-resolution network," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10440–10450, 2021.

[131] Y. Ci, Y. Wang, M. Chen, S. Tang, L. Bai, F. Zhu, R. Zhao, F. Yu, D. Qi, and W. Ouyang, "Unihcp: A unified model for human-centric perceptions," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 17840–17852, 2023.

[132] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[133] Z. Geng, C. Wang, Y. Wei, Z. Liu, H. Li, and H. Hu, "Human pose as compositional tokens," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 660–671, 2023.

[134] M. Jones and P. Viola, "Fast multi-view face detection," *Mitsubishi Electric Research Lab TR-20003-96*, vol. 3, no. 14, p. 2, 2003.

[135] M. Eichner and V. Ferrari, "Appearance sharing for collective human pose estimation," in *Asian Conference on Computer Vision*, pp. 138–151, Springer, 2012.

[136] A. Nibali, Z. He, S. Morgan, and L. Prendergast, "Numerical coordinate regression with convolutional neural networks," *arXiv preprint arXiv:1801.07372*, 2018.

[137] X. Sun, J. Shang, S. Liang, and Y. Wei, "Compositional human pose regression," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2602–2611, 2017.

[138] R. Poppe, "Vision-based human motion analysis: An overview," *Computer vision and image understanding*, vol. 108, no. 1-2, pp. 4–18, 2007.

[139] V. Ramakrishna, D. Munoz, M. Hebert, J. A. Bagnell, and Y. Sheikh, "Pose machines: Articulated pose estimation via inference machines," in *European Conference on Computer Vision*, pp. 33–47, Springer, 2014.

[140] A. Jain, J. Tompson, M. Andriluka, G. W. Taylor, and C. Bregler, "Learning human pose estimation features with convolutional networks," *arXiv preprint arXiv:1312.7302*, 2013.

[141] U. Iqbal and J. Gall, "Multi-person pose estimation with local joint-to-person associations," in *European Conference on Computer Vision*, pp. 627–642, Springer, 2016.

[142] H.-S. Fang, S. Xie, Y.-W. Tai, and C. Lu, "Rmpe: Regional multi-person pose estimation," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2334–2343, 2017.

[143] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 4, pp. 834–848, 2017.

[144] S. Huang, M. Gong, and D. Tao, "A coarse-fine network for keypoint localization," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 3028–3037, 2017.

[145] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *Proceedings of the IEEE international conference on computer vision*, pp. 2961–2969, 2017.

[146] B. Xiao, H. Wu, and Y. Wei, "Simple baselines for human pose estimation and tracking," in *Proceedings of the European conference on computer vision (ECCV)*, pp. 466–481, 2018.

[147] Y. Chen, Z. Wang, Y. Peng, Z. Zhang, G. Yu, and J. Sun, "Cascaded pyramid network for multi-person pose estimation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7103–7112, 2018.

[148] G. Moon, J. Y. Chang, and K. M. Lee, "Posefix: Model-agnostic general human pose refinement network," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7773–7781, 2019.

[149] G. Ning, J. Pei, and H. Huang, "Lighttrack: A generic framework for online top-down human pose tracking," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pp. 1034–1035, 2020.

[150] Z. Kan, S. Chen, Z. Li, and Z. He, "Self-constrained inference optimization on structural groups for human pose estimation," in *European Conference on Computer Vision*, pp. 729–745, Springer, 2022.

[151] E. Insafutdinov, L. Pishchulin, B. Andres, M. Andriluka, and B. Schiele, "Deepercut: A deeper, stronger, and faster multi-person pose estimation model," in *European Conference on Computer Vision*, pp. 34–50, Springer, 2016.

[152] A. Newell, Z. Huang, and J. Deng, "Associative embedding: End-to-end learning for joint detection and grouping," in *Advances in neural information processing systems*, pp. 2277–2287, 2017.

[153] X. Nie, J. Feng, J. Xing, and S. Yan, "Pose partition networks for multi-person pose estimation," in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 684–699, 2018.

[154] G. Papandreou, T. Zhu, L.-C. Chen, S. Gidaris, J. Tompson, and K. Murphy, "Personlab: Person pose estimation and instance segmentation with a bottom-up, part-based, geometric embedding model," in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 269–286, 2018.

[155] M. Kocabas, S. Karagoz, and E. Akbas, "Multiposenet: Fast multi-person pose estimation using pose residual network," in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 417–433, 2018.

[156] S. Kreiss, L. Bertoni, and A. Alahi, "Pifpaf: Composite fields for human pose estimation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 11977–11986, 2019.

[157] Z. Tian, C. Shen, H. Chen, and T. He, "Fcos: Fully convolutional one-stage object detection," in *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 9627–9636, 2019.

[158] J. Li, W. Su, and Z. Wang, "Simple pose: Rethinking and improving a bottom-up approach for multi-person pose estimation," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 34, pp. 11354–11361, 2020.

[159] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, pp. 91–99, 2015.

[160] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2117–2125, 2017.

[161] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[162] Oxford Metrics, "Vicon motion capture system." `https://www.vicon.com/`, 2019. [Online; accessed 8-April-2019].

[163] D. E. Worrall, S. J. Garbin, D. Turmukhambetov, and G. J. Brostow, "Harmonic networks: Deep translation and rotation equivariance," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5028–5037, 2017.

[164] Y. Zhou, C. Barnes, J. Lu, J. Yang, and H. Li, "On the continuity of rotation representations in neural networks," *arXiv preprint arXiv:1812.07035*, 2018.

[165] B. Wandt, H. Ackermann, and B. Rosenhahn, "A kinematic chain space for monocular motion capture," in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 0–0, 2018.

[166] S. Sabour, N. Frosst, and G. E. Hinton, "Dynamic routing between capsules," in *Advances in neural information processing systems*, pp. 3856–3866, 2017.

[167] S. Li and A. B. Chan, "3d human pose estimation from monocular images with deep convolutional neural network," in *Asian Conference on Computer Vision*, pp. 332–347, Springer, 2014.

[168] S. Li, W. Zhang, and A. B. Chan, "Maximum-margin structured learning with deep networks for 3d human pose estimation," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2848–2856, 2015.

[169] B. Tekin, I. Katircioglu, M. Salzmann, V. Lepetit, and P. Fua, "Structured prediction of 3d human pose with deep neural networks," *arXiv preprint arXiv:1605.05180*, 2016.

[170] B. Tekin, P. Márquez-Neila, M. Salzmann, and P. Fua, "Learning to fuse 2d and 3d image cues for monocular body pose estimation," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 3941–3950, 2017.

[171] C.-H. Chen and D. Ramanan, "3d human pose estimation= 2d pose estimation+ matching," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7035–7043, 2017.

[172] F. Moreno-Noguer, "3d human pose estimation from a single image via distance matrix regression," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2823–2832, 2017.

[173] G. Pavlakos, X. Zhou, K. G. Derpanis, and K. Daniilidis, "Coarse-to-fine volumetric prediction for single-image 3d human pose," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7025–7034, 2017.

[174] X. Zhou, Q. Huang, X. Sun, X. Xue, and Y. Wei, "Towards 3d human pose estimation in the wild: a weakly-supervised approach," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 398–407, 2017.

[175] J. Martinez, R. Hossain, J. Romero, and J. J. Little, "A simple yet effective baseline for 3d human pose estimation," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2640–2649, 2017.

[176] W. Yang, W. Ouyang, X. Wang, J. Ren, H. Li, and X. Wang, "3d human pose estimation in the wild by adversarial learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5255–5264, 2018.

[177] G. Pavlakos, L. Zhu, X. Zhou, and K. Daniilidis, "Learning to estimate 3d human pose and shape from a single color image," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 459–468, 2018.

[178] X. Sun, B. Xiao, F. Wei, S. Liang, and Y. Wei, "Integral human pose regression," in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 529–545, 2018.

[179] C. Li and G. H. Lee, "Generating multiple hypotheses for 3d human pose estimation with mixture density network," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 9887–9895, 2019.

[180] K. Zhou, X. Han, N. Jiang, K. Jia, and J. Lu, "Hemlets pose: Learning part-centric heatmap triplets for accurate 3d human pose estimation," in *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 2344–2353, 2019.

[181] C. Zheng, S. Zhu, M. Mendieta, T. Yang, C. Chen, and Z. Ding, "3d human pose estimation with spatial and temporal transformers," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 11656–11665, 2021.

[182] W. Shan, Z. Liu, X. Zhang, S. Wang, S. Ma, and W. Gao, "P-stmo: Pre-trained spatial temporal many-to-one model for 3d human pose estimation," in *European Conference on Computer Vision*, pp. 461–478, Springer, 2022.

[183] J. Zhang, Z. Tu, J. Yang, Y. Chen, and J. Yuan, "Mixste: Seq2seq mixed spatio-temporal encoder for 3d human pose estimation in video," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 13232–13242, 2022.

[184] S. Mehraban, V. Adeli, and B. Taati, "Motionagformer: Enhancing 3d human pose estimation with a transformer-gcnformer network," *arXiv preprint arXiv:2310.16288*, 2023.

[185] F. Bogo, A. Kanazawa, C. Lassner, P. Gehler, J. Romero, and M. J. Black, "Keep it smpl: Automatic estimation of 3d human pose and shape from a single image," in *European Conference on Computer Vision*, pp. 561–578, Springer, 2016.

[186] X. Zhou, X. Sun, W. Zhang, S. Liang, and Y. Wei, "Deep kinematic pose regression," in *European Conference on Computer Vision*, pp. 186–201, Springer, 2016.

[187] D. Mehta, S. Sridhar, O. Sotnychenko, H. Rhodin, M. Shafiei, H.-P. Seidel, W. Xu, D. Casas, and C. Theobalt, "Vnect: Real-time 3d human pose estimation with a single rgb camera," *ACM Transactions on Graphics (TOG)*, vol. 36, no. 4, pp. 1–14, 2017.

[188] D. Mehta, H. Rhodin, D. Casas, P. Fua, O. Sotnychenko, W. Xu, and C. Theobalt, "Monocular 3d human pose estimation in the wild using improved cnn supervision," in *2017 International Conference on 3D Vision (3DV)*, pp. 506–516, IEEE, 2017.

[189] V. Tan, I. Budvytis, and R. Cipolla, "Indirect deep structured learning for 3d human body shape and pose prediction," in *British Machine Vision Conference*, 2017.

[190] B. X. Nie, P. Wei, and S.-C. Zhu, "Monocular 3d human pose estimation by predicting depth on joints," in *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 3467–3475, IEEE, 2017.

[191] D. Tome, C. Russell, and L. Agapito, "Lifting from the deep: Convolutional 3d pose estimation from a single image," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2500–2509, 2017.

[192] A. Kanazawa, M. J. Black, D. W. Jacobs, and J. Malik, "End-to-end recovery of human shape and pose," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7122–7131, 2018.

[193] M. Omran, C. Lassner, G. Pons-Moll, P. Gehler, and B. Schiele, "Neural body fitting: Unifying deep learning and model based human pose and shape estimation," in *2018 International Conference on 3D Vision (3DV)*, pp. 484–494, IEEE, 2018.

[194] G. Varol, D. Ceylan, B. Russell, J. Yang, E. Yumer, I. Laptev, and C. Schmid, "Bodynet: Volumetric inference of 3d human body shapes," in *ECCV*, 2018.

[195] H. Rhodin, M. Salzmann, and P. Fua, "Unsupervised geometry-aware representation for 3d human pose estimation," in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 750–767, 2018.

[196] A. Arnab, C. Doersch, and A. Zisserman, "Exploiting temporal context for 3d human pose estimation in the wild," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3395–3404, 2019.

[197] D.-H. Hwang, S. Kim, N. Monet, H. Koike, and S. Bae, "Lightweight 3d human pose estimation network training using teacher-student learning," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 479–488, 2020.

[198] M. Shi, K. Aberman, A. Aristidou, T. Komura, D. Lischinski, D. Cohen-Or, and B. Chen, "Motionet: 3d human motion reconstruction from monocular video with skeleton consistency," *ACM Transactions on Graphics (TOG)*, vol. 40, no. 1, pp. 1–15, 2020.

[199] W. Zhu, X. Ma, Z. Liu, L. Liu, W. Wu, and Y. Wang, "Motionbert: A unified perspective on learning human motion representations," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 15085–15099, 2023.

[200] M. Sanzari, V. Ntouskos, and F. Pirri, "Bayesian image based 3d pose estimation," in *European conference on computer vision*, pp. 566–582, Springer, 2016.

[201] Y. Du, Y. Wong, Y. Liu, F. Han, Y. Gui, Z. Wang, M. Kankanhalli, and W. Geng, "Marker-less 3d human motion capture with monocular image sequence and height-maps," in *European Conference on Computer Vision*, pp. 20–36, Springer, 2016.

[202] M. F. Ghezelghieh, R. Kasturi, and S. Sarkar, "Learning camera viewpoint using cnn to improve 3d body pose estimation," in *2016 Fourth International Conference on 3D Vision (3DV)*, pp. 685–693, IEEE, 2016.

[203] G. Rogez and C. Schmid, "Mocap-guided data augmentation for 3d pose estimation in the wild," in *Advances in Neural Information Processing Systems* (D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, eds.), pp. 3108–3116, Curran Associates, Inc., 2016.

[204] G. Rogez, P. Weinzaepfel, and C. Schmid, "Lcr-net: Localization-classification-regression for human pose," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3433–3441, 2017.

[205] G. Rogez, P. Weinzaepfel, and C. Schmid, "LCR-Net++: Multi-person 2D and 3D Pose Detection in Natural Images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019.

[206] R. Alp Güler, N. Neverova, and I. Kokkinos, "Densepose: Dense human pose estimation in the wild," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7297–7306, 2018.

[207] M. Loper, N. Mahmood, J. Romero, G. Pons-Moll, and M. J. Black, "Smpl: A skinned multi-person linear model," *ACM transactions on graphics (TOG)*, vol. 34, no. 6, p. 248, 2015.

[208] A. Elhayek, E. de Aguiar, A. Jain, J. Tompson, L. Pishchulin, M. Andriluka, C. Bregler, B. Schiele, and C. Theobalt, "Efficient convnet-based marker-less motion capture in general scenes with a low number of cameras," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3810–3818, 2015.

[209] G. Pavlakos, X. Zhou, K. G. Derpanis, and K. Daniilidis, "Harvesting multiple views for marker-less 3d human pose annotations," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 6988–6997, 2017.

[210] X. Zhou, M. Zhu, S. Leonardos, K. G. Derpanis, and K. Daniilidis, "Sparseness meets deepness: 3d human pose estimation from monocular video," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4966–4975, 2016.

[211] S. Goel, G. Pavlakos, J. Rajasegaran, A. Kanazawa, and J. Malik, "Humans in 4d: Reconstructing and tracking humans with transformers," *arXiv preprint arXiv:2305.20091*, 2023.

[212] J. Kim, G.-M. Um, J. Seo, and W. Kim, "Part-attentive kinematic chain-based regressor for 3d human modeling," *Journal of Visual Communication and Image Representation*, vol. 95, p. 103881, 2023.

[213] G. Varol, J. Romero, X. Martin, N. Mahmood, M. J. Black, I. Laptev, and C. Schmid, "Learning from synthetic humans," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 109–117, 2017.

[214] W. Chen, H. Wang, Y. Li, H. Su, Z. Wang, C. Tu, D. Lischinski, D. Cohen-Or, and B. Chen, "Synthesizing training images for boosting human 3d pose estimation," in *2016 Fourth International Conference on 3D Vision (3DV)*, pp. 479–488, IEEE, 2016.

[215] A. Qammaz, D. Michel, and A. A. Argyros, "A hybrid method for 3d pose estimation of personalized human body models," in *IEEE Winter Conference on Applications of Computer Vision (WACV 2018)*, IEEE, March 2018.

[216] J. Kennedy, "Particle swarm optimization," *Encyclopedia of machine learning*, pp. 760–766, 2010.

[217] D. Mehta, S. Sridhar, O. Sotnychenko, H. Rhodin, M. Shafiei, H.-P. Seidel, W. Xu, D. Casas, and C. Theobalt, "Vnect: Real-time 3d human pose estimation with a single rgb camera," *ACM Transactions on Graphics*, vol. 36, no. 4, 2017.

[218] H.-Y. F. Tung, A. W. Harley, W. Seto, and K. Fragkiadaki, "Adversarial inverse graphics networks: Learning 2d-to-3d lifting and image-to-image translation from unpaired supervision," in *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 4364–4372, IEEE, 2017.

[219] T. Yu, Z. Zheng, Y. Zhong, J. Zhao, Q. Dai, G. Pons-Moll, and Y. Liu, "Simulcap: Single-view human performance capture with cloth simulation," *arXiv preprint arXiv:1903.06323*, 2019.

[220] C. Hong, J. Yu, J. Wan, D. Tao, and M. Wang, "Multimodal deep autoencoder for human pose recovery," *IEEE Transactions on Image Processing*, vol. 24, no. 12, pp. 5659–5670, 2015.

[221] X. B. Peng, P. Abbeel, S. Levine, and M. van de Panne, "Deepmimic: Example-guided deep reinforcement learning of physics-based character skills," *ACM Transactions on Graphics (TOG)*, vol. 37, no. 4, p. 143, 2018.

[222] R. Tous, J. Nin, and L. Igual, "Human pose completion in partial body camera shots," *Journal of Experimental & Theoretical Artificial Intelligence*, pp. 1–11, 2023.

[223] B. Tekin, X. Sun, X. Wang, V. Lepetit, and P. Fua, "Predicting people's 3d poses from short sequences," *arXiv preprint arXiv:1504.08200*, vol. 2, no. 5, p. 6, 2015.

[224] B. Tekin, A. Rozantsev, V. Lepetit, and P. Fua, "Direct prediction of 3d body poses from motion compensated sequences," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 991–1000, 2016.

[225] B. Tekin, P. Márquez-Neila, M. Salzmann, and P. Fua, "Fusing 2d uncertainty and 3d cues for monocular body pose estimation," *arXiv preprint arXiv:1611.05708*, vol. 2, no. 3, 2016.

[226] X. Zhou, Q. Wan, W. Zhang, X. Xue, and Y. Wei, "Model-based deep hand pose estimation," *arXiv preprint arXiv:1606.06854*, 2016.

[227] A. Zanfir, E. Marinoiu, and C. Sminchisescu, "Monocular 3d pose and shape estimation of multiple people in natural scenes-the importance of multiple scene constraints," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2148–2157, 2018.

[228] A.-I. Popa, M. Zanfir, and C. Sminchisescu, "Deep multitask architecture for integrated 2d and 3d human sensing," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6289–6298, 2017.

[229] D. Mehta, O. Sotnychenko, F. Mueller, W. Xu, M. Elgharib, P. Fua, H.-P. Seidel, H. Rhodin, G. Pons-Moll, and C. Theobalt, "Xnect: Real-time multi-person 3d human pose estimation with a single rgb camera," *arXiv preprint arXiv:1907.00837*, 2019.

[230] D. Mehta, O. Sotnychenko, F. Mueller, W. Xu, S. Sridhar, G. Pons-Moll, and C. Theobalt, "Single-shot multi-person 3d pose estimation from monocular rgb," in *2018 International Conference on 3D Vision (3DV)*, pp. 120–130, IEEE, 2018.

[231] W. Jiang, N. Kolotouros, G. Pavlakos, X. Zhou, and K. Daniilidis, "Coherent reconstruction of multiple humans from a single image," in *CVPR*, 2020.

[232] Y. Guo, Z. Li, Z. Li, X. Du, S. Quan, and Y. Xu, "Pop-net: Pose over parts network for multi-person 3d pose estimation from a depth image," *arXiv preprint arXiv:2012.06734*, 2020.

[233] N. D. Reddy, L. Guigues, L. Pishchulin, J. Eledath, and S. G. Narasimhan, "Tessetrack: End-to-end learnable multi-person articulated 3d pose tracking," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 15190–15200, 2021.

[234] L. Kumarapu and P. Mukherjee, "Animepose: Multi-person 3d pose estimation and animation," *Pattern Recognition Letters*, vol. 147, pp. 16–24, 2021.

[235] Z. Zhou, Q. Shuai, Y. Wang, Q. Fang, X. Ji, F. Li, H. Bao, and X. Zhou, "Quickpose: Real-time multi-view multi-person pose estimation in crowded scenes," in *ACM SIGGRAPH 2022 Conference Proceedings*, pp. 1–9, 2022.

[236] F. Yang, S. Odashima, S. Yamao, H. Fujimoto, S. Masui, and S. Jiang, "A unified multi-view multi-person tracking framework," *arXiv preprint arXiv:2302.03820*, 2023.

[237] I. Oikonomidis, N. Kyriazis, and A. A. Argyros, "Efficient model-based 3d tracking of hand articulations using kinect.," in *BmVC*, vol. 1, p. 3, 2011.

[238] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of ICNN'95-international conference on neural networks*, vol. 4, pp. 1942–1948, IEEE, 1995.

[239] K. Levenberg, "A method for the solution of certain non-linear problems in least squares," *Quarterly of applied mathematics*, vol. 2, no. 2, pp. 164–168, 1944.

[240] R. Fletcher, *Practical methods of optimization.* John Wiley & Sons, 2013.

[241] P. Suryanarayan, A. Subramanian, and D. Mandalapu, "Dynamic hand pose recognition using depth data," in *2010 20th International Conference on Pattern Recognition*, pp. 3105–3108, IEEE, 2010.

[242] X. Sun, Y. Wei, S. Liang, X. Tang, and J. Sun, "Cascaded hand pose regression," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 824–832, 2015.

[243] H. Liang, J. Yuan, D. Thalmann, and Z. Zhang, "Model-based hand pose estimation via spatial-temporal hand parsing and 3d fingertip localization," *The Visual Computer*, vol. 29, no. 6, pp. 837–848, 2013.

[244] S. Melax, L. Keselman, and S. Orsten, "Dynamics based 3d skeletal hand tracking," in *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, pp. 184–184, 2013.

[245] S. Fleishman, M. Kliger, A. Lerner, and G. Kutliroff, "Icpik: Inverse kinematics based articulated-icp," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 28–35, 2015.

[246] A. Makris, N. Kyriazis, and A. A. Argyros, "Hierarchical particle filtering for 3d hand tracking," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 8–17, 2015.

[247] M. Schröder, J. Maycock, H. Ritter, and M. Botsch, "Real-time hand tracking using synergistic inverse kinematics," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5447–5454, IEEE, 2014.

[248] K. Roditakis and A. A. Argyros, "Quantifying the effect of a colored glove in the 3d tracking of a human hand," in *ICVS*, (Copenhagen, Denmark), pp. 404–414, Springer, July 2015.

[249] J. Tompson, M. Stein, Y. Lecun, and K. Perlin, "Real-time continuous pose recovery of human hands using convolutional networks," *ACM Transactions on Graphics (ToG)*, vol. 33, no. 5, p. 169, 2014.

[250] G. Moon, S.-I. Yu, H. Wen, T. Shiratori, and K. M. Lee, "Interhand2.6m: A dataset and baseline for 3d interacting hand pose estimation from a single rgb image," *arXiv preprint arXiv:2008.09309*, 2020.

[251] J. Gu, Z. Wang, W. Ouyang, J. Li, L. Zhuo, *et al.*, "3d hand pose estimation with disentangled cross-modal latent space," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 391–400, 2020.

[252] T. Theodoridis, T. Chatzis, V. Solachidis, K. Dimitropoulos, and P. Daras, "Cross-modal variational alignment of latent spaces," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pp. 960–961, 2020.

[253] A. e. a. Stergioulas, "3d hand pose estimation via aligned latent space injection and kinematic losses," in *Proceedings of the IEEE/CVF Conference on CVPR*, pp. 1730–1739, 2021.

[254] P. Ren, Y. Chen, J. Hao, H. Sun, Q. Qi, J. Wang, and J. Liao, "Two heads are better than one: Image-point cloud network for depth-based 3d hand pose estimation," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, pp. 2163–2171, 2023.

[255] D. Xiang, W. Xu, Y. Zhang, B. Peng, G. Wang, and K. Li, "Mtmvc: Semi-supervised 3d hand pose estimation using multi-task and multi-view consistency," *Journal of Visual Communication and Image Representation*, p. 103902, 2023.

[256] Z. Liu, G. Lin, C. Wang, M. Zheng, and F. Zhu, "Handmim: Pose-aware self-supervised learning for 3d hand mesh estimation," *arXiv preprint arXiv:2307.16061*, 2023.

[257] Q. Fu, X. Liu, R. Xu, J. C. Niebles, and K. M. Kitani, "Deformer: Dynamic fusion transformer for robust hand pose estimation," *arXiv preprint arXiv:2303.04991*, 2023.

[258] A. F. Abate, C. Bisogni, A. Castiglione, and M. Nappi, "Head pose estimation: An extensive survey on recent techniques and applications," *Pattern Recognition*, vol. 127, p. 108591, 2022.

[259] P. Padeleris, X. Zabulis, and A. A. Argyros, "Head pose estimation on depth data based on particle swarm optimization," in *IEEE Computer Vision and Pattern Recognition Workshops (CVPRW 2012)*, (Providence, Rhode Island, USA), pp. 42–49, IEEE, June 2012.

[260] X. Liu, W. Liang, Y. Wang, S. Li, and M. Pei, "3d head pose estimation with convolutional neural network trained on synthetic images," in *2016 IEEE International Conference on Image Processing (ICIP)*, pp. 1289–1293, 2016.

[261] R. Valle, J. M. Buenaposada, and L. Baumela, "Multi-task head pose estimation in-the-wild," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, pp. 2874–2881, aug 2021.

[262] A. P. Fard, H. Abdollahi, and M. Mahoor, "Asmnet: A lightweight deep neural network for face alignment and pose estimation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1521–1530, 2021.

[263] Y. Cheng, H. Wang, Y. Bao, and F. Lu, "Appearance-based gaze estimation with deep learning: A review and benchmark," *arXiv preprint arXiv:2104.12668*, 2021.

[264] K. Krafka, A. Khosla, P. Kellnhofer, H. Kannan, S. Bhandarkar, W. Matusik, and A. Torralba, "Eye tracking for everyone," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2176–2184, 2016.

[265] X. Zhang, Y. Sugano, M. Fritz, and A. Bulling, "Mpiigaze: Real-world dataset and deep appearance-based gaze estimation," *IEEE transactions on pattern analysis and machine intelligence*, vol. 41, no. 1, pp. 162–175, 2017.

[266] Z. Wu, S. Rajendran, T. Van As, V. Badrinarayanan, and A. Rabinovich, "Eyenet: A multi-task deep network for off-axis eye gaze estimation," in *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*, pp. 3683–3687, IEEE, 2019.

[267] S. Nonaka, S. Nobuhara, and K. Nishino, "Dynamic 3d gaze from afar: Deep gaze estimation from temporal eye-head-body coordination," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2192–2201, 2022.

[268] T. Fischer, H. J. Chang, and Y. Demiris, "Rt-gene: Real-time eye gaze estimation in natural environments," in *Proceedings of the European conference on computer vision (ECCV)*, pp. 334–352, 2018.

[269] E. Wood, T. Baltrušaitis, L.-P. Morency, P. Robinson, and A. Bulling, "Gazedirector: Fully articulated eye gaze redirection in video," in *Computer Graphics Forum*, vol. 37, pp. 217–225, Wiley Online Library, 2018.

[270] M. R. Koujan, M. C. Doukas, A. Roussos, and S. Zafeiriou, "Head2head: Video-based neural head synthesis," in *2020 15th IEEE International Conference on Automatic Face and Gesture Recognition (FG 2020)*, pp. 16–23, IEEE, 2020.

[271] M. C. Doukas, M. R. Koujan, V. Sharmanska, A. Roussos, and S. Zafeiriou, "Head2head++: Deep facial attributes re-targeting," *IEEE Transactions on Biometrics, Behavior, and Identity Science*, vol. 3, no. 1, pp. 31–43, 2021.

[272] M. Zollhöfer, J. Thies, P. Garrido, D. Bradley, T. Beeler, P. Pérez, M. Stamminger, M. Nießner, and C. Theobalt, "State of the art on monocular 3d face reconstruction, tracking, and applications," in *Computer Graphics Forum*, vol. 37, pp. 523–550, Wiley Online Library, 2018.

[273] P.-L. Hsieh, C. Ma, J. Yu, and H. Li, "Unconstrained realtime facial performance capture," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1675–1683, 2015.

[274] M. C. Doukas, E. Ververas, V. Sharmanska, and S. Zafeiriou, "Free-headgan: Neural talking head synthesis with explicit gaze control," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023.

[275] M. Boháček and H. Farid, "A geometric and photometric exploration of gan and diffusion synthesized faces," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pp. 874–883, June 2023.

[276] F. P. Papantoniou, A. Lattas, S. Moschoglou, and S. Zafeiriou, "Relightify: Relightable 3d faces from a single image via diffusion models," 2023.

[277] R. Daněček, M. J. Black, and T. Bolkart, "Emoca: Emotion driven monocular face capture and animation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 20311–20322, 2022.

[278] P.-W. Grassal, M. Prinzler, T. Leistner, C. Rother, M. Nießner, and J. Thies, "Neural head avatars from monocular rgb videos," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 18653–18664, 2022.

[279] S. Athar, Z. Xu, K. Sunkavalli, E. Shechtman, and Z. Shu, "Rignerf: Fully controllable neural 3d portraits," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 20364–20373, June 2022.

[280] Y. Kartynnik, A. Ablavatski, I. Grishchenko, and M. Grundmann, "Real-time facial surface geometry from monocular video on mobile gpus," *arXiv preprint arXiv:1907.06724*, 2019.

[281] H. Wang, J. Liu, J. Tang, and G. Wu, "Lightweight super-resolution head for human pose estimation," *arXiv preprint arXiv:2307.16765*, 2023.

[282] S. Suzuki *et al.*, "Topological structural analysis of digitized binary images by border following," *Computer vision, graphics, and image processing*, vol. 30, no. 1, pp. 32–46, 1985.

[283] S. X. Ju, M. J. Black, and Y. Yacoob, "Cardboard people: A parameterized model of articulated image motion," in *Proceedings of the Second International Conference on Automatic Face and Gesture Recognition*, pp. 38–44, IEEE, 1996.

[284] T. F. Cootes, C. J. Taylor, D. H. Cooper, and J. Graham, "Active shape models-their training and application," *Computer vision and image understanding*, vol. 61, no. 1, pp. 38–59, 1995.

[285] H. Sidenbladh, F. De la Torre, and M. J. Black, "A framework for modeling the appearance of 3d articulated figures," in *Proceedings Fourth IEEE International Conference on Automatic Face and Gesture Recognition (Cat. No. PR00580)*, pp. 368–375, IEEE, 2000.

[286] D. Anguelov, P. Srinivasan, D. Koller, S. Thrun, J. Rodgers, and J. Davis, "Scape: shape completion and animation of people," in *ACM transactions on graphics (TOG)*, vol. 24, pp. 408–416, ACM, 2005.

[287] A. Shapiro, A. Feng, R. Wang, H. Li, M. Bolas, G. Medioni, and E. Suma, "Rapid avatar capture and simulation using commodity depth sensors," *Computer Animation and Virtual Worlds*, vol. 25, no. 3-4, pp. 201–211, 2014.

[288] A. Feng, D. Casas, and A. Shapiro, "Avatar reshaping and automatic rigging using a deformable model," in *Proceedings of the 8th ACM SIGGRAPH Conference on Motion in Games*, pp. 57–64, ACM, 2015.

[289] H. Joo, T. Simon, and Y. Sheikh, "Total capture: A 3d deformation model for tracking faces, hands, and bodies," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 8320–8329, 2018.

[290] H. Xu, E. G. Bazavan, A. Zanfir, W. T. Freeman, R. Sukthankar, and C. Sminchisescu, "Ghum & ghuml: Generative 3d human shape and articulated pose models," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6184–6193, 2020.

[291] J. Romero, D. Tzionas, and M. J. Black, "Embodied hands: Modeling and capturing hands and bodies together," *ACM Transactions on Graphics (TOG)*, vol. 36, no. 6, p. 245, 2017.

[292] A. inc., "Autodesk motionbuilder: Bring characters and creatures to life." `https://www.autodesk.com/products/motionbuilder/`, 2019. Accessed: 2019-10-05.

[293] B. Egger, W. A. Smith, A. Tewari, S. Wuhrer, M. Zollhoefer, T. Beeler, F. Bernard, T. Bolkart, A. Kortylewski, S. Romdhani, *et al.*, "3d morphable face models—past, present, and future," *ACM Transactions on Graphics (TOG)*, vol. 39, no. 5, pp. 1–38, 2020.

[294] A. Tewari, M. Zollhofer, H. Kim, P. Garrido, F. Bernard, P. Perez, and C. Theobalt, "Mofa: Model-based deep convolutional face autoencoder for unsupervised monocular reconstruction," in *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pp. 1274–1283, 2017.

[295] K. Messer, J. Matas, J. Kittler, J. Luettin, G. Maitre, *et al.*, "Xm2vtsdb: The extended m2vts database," in *Second international conference on audio and video-based biometric person authentication*, vol. 964, pp. 965–966, Citeseer, 1999.

[296] Z. e. a. Cao, "Realtime multi-person 2d pose estimation using part affinity fields," in *CVPR*, 2017.

[297] T. von Marcard, R. Henschel, M. J. Black, B. Rosenhahn, and G. Pons-Moll, "Recovering accurate 3d human pose in the wild using imus and a moving camera," in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 601–617, 2018.

[298] R. A. e. a. Güler, "Densepose: Dense human pose estimation in the wild," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7297–7306, 2018.

[299] N. Mahmood, N. Ghorbani, N. F. Troje, G. Pons-Moll, and M. J. Black, "Amass: Archive of motion capture as surface shapes," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 5442–5451, 2019.

[300] H. Joo, H. Liu, L. Tan, L. Gui, B. Nabbe, I. Matthews, T. Kanade, S. Nobuhara, and Y. Sheikh, "Panoptic studio: A massively multiview system for social motion capture," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 3334–3342, 2015.

[301] F. G. Harvey, M. Yurick, D. Nowrouzezahrai, and C. Pal, "Robust motion in-betweening," *ACM Transactions on Graphics (TOG)*, vol. 39, no. 4, pp. 60–1, 2020.

[302] I. Akhter and M. J. Black, "Pose-conditioned joint angle limits for 3d human pose reconstruction," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1446–1455, 2015.

[303] R. Li, S. Yang, D. A. Ross, and A. Kanazawa, "Ai choreographer: Music conditioned 3d dance generation with aist++," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 13401–13412, 2021.

[304] H. Rhodin, C. Richardt, D. Casas, E. Insafutdinov, M. Shafiei, H.-P. Seidel, B. Schiele, and C. Theobalt, "Egocap: egocentric marker-less motion capture with two fisheye cameras," *ACM Transactions on Graphics (TOG)*, vol. 35, no. 6, pp. 1–11, 2016.

[305] A. Weitz, L. Colucci, S. Primas, and B. Bent, "Infiniteform: A synthetic, minimal bias dataset for fitness applications," *arXiv preprint arXiv:2110.01330*, 2021.

[306] G. Maldonado, F. Bailly, P. Souères, and B. Watier, "Angular momentum regulation strategies for highly dynamic landing in parkour," *Computer Methods in Biomechanics and Biomedical Engineering*, vol. 20, no. S1, pp. 123–124, 2017.

[307] H. Zhu, Y. Cao, H. Jin, W. Chen, D. Du, Z. Wang, S. Cui, and X. Han, "Deep fashion3d: A dataset and benchmark for 3d garment reconstruction from single images," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part I 16*, pp. 512–530, Springer, 2020.

[308] Z. Wang, L. Chen, S. Rathore, D. Shin, and C. Fowlkes, "Geometric pose affordance: 3d human pose with scene constraints," *arXiv preprint arXiv:1905.07718*, 2019.

[309] J. Li, R. Villegas, D. Ceylan, J. Yang, Z. Kuang, H. Li, and Y. Zhao, "Task-generic hierarchical human motion prior using vaes," in *2021 International Conference on 3D Vision (3DV)*, pp. 771–781, IEEE, 2021.

[310] Y. Duan, T. Shi, Z. Zou, Y. Lin, Z. Qian, B. Zhang, and Y. Yuan, "Single-shot motion completion with transformer," *arXiv preprint arXiv:2103.00776*, 2021.

[311] B. N. Oreshkin, A. Valkanas, F. G. Harvey, L.-S. Ménard, F. Bocquelet, and M. J. Coates, "Motion inbetweening via deep δ-interpolator," *arXiv preprint arXiv:2201.06701*, 2022.

[312] J. Zhang, J. Jiao, M. Chen, L. Qu, X. Xu, and Q. Yang, "3d hand pose tracking and estimation using stereo matching," *arXiv preprint arXiv:1610.07214*, 2016.

[313] S. Yuan, Q. Ye, B. Stenger, S. Jain, and T.-K. Kim, "Bighand2. 2m benchmark: Hand pose dataset and state of the art analysis," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4866–4874, 2017.

[314] F. Mueller, D. Mehta, O. Sotnychenko, S. Sridhar, D. Casas, and C. Theobalt, "Real-time hand tracking under occlusion from an egocentric rgb-d sensor," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1154–1163, 2017.

[315] C. Zimmermann and T. Brox, "Learning to estimate 3d hand pose from single rgb images," in *Proceedings of the IEEE international conference on computer vision*, pp. 4903–4911, 2017.

[316] T. Simon, H. Joo, I. Matthews, and Y. Sheikh, "Hand keypoint detection in single images using multiview bootstrapping," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pp. 1145–1153, 2017.

[317] Y. Wang, C. Peng, and Y. Liu, "Mask-pose cascaded cnn for 2d hand pose estimation from single color image," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 29, no. 11, pp. 3258–3268, 2018.

[318] F. Mueller, F. Bernard, O. Sotnychenko, D. Mehta, S. Sridhar, D. Casas, and C. Theobalt, "Ganerated hands for real-time 3d hand tracking from monocular rgb," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 49–59, 2018.

[319] C. Zimmermann, D. Ceylan, J. Yang, B. Russell, M. Argus, and T. Brox, "Freihand: A dataset for markerless capture of hand pose and shape from single rgb images," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 813–822, 2019.

[320] F. Gomez-Donoso, S. Orts-Escolano, and M. Cazorla, "Large-scale multiview 3d hand pose dataset," *Image and Vision Computing*, vol. 81, pp. 25–33, 2019.

[321] M. Koestinger, P. Wohlhart, P. M. Roth, and H. Bischof, "Annotated facial landmarks in the wild: A large-scale, real-world database for facial landmark localization," in *2011 IEEE international conference on computer vision workshops (ICCV workshops)*, pp. 2144–2151, IEEE, 2011.

[322] X. P. Burgos-Artizzu, P. Perona, and P. Dollár, "Robust face landmark estimation under occlusion," in *Proceedings of the IEEE international conference on computer vision*, pp. 1513–1520, 2013.

[323] B. A. Smith, Q. Yin, S. K. Feiner, and S. K. Nayar, "Gaze locking: passive eye contact detection for human-object interaction," in *Proceedings of the 26th annual ACM symposium on User interface software and technology*, pp. 271–280, 2013.

[324] W. Wu, C. Qian, S. Yang, Q. Wang, Y. Cai, and Q. Zhou, "Look at boundary: A boundary-aware face alignment algorithm," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2129–2138, 2018.

[325] X. Zhu, Z. Lei, X. Liu, H. Shi, and S. Z. Li, "Face alignment across large poses: A 3d solution," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 146–155, 2016.

[326] G. Fanelli, T. Weise, J. Gall, and L. Van Gool, "Real time head pose estimation from consumer depth cameras," in *Joint pattern recognition symposium*, pp. 101–110, Springer, 2011.

[327] H. Fahmy, F. Pastore, M. Bagherzadeh, and L. Briand, "Supporting deep neural network safety analysis and retraining through heatmap-based unsupervised learning," *IEEE Transactions on Reliability*, vol. 70, no. 4, pp. 1641–1657, 2021.

[328] G. Borghi, M. Venturelli, R. Vezzani, and R. Cucchiara, "Poseidon: Face-from-depth for driver pose estimation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4661–4670, 2017.

[329] K. Messer, J. Kittler, M. Sadeghi, S. Marcel, C. Marcel, S. Bengio, F. Cardinaux, C. Sanderson, J. Czyz, L. Vandendorpe, *et al.*, "Face verification competition on the xm2vts database," in *Audio-and Video-Based Biometric Person Authentication: 4th International Conference, AVBPA 2003 Guildford, UK, June 9–11, 2003 Proceedings 4*, pp. 964–974, Springer, 2003.

[330] National Institute of Standards and Technology, "Face recognition grand challenge (frgc)." `https://www.nist.gov/programs-projects/face-recognition-grand-challenge-frgc`, 2010. [Online; accessed 14-June-2023].

[331] X. Zhu and D. Ramanan, "Face detection, pose estimation, and landmark localization in the wild," in *2012 IEEE conference on computer vision and pattern recognition*, pp. 2879–2886, IEEE, 2012.

[332] P. N. Belhumeur, D. W. Jacobs, D. J. Kriegman, and N. Kumar, "Localizing parts of faces using a consensus of exemplars," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 12, pp. 2930–2940, 2013.

[333] C. Sagonas, G. Tzimiropoulos, S. Zafeiriou, and M. Pantic, "300 faces in-the-wild challenge: The first facial landmark localization challenge," in *Proceedings of the IEEE international conference on computer vision workshops*, pp. 397–403, 2013.

[334] E. Zhou, H. Fan, Z. Cao, Y. Jiang, and Q. Yin, "Extensive facial landmark localization with coarse-to-fine convolutional network cascade," in *Proceedings of the IEEE international conference on computer vision workshops*, pp. 386–391, 2013.

[335] L. A. Jeni, S. Tulyakov, L. Yin, N. Sebe, and J. F. Cohn, "The first 3d face alignment in the wild (3dfaw) challenge," in *Computer Vision–ECCV 2016 Workshops: Amsterdam, The Netherlands, October 8-10 and 15-16, 2016, Proceedings, Part II 14*, pp. 511–520, Springer, 2016.

[336] C. Sagonas, E. Antonakos, G, Tzimiropoulos, S. Zafeiriou, M. Pantic., "I-bug facial annotations for xm2vts, frgc ver.2, lfpw, helen, afw and 300w." `https://ibug.doc.ic.ac.uk/resources/facial-point-annotations/`, 2023. [Online; accessed 13-June-2023].

[337] E. Ilkou and M. Koutraki, "Symbolic vs sub-symbolic ai methods: Friends or enemies?," in *CIKM (Workshops)*, 2020.

[338] D. Metaxas and S. Zhang, "A review of motion analysis methods for human nonverbal communication computing," *Image and Vision Computing*, vol. 31, no. 6-7, pp. 421–433, 2013.

[339] F. Rosenblatt, "The perceptron: a probabilistic model for information storage and organization in the brain.," *Psychological review*, vol. 65, no. 6, p. 386, 1958.

[340] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural networks*, vol. 2, no. 5, pp. 359–366, 1989.

[341] Y. LeCun, D. Touresky, G. Hinton, and T. Sejnowski, "A theoretical framework for back-propagation," in *Proceedings of the 1988 connectionist models summer school*, vol. 1, pp. 21–28, San Mateo, CA, USA, 1988.

[342] J. J. DiCarlo, D. Zoccolan, and N. C. Rust, "How does the brain solve visual object recognition?," *Neuron*, vol. 73, no. 3, pp. 415–434, 2012.

[343] V. Ayzenberg and S. F. Lourenco, "Skeletal descriptions of shape provide unique perceptual information for object recognition," *Scientific reports*, vol. 9, no. 1, p. 9359, 2019.

[344] X. Yang, J. Yan, W. Wang, S. Li, B. Hu, and J. Lin, "Brain-inspired models for visual object recognition: an overview," *Artificial Intelligence Review*, vol. 55, no. 7, pp. 5263–5311, 2022.

[345] N. K. Logothetis, T. Vetter, A. Hurlbert, and T. Poggio, "View-based models of 3d object recognition and class-specific invariances," *AI Memos (1959 - 2004)*, 1994.

[346] M. Tancik, P. Srinivasan, B. Mildenhall, S. Fridovich-Keil, N. Raghavan, U. Singhal, R. Ramamoorthi, J. Barron, and R. Ng, "Fourier features let networks learn high frequency functions in low dimensional domains," *Advances in Neural Information Processing Systems*, vol. 33, pp. 7537–7547, 2020.

[347] G. L. Marchetti, C. Hillar, D. Kragic, and S. Sanborn, "Harmonics of learning: Universal fourier features emerge in invariant networks," 2023.

[348] R. Shwartz-Ziv and Y. LeCun, "To compress or not to compress–self-supervised learning and information theory: A review," *arXiv preprint arXiv:2304.09355*, 2023.

[349] J.-P. Eckmann, S. O. Kamphorst, D. Ruelle, *et al.*, "Recurrence plots of dynamical systems," *World Scientific Series on Nonlinear Science Series A*, vol. 16, pp. 441–446, 1995.

[350] N. Marwan, M. C. Romano, M. Thiel, and J. Kurths, "Recurrence plots for the analysis of complex systems," *Physics reports*, vol. 438, no. 5-6, pp. 237–329, 2007.

[351] Z. Wang, T. Oates, *et al.*, "Encoding time series as images for visual inspection and classification using tiled convolutional neural networks," in *Workshops at the twenty-ninth AAAI conference on artificial intelligence*, vol. 1, AAAI Menlo Park, CA, USA, 2015.

[352] G. Zhang, Y. Si, D. Wang, W. Yang, and Y. Sun, "Automated detection of myocardial infarction using a gramian angular field and principal component analysis network," *IEEE Access*, vol. 7, pp. 171570–171583, 2019.

[353] M. M. Islam and M. M. H. Shuvo, "Densenet based speech imagery eeg signal classification using gramian angular field," in *2019 5th International Conference on Advances in Electrical Engineering (ICAEE)*, pp. 149–154, IEEE, 2019.

[354] S. Hong, J. Yoon, Y. Ham, B. Lee, and H. Kim, "Monitoring safety behaviors of scaffolding workers using gramian angular field convolution neural network based on imu sensing data," *Automation in Construction*, vol. 148, p. 104748, 2023.

[355] H. Bay, T. Tuytelaars, and L. Van Gool, "Surf: Speeded up robust features," in *Computer Vision–ECCV 2006: 9th European Conference on Computer Vision, Graz, Austria, May 7-13, 2006. Proceedings, Part I 9*, pp. 404–417, Springer, 2006.

[356] L. Zheng, Y. Yang, and Q. Tian, "Sift meets cnn: A decade survey of instance retrieval," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 5, pp. 1224–1244, 2017.

[357] A. Qammaz, "Rgbdacquisition, bvh motion capture loader." `https://github.com/AmmarkoV/RGBDAcquisition/tree/master/opengl_acquisition_shared_library/opengl_depth_and_color_renderer/src/Library/MotionCaptureLoader/`, 2019.

[358] Z. Zhang, "Flexible camera calibration by viewing a plane from unknown orientations," in *Proceedings of the seventh ieee international conference on computer vision*, vol. 1, pp. 666–673, Ieee, 1999.

[359] Coursera, Andrew Ng, "Mini-batch gradient descent," 2020. [Online; accessed 8-April-2020].

[360] N. Neverova, C. Wolf, F. Nebout, and G. W. Taylor, "Hand pose estimation through semi-supervised and weakly-supervised learning," *Computer Vision and Image Understanding*, vol. 164, pp. 56–67, 2017.

[361] A. Spurr, U. Iqbal, P. Molchanov, O. Hilliges, and J. Kautz, "Weakly supervised 3d hand pose estimation via biomechanical constraints," *arXiv preprint arXiv:2003.09282*, vol. 8, 2020.

[362] S. Cobos, M. Ferre, M. Sanchez Uran, J. Ortego, and C. Pena, "Efficient human hand kinematics for manipulation tasks," in *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2246–2251, 2008.

[363] M. Soubeyrand, B. Assabah, M. Bégin, E. Laemmel, A. Dos Santos, and M. Crézé, "Pronation and supination of the hand: anatomy and biomechanics," *Hand Surgery and Rehabilitation*, vol. 36, no. 1, pp. 2–11, 2017.

[364] MakeHuman Community, "Makehuman for blender 2 (mpfb2)." `https://github.com/makehumancommunity/mpfb2`, 2023. [Online; accessed 13-June-2023].

[365] V. Bazarevsky, Y. Kartynnik, A. Vakunov, K. Raveendran, and M. Grundmann, "Blazeface: Sub-millisecond neural face detection on mobile gpus," *arXiv preprint arXiv:1907.05047*, 2019.

[366] H.-J. Lee and Z. Chen, "Determination of 3d human body postures from a single view," *Computer Vision, Graphics, and Image Processing*, vol. 30, no. 2, pp. 148–168, 1985.

[367] H. Jiang, "3d human pose reconstruction using millions of exemplars," in *2010 20th International Conference on Pattern Recognition*, pp. 1674–1677, IEEE, 2010.

[368] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Communications of the ACM*, vol. 18, no. 9, pp. 509–517, 1975.

[369] A. Agarwal and B. Triggs, "Recovering 3d human pose from monocular images," *IEEE transactions on pattern analysis and machine intelligence*, vol. 28, no. 1, pp. 44–58, 2005.

[370] A. S. Micilotta, E.-J. Ong, and R. Bowden, "Real-time upper body detection and 3d pose estimation in monoscopic images," in *Computer Vision–ECCV 2006: 9th European Conference on Computer Vision, Graz, Austria, May 7-13, 2006, Proceedings, Part III 9*, pp. 139–150, Springer, 2006.

[371] A. Gupta, J. Martinez, J. J. Little, and R. J. Woodham, "3d pose from motion for cross-view action recognition via non-linear circulant temporal encoding," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2601–2608, 2014.

[372] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Mathematics of control, signals and systems*, vol. 2, no. 4, pp. 303–314, 1989.

[373] K. Hornik, "Approximation capabilities of multilayer feedforward networks," *Neural networks*, vol. 4, no. 2, pp. 251–257, 1991.

[374] M. Kubat, "Neural networks: a comprehensive foundation by simon haykin, macmillan, 1994, isbn 0-02-352781-7.," *The Knowledge Engineering Review*, vol. 13, no. 4, pp. 409–412, 1999.

[375] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[376] L. Bottou, "Stochastic gradient descent tricks," in *Neural Networks: Tricks of the Trade: Second Edition*, pp. 421–436, Springer, 2012.

[377] N. Tishby and N. Zaslavsky, "Deep learning and the information bottleneck principle," in *2015 ieee information theory workshop (itw)*, pp. 1–5, IEEE, 2015.

[378] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems." `https://www.tensorflow.org/`, 2015. Software available from tensorflow.org.

[379] E. Yazan and M. F. Talu, "Comparison of the stochastic gradient descent based optimization techniques," in *2017 International Artificial Intelligence and Data Processing Symposium (IDAP)*, pp. 1–5, IEEE, 2017.

[380] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics* (G. Gordon, D. Dunson, and M. Dudík, eds.), vol. 15 of *Proceedings of Machine Learning Research*, (Fort Lauderdale, FL, USA), pp. 315–323, PMLR, 11–13 Apr 2011.

[381] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International conference on machine learning*, pp. 448–456, pmlr, 2015.

[382] W. Hao, W. Yizhou, L. Yaqin, and S. Zhili, "The role of activation function in cnn," in *2020 2nd International Conference on Information Technology and Computer Application (ITCA)*, pp. 429–432, IEEE, 2020.

[383] T. Cohen and M. Welling, "Group equivariant convolutional networks," in *Proceedings of The 33rd International Conference on Machine Learning* (M. F. Balcan and K. Q. Weinberger, eds.), vol. 48 of *Proceedings of Machine Learning Research*, (New York, New York, USA), pp. 2990–2999, PMLR, 20–22 Jun 2016.

[384] L. E. MacDonald, S. Ramasinghe, and S. Lucey, "Enabling equivariance for arbitrary lie groups," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8183–8192, 2022.

[385] C. Esteves, C. Allen-Blanchette, X. Zhou, and K. Daniilidis, "Polar transformer networks," *arXiv preprint arXiv:1709.01889*, 2017.

[386] M. Jaderberg, K. Simonyan, A. Zisserman, *et al.*, "Spatial transformer networks," *Advances in neural information processing systems*, vol. 28, 2015.

[387] K. S. Tai, P. Bailis, and G. Valiant, "Equivariant transformer networks," in *International Conference on Machine Learning*, pp. 6086–6095, PMLR, 2019.

[388] A. e. a. Vaswani, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[389] S. Chhabra, P. B. Dutta, H. Venkateswara, and B. Li, "Patchrot: A self-supervised technique for training vision transformers," *arXiv preprint arXiv:2210.15722*, 2022.

[390] S. Gidaris, P. Singh, and N. Komodakis, "Unsupervised representation learning by predicting image rotations," *arXiv preprint arXiv:1803.07728*, 2018.

[391] Z. Liu, H. Mao, C.-Y. Wu, C. Feichtenhofer, T. Darrell, and S. Xie, "A convnet for the 2020s," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 11976–11986, 2022.

[392] S. L. Smith, A. Brock, L. Berrada, and S. De, "Convnets match vision transformers at scale," 2023.

[393] Wikipedia contributors, "Central limit theorem — Wikipedia, the free encyclopedia." `https://en.wikipedia.org/w/index.php?title=Central_limit_theorem&oldid=1172251689`, 2023. [Online; accessed 4-September-2023].

[394] G. e. Huang, "Densely connected convolutional networks," in *Proceedings of the IEEE CVPR*, pp. 4700–4708, 2017.

[395] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[396] P. Baldi and P. J. Sadowski, "Understanding dropout," in *NIPS*, pp. 2814–2822, 2013.

[397] R. K. Srivastava, K. Greff, and J. Schmidhuber, "Highway networks," *arXiv preprint arXiv:1505.00387*, 2015.

[398] D. Molchanov, A. Ashukha, and D. Vetrov, "Variational dropout sparsifies deep neural networks," in *ICML*, pp. 2498–2507, JMLR. org, 2017.

[399] A. Krizhevsky, G. Hinton, *et al.*, "Learning multiple layers of features from tiny images," 2009.

[400] C. Schuhmann, R. Beaumont, R. Vencu, C. Gordon, R. Wightman, M. Cherti, T. Coombes, A. Katta, C. Mullis, M. Wortsman, *et al.*, "Laion-5b: An open large-scale dataset for training next generation image-text models," *Advances in Neural Information Processing Systems*, vol. 35, pp. 25278–25294, 2022.

[401] J. Frankle and M. Carbin, "The lottery ticket hypothesis: Finding sparse, trainable neural networks," *arXiv preprint arXiv:1803.03635*, 2018.

[402] F. Chollet *et al.*, "Keras documentation - model checkpoint." `https://keras.io/callbacks/#modelcheckpoint`, 2015. [Online; accessed 8-April-2019].

[403] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," 2019.

[404] W. X. Zhao, K. Zhou, J. Li, T. Tang, X. Wang, Y. Hou, Y. Min, B. Zhang, J. Zhang, Z. Dong, Y. Du, C. Yang, Y. Chen, Z. Chen, J. Jiang, R. Ren, Y. Li, X. Tang, Z. Liu, P. Liu, J.-Y. Nie, and J.-R. Wen, "A survey of large language models," 2023.

[405] A. Qammaz, "Mocapnet 4 total capture, google collab notebook demo." `https://colab.research.google.com/github/FORTH-ModelBasedTracker/MocapNET/blob/mnet4/mocapnet4.ipynb`, 2023. [Online; accessed 14-September-2023].

[406] S. Shankar and A. Reuther, "Trends in energy estimates for computing in ai/machine learning accelerators, supercomputers, and compute-intensive applications," in *2022 IEEE High Performance Extreme Computing Conference (HPEC)*, pp. 1–8, IEEE, 2022.

[407] E. Scenarios, "Ipcc special report," *Cambridge Univ, Cambridge*, 2000.

[408] Z. Jiang, M. Yang, M. Tsirlin, R. Tang, Y. Dai, and J. Lin, ""low-resource" text classification: A parameter-free classification method with compressors," in *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 6810–6828, 2023.

[409] I. O. Tolstikhin, N. Houlsby, A. Kolesnikov, L. Beyer, X. Zhai, T. Unterthiner, J. Yung, A. Steiner, D. Keysers, J. Uszkoreit, *et al.*, "Mlp-mixer: An all-mlp architecture for vision," *Advances in Neural Information Processing Systems*, vol. 34, pp. 24261–24272, 2021.

[410] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, "Language models are few-shot learners," 2020.

[411] Google, *Tensorflow Weight Clustering comprehensive guide*, Oct. 2022. `https://www.tensorflow.org/model_optimization/guide/clustering/clustering_comprehensive_guide`.

[412] Google, *Tensorflow Quantization aware training comprehensive guide*, Oct. 2022. `https://www.tensorflow.org/model_optimization/guide/quantization/training_comprehensive_guide`.

[413] Google, *Tensorflow Model Pruning comprehensive guide*, Oct. 2022. `https://www.tensorflow.org/model_optimization/guide/pruning/comprehensive_guide`.

[414] Z. Liu, M. Sun, T. Zhou, G. Huang, and T. Darrell, "Rethinking the value of network pruning," *arXiv preprint arXiv:1810.05270*, 2018.

[415] J. e. a. Frankle, "Pruning neural networks at initialization: Why are we missing the mark?," *arXiv preprint arXiv:2009.08576*, 2020.

[416] J. Frankle, D. J. Schwab, and A. S. Morcos, "The early phase of neural network training," *arXiv preprint arXiv:2002.10365*, 2020.

[417] A. Shahroudnejad, "A survey on understanding, visualizations, and explanation of deep neural networks," *arXiv preprint arXiv:2102.01792*, 2021.

[418] Foundation for Research and Technology – Hellas, Institute of Computer Science, "Ambient intelligence programme." `https://ami.ics.forth.gr/en/home/`, 2023. [Online; accessed 14-September-2023].

[419] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[420] H. e. a. Abdi, "Principal component analysis," *Wiley interdisciplinary reviews: computational statistics*, vol. 2, no. 4, pp. 433–459, 2010.

[421] L. McInnes, J. Healy, and J. Melville, "Umap: Uniform manifold approximation and projection for dimension reduction," *arXiv preprint arXiv:1802.03426*, 2018.

[422] D. D. Lee and H. S. Seung, "Learning the parts of objects by non-negative matrix factorization," *Nature*, vol. 401, no. 6755, pp. 788–791, 1999.

[423] J. Mairal, F. Bach, J. Ponce, and G. Sapiro, "Online dictionary learning for sparse coding," in *Proceedings of the 26th annual international conference on machine learning*, pp. 689–696, 2009.

[424] I. T. Jolliffe, *Principal component analysis for special types of data*. Springer, 2002.

[425] N. Halko, P.-G. Martinsson, and J. A. Tropp, "Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions," *SIAM review*, vol. 53, no. 2, pp. 217–288, 2011.

[426] P.-G. Martinsson, V. Rokhlin, and M. Tygert, "A randomized algorithm for the decomposition of matrices," *Applied and Computational Harmonic Analysis*, vol. 30, no. 1, pp. 47–68, 2011.

[427] D. Barber, *Bayesian reasoning and machine learning. Algorithm 21.1*. Cambridge University Press, 2012.

[428] A. Hyvarinen, "Fast and robust fixed-point algorithms for independent component analysis," *IEEE transactions on Neural Networks*, vol. 10, no. 3, pp. 626–634, 1999.

[429] J. Su, D. V. Vargas, and K. Sakurai, "One pixel attack for fooling deep neural networks," *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 5, pp. 828–841, 2019.

[430] D. G. Lowe, "Robust model-based motion tracking through the integration of search and estimation," *International Journal of Computer Vision*, vol. 8, no. 2, pp. 113–122, 1992.

[431] S. Agarwal, K. Mierle, and Others, "Ceres solver." `http://ceres-solver.org`.

[432] J. J. Moré, "The levenberg-marquardt algorithm: implementation and theory," in *Numerical analysis*, pp. 105–116, Springer, 1978.

[433] M. I. Lourakis *et al.*, "A brief description of the levenberg-marquardt algorithm implemented by levmar," *Foundation of Research and Technology*, vol. 4, no. 1, pp. 1–6, 2005.

[434] Wikipedia contributors, "Newton's method — Wikipedia, the free encyclopedia," 2020. [Online; accessed 24-April-2020].

[435] A. Aristidou and J. Lasenby, "Fabrik: A fast, iterative solver for the inverse kinematics problem," *Graphical Models*, vol. 73, no. 5, pp. 243–260, 2011.

[436] J. Konečnỳ, Z. Qu, and P. Richtárik, "Semi-stochastic coordinate descent," *optimization Methods and Software*, vol. 32, no. 5, pp. 993–1005, 2017.

[437] Wikipedia contributors, "Butterworth filter — Wikipedia, the free encyclopedia," 2020. [Online; accessed 26-April-2020].

[438] I. W. Selesnick and C. S. Burrus, "Generalized digital butterworth filter design," *IEEE Transactions on signal processing*, vol. 46, no. 6, pp. 1688–1694, 1998.

[439] J. G. Wardrop, "Road paper. some theoretical aspects of road traffic research.," *Proceedings of the institution of civil engineers*, vol. 1, no. 3, pp. 325–362, 1952.

[440] J. Nash, "Non-cooperative games," *Annals of mathematics*, pp. 286–295, 1951.

[441] S. Dohare, J. F. Hernandez-Garcia, P. Rahman, R. S. Sutton, and A. R. Mahmood, "Maintaining plasticity in deep continual learning," *arXiv preprint arXiv:2306.13812*, 2023.

[442] R. S. Zimmermann, Y. Sharma, S. Schneider, M. Bethge, and W. Brendel, "Contrastive learning inverts the data generating process," in *Proceedings of the 38th International Conference on Machine Learning* (M. Meila and T. Zhang, eds.), vol. 139 of *Proceedings of Machine Learning Research*, pp. 12979–12990, PMLR, 18–24 Jul 2021.

[443] M. Kirby and L. Sirovich, "Application of the karhunen-loeve procedure for the characterization of human faces," *IEEE Transactions on Pattern analysis and Machine intelligence*, vol. 12, no. 1, pp. 103–108, 1990.

[444] Y. Song and S. Ermon, "Generative modeling by estimating gradients of the data distribution," in *Advances in Neural Information Processing Systems*, pp. 11895–11907, 2019.

[445] Y. Song and S. Ermon, "Improved techniques for training score-based generative models," in *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual* (H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, eds.), 2020.

[446] Yang Song, Ammar Qammaz, Kostis Tzevelekakis, "Pose diffusion based on improved techniques for training score-based generative models." `https://github.com/AmmarkoV/ncsnv2_archimedes`, 2022. [Online; accessed 9-September-2023].

[447] Z. Jiang, Z. Zhou, L. Li, W. Chai, C.-Y. Yang, and J.-N. Hwang, "Back to optimization: Diffusion-based zero-shot 3d human pose estimation," *arXiv preprint arXiv:2307.03833*, 2023.

[448] J. Wang, C. Rupprecht, and D. Novotny, "Posediffusion: Solving pose estimation via diffusion-aided bundle adjustment," *arXiv preprint arXiv:2306.15667*, 2023.

[449] J. Gong, L. G. Foo, Z. Fan, Q. Ke, H. Rahmani, and J. Liu, "Diffpose: Toward more reliable 3d pose estimation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2023.

[450] Qualcomm Inc., "World's first on device demonstration of stable diffusion on android." `https://www.qualcomm.com/news/onq/2023/02/worlds-first-on-device-demonstration-of-stable-diffusion-on-android`, 2023. [Online; accessed 9-October-2023].

[451] I. Kostrikov and J. Gall, "Depth sweep regression forests for estimating 3d human pose from images.," in *BMVC*, vol. 1, p. 5, 2014.

[452] L. Bo and C. Sminchisescu, "Twin gaussian processes for structured prediction," *International Journal of Computer Vision*, vol. 87, no. 1-2, p. 28, 2010.

[453] H. Yasin, U. Iqbal, B. Kruger, A. Weber, and J. Gall, "A dual-source approach for 3d pose estimation from a single image," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4948–4956, 2016.

[454] U. Von Agris and K.-F. Kraiss, "Towards a video corpus for signer-independent continuous sign language recognition," *Gesture in Human-Computer Interaction and Simulation, Lisbon, Portugal, May*, vol. 11, 2007.

[455] L. Yang and A. Yao, "Disentangling latent hands for image synthesis and pose estimation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9877–9886, 2019.

[456] A. Spurr, J. Song, S. Park, and O. Hilliges, "Cross-modal deep variational hand pose estimation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 89–98, 2018.

[457] U. Iqbal, P. Molchanov, T. B. J. Gall, and J. Kautz, "Hand pose estimation via latent 2.5 d heatmap regression," in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 118–134, 2018.

[458] P. Ramachandran, B. Zoph, and Q. V. Le, "Searching for activation functions," *arXiv preprint arXiv:1710.05941*, 2017.

[459] Y. Ganin, D. Kononenko, D. Sungatullina, and V. Lempitsky, "Deepwarp: Photorealistic image resynthesis for gaze manipulation," in *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part II 14*, pp. 311–326, Springer, 2016.

[460] Z. He, A. Spurr, X. Zhang, and O. Hilliges, "Photo-realistic monocular gaze redirection using generative adversarial networks," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 6932–6941, 2019.

[461] W. Xia, Y. Yang, J.-H. Xue, and W. Feng, "Controllable continuous gaze redirection," in *Proceedings of the 28th ACM International Conference on Multimedia*, pp. 1782–1790, 2020.

[462] B. Martinez and M. F. Valstar, "L2, 1-based regression and prediction accumulation across views for robust facial landmark detection," *Image and Vision Computing*, vol. 47, pp. 36–44, 2016.

[463] J. Čech, V. Franc, M. Uřičář, and J. Matas, "Multi-view facial landmark detection by using a 3d shape model," *Image and Vision Computing*, vol. 47, pp. 60–70, 2016.

[464] M. Uřičář, V. Franc, D. Thomas, A. Sugimoto, and V. Hlaváč, "Multi-view facial landmark detector learned by the structured output svm," *Image and Vision Computing*, vol. 47, pp. 45–59, 2016.

[465] J. Deng, Q. Liu, J. Yang, and D. Tao, "M3 csr: Multi-view, multi-scale and multi-component cascade shape regression," *Image and Vision Computing*, vol. 47, pp. 19–26, 2016.

[466] H. Fan and E. Zhou, "Approaching human level facial landmark localization by deep learning," *Image and Vision Computing*, vol. 47, pp. 27–35, 2016.

[467] L. Zhang, A. Rao, and M. Agrawala, "Adding conditional control to text-to-image diffusion models."

[468] Y. Du, W. Wang, and L. Wang, "Hierarchical recurrent neural network for skeleton based action recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1110–1118, 2015.

[469] S. Tripathi, L. Müller, C.-H. P. Huang, O. Taheri, M. J. Black, and D. Tzionas, "3d human pose estimation via intuitive physics," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4713–4725, 2023.

[470] S. Tripathi, A. Chatterjee, J.-C. Passy, H. Yi, D. Tzionas, and M. J. Black, "Deco: Dense estimation of 3d human-scene contact in the wild," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 8001–8013, 2023.

[471] S. Xu, Z. Li, Y.-X. Wang, and L.-Y. Gui, "Interdiff: Generating 3d human-object interactions with physics-informed diffusion," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 14928–14940, 2023.

[472] Y. Yuan, J. Song, U. Iqbal, A. Vahdat, and J. Kautz, "Physdiff: Physics-guided human motion diffusion model," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 16010–16021, 2023.

[473] H. Guo and Q. Ji, "Physics-augmented autoencoder for 3d skeleton-based gait recognition," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 19627–19638, 2023.

[474] V. Sklyarova, J. Chelishev, A. Dogaru, I. Medvedev, V. Lempitsky, and E. Zakharov, "Neural haircut: Prior-guided strand-based hair reconstruction," *arXiv preprint arXiv:2306.05872*, 2023.

[475] J. Karras, A. Holynski, T.-C. Wang, and I. Kemelmacher-Shlizerman, "Dreampose: Fashion video synthesis with stable diffusion," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 22680–22690, 2023.

[476] A. Baldrati, D. Morelli, G. Cartella, M. Cornia, M. Bertini, and R. Cucchiara, "Multimodal garment designer: Human-centric latent diffusion models for fashion image editing supplementary material,"

[477] L. Dai, L. Ma, S. Qian, H. Liu, Z. Liu, and H. Xiong, "Cloth2body: Generating 3d human body mesh from 2d clothing," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 15007–15017, 2023.

[478] X. Zhang, B. Yang, M. C. Kampffmeyer, W. Zhang, S. Zhang, G. Lu, L. Lin, H. Xu, and X. Liang, "Diffcloth: Diffusion based garment synthesis and manipulation via structural cross-modal semantic alignment," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 23154–23163, 2023.

[479] W. Xu, W. Du, H. Xue, Y. Li, R. Ye, Y.-F. Wang, and C. Lu, "Clothpose: A real-world benchmark for visual analysis of garment pose via an indirect recording solution," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 58–68, 2023.

[480] B. Zhou, H. Zhou, T. Liang, Q. Yu, S. Zhao, Y. Zeng, J. Lv, S. Luo, Q. Wang, X. Yu, *et al.*, "Clothesnet: An information-rich 3d garment model repository with simulated clothes environment," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 20428–20438, 2023.

[481] C.-S. Lin and C.-K. Li, "A sum-of-product neural network (sopnn)," *Neurocomputing*, vol. 30, no. 1-4, pp. 273–291, 2000.

[482] C. H. Papadimitriou, S. S. Vempala, D. Mitropolsky, M. Collins, and W. Maass, "Brain computation by assemblies of neurons," *Proceedings of the National Academy of Sciences*, vol. 117, no. 25, pp. 14464–14472, 2020.

[483] S. Ghosh-Dastidar and H. Adeli, "Spiking neural networks," *International journal of neural systems*, vol. 19, no. 04, pp. 295–308, 2009.

[484] A. Gasnikov and Y. Nesterov, "Universal fast gradient method for stochastic composit optimization problems," *arXiv preprint arXiv:1604.05275*, 2016.

[485] O. Borysenko and M. Byshkin, "Coolmomentum: A method for stochastic optimization by langevin dynamics with simulated annealing," *Scientific Reports*, vol. 11, no. 1, p. 10705, 2021.

[486] X. Chen, Z. Liu, S. Xie, and K. He, "Deconstructing denoising diffusion models for self-supervised learning," 2024.

[487] N. Hesse, S. Pujades, M. J. Black, M. Arens, U. G. Hofmann, and A. S. Schroeder, "Learning and tracking the 3d body shape of freely moving infants from rgb-d sequences," *IEEE transactions on pattern analysis and machine intelligence*, vol. 42, no. 10, pp. 2540–2551, 2019.

[488] A. Soualmi, C. Ducottet, H. Patural, A. Giraud, and O. Alata, "A 3d pose estimation framework for preterm infants hospitalized in the neonatal unit," *Multimedia Tools and Applications*, pp. 1–18, 2023.

[489] L. Ruddigkeit, R. Van Deursen, L. C. Blum, and J.-L. Reymond, "Enumeration of 166 billion organic small molecules in the chemical universe database gdb-17," *Journal of chemical information and modeling*, vol. 52, no. 11, pp. 2864–2875, 2012.

[490] D. Grammenos, X. Zabulis, A. A. Argyros, and C. Stephanidis, "Forth-ics internal rtd programme ambient intelligence and smart environments," in *3rd European Conference on Ambient Intelligence, Salzburg, Austria*, 2009.

[491] A. Qammaz, "Mocapnet 1 / october 2019 demo video." `https://www.youtube.com/watch?v=pKTA_odo9Xw`, 2020.

[492] B. Kerbl, G. Kopanas, T. Leimkühler, and G. Drettakis, "3d gaussian splatting for real-time radiance field rendering," *ACM Transactions on Graphics (ToG)*, vol. 42, no. 4, pp. 1–14, 2023.

[493] J. Kerr, C. M. Kim, K. Goldberg, A. Kanazawa, and M. Tancik, "Lerf: Language embedded radiance fields," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 19729–19739, 2023.

[494] A. Qammaz, "Mocapnet repository branch for its ros compatible wrapper." `https://github.com/FORTH-ModelBasedTracker/mocapnet_rosnode`, 2020. [Online; accessed 9-August-2023].

[495] S. Piperakis, A. Qammaz, "Action client wrapper for nao control using the mocapnet ros wrapper." `https://github.com/mrsp/action_client_test/blob/main/src/mocapnet_control.py`, 2021. [Online; accessed 9-August-2023].

[496] A. Qammaz, "Mocapnet / blender 3.4+ / makehuman mpfb2 virtual human avatar animation from rgb videos." `https://www.youtube.com/watch?v=ooLRUS5j4AI`, 2023. [Online; accessed 11-July-2023].

[497] P. Tassi, *Mark Zuckerbergs metaverse legs demo was staged with motion capture*. Forbes, Oct. 2022. `https://www.forbes.com/sites/paultassi/2022/10/14/mark-zuckerbergs-metaverse-legs-demo-was-staged-with-motion-capture/`.

[498] HTC, "Vrchat - full-body tracking guide," 2020. [Online; accessed 16-Feburary-2021].

[499] HTC, "Htc - vive tracker," 2020. [Online; accessed 16-Feburary-2021].

Appendix A
Publications, Systems and Models

Publications

The research activity related to this thesis has so far produced the following publications :

(1) A. Qammaz and A.A. Argyros, **"MocapNET: Ensemble of SNN Encoders for 3D Human Pose Estimation in RGB Images"** , In British Machine Vision Conference (BMVC 2019), BMVA, Cardiff, UK, September 2019.

(2) M. Bajones, D. Fischinger, A. Weiss, P.D.L. Puente, D. Wolf, M. Vincze, T. Körtner, M. Weninger, K. Papoutsakis, D. Michel, A. Qammaz, P. Panteleris, M. Foukarakis, I. Adami, D. Ioannidi, A. Leonidis, M. Antona, A. Argyros, P. Mayer, P. Panek, H. Eftring and S. Frennert, **"Results of Field Trials with a Mobile Service Robot for Older Adults in 16 Private Households"** , ACM Transactions on Human-Robot Interaction, ACM, vol. 9, no. 2, pp. 10:1-10:27, December 2019.

(3) A. Qammaz and A.A. Argyros, **"Occlusion-tolerant and personalized 3D human pose estimation in RGB images"** , In IEEE International Conference on Pattern Recognition (ICPR 2020), pp. 6904-6911, January 2021.

(4) A. Qammaz and A.A. Argyros, **"Towards Holistic Real-time Human 3D Pose Estimation using MocapNETs"** , In British Machine Vision Conference (BMVC 2021), BMVA, Virtual, UK, November 2021.

(5) H. Hauser, C. Beisswenger, N. Partarakis, X. Zabulis, I. Adami, E. Zidianakis, A. Patakos, N. Patsiouras, E. Karuzaki, M. Foukarakis, A. Tsoli, A. Qammaz, A. Argyros, N. Cadi, E. Baka, N.M. Thalmann, B. Olivias, D. Makrygiannis, A. Glushkova, S. Manitsaris, V. Nitti and L. Panesse, **"Multimodal Narratives for the Presentation of Silk Heritage in the Museum"** , MDPI, vol. 5, no. 1, pp. 461-487, 2022.

(6) A. Qammaz and A. Argyros, **"Compacting MocapNET-based 3D Human Pose Estimation via Dimensionality Reduction"** , In PETRA 2023, ACM, Corfu, Greece, July 2023.

(7) S. Panagou, M. Sileo, K. Papoutsakis, F. Fruggiero, A. Qammaz and A.A. Argyros, **"Complexity based investigation in collaborative assembly scenarios via non intrusive techniques"** , Procedia Computer Science, Special issue, 4th ISM 2022, Elsevier, vol. 217, pp. 478-485, 2023.

(8) A. Qammaz and A. Argyros, **"A Unified Approach for Occlusion Tolerant 3D Facial Pose Capture and Gaze Estimation using MocapNETs"** , In International Conference on Computer Vision Workshops (AMFG 2023 - ICCVW 2023), (to appear), IEEE, Paris, France, October 2023.

A.1 Posters

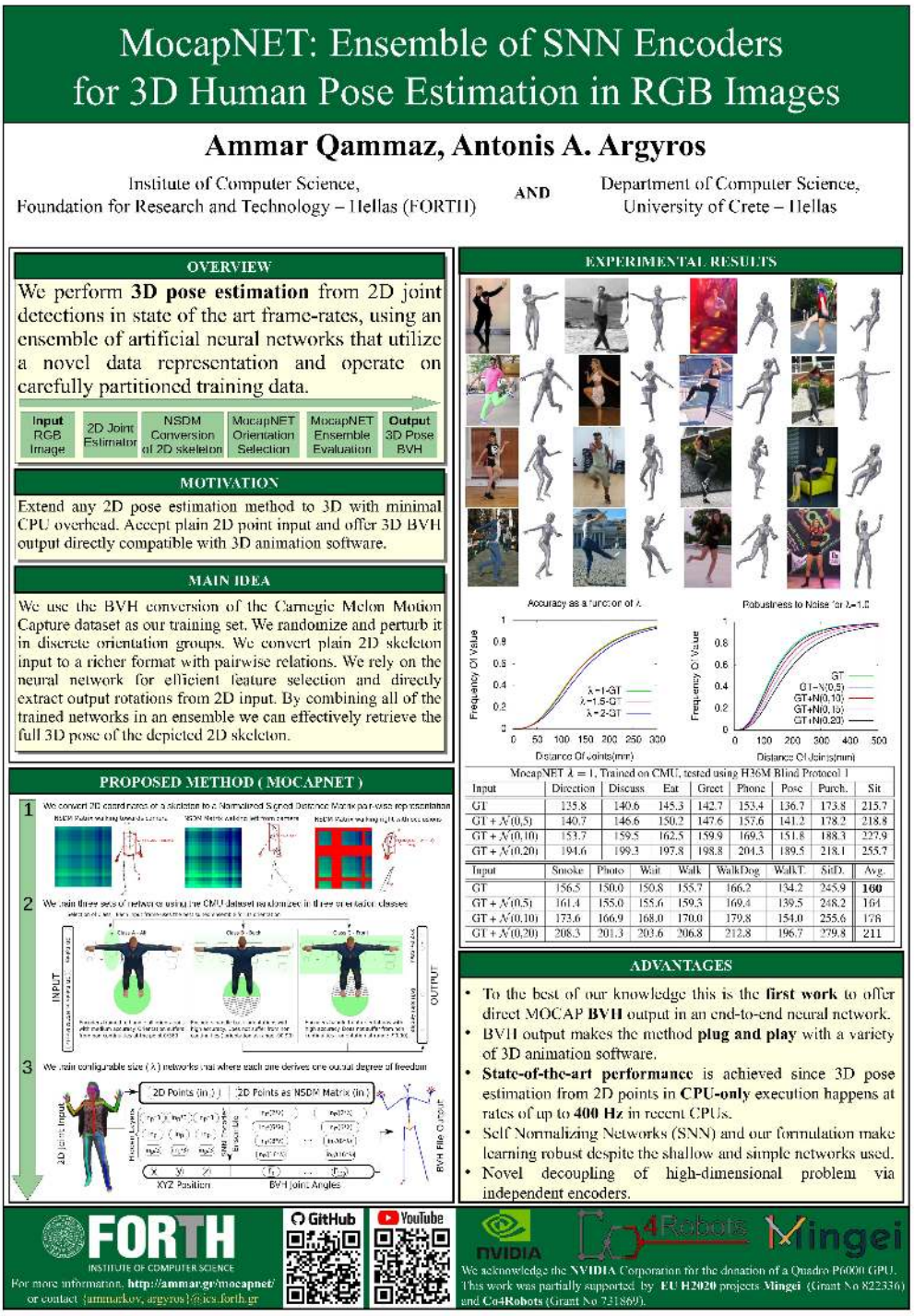


Figure A.1: BMVC 2019 Poster of MocapNET 1 [6]

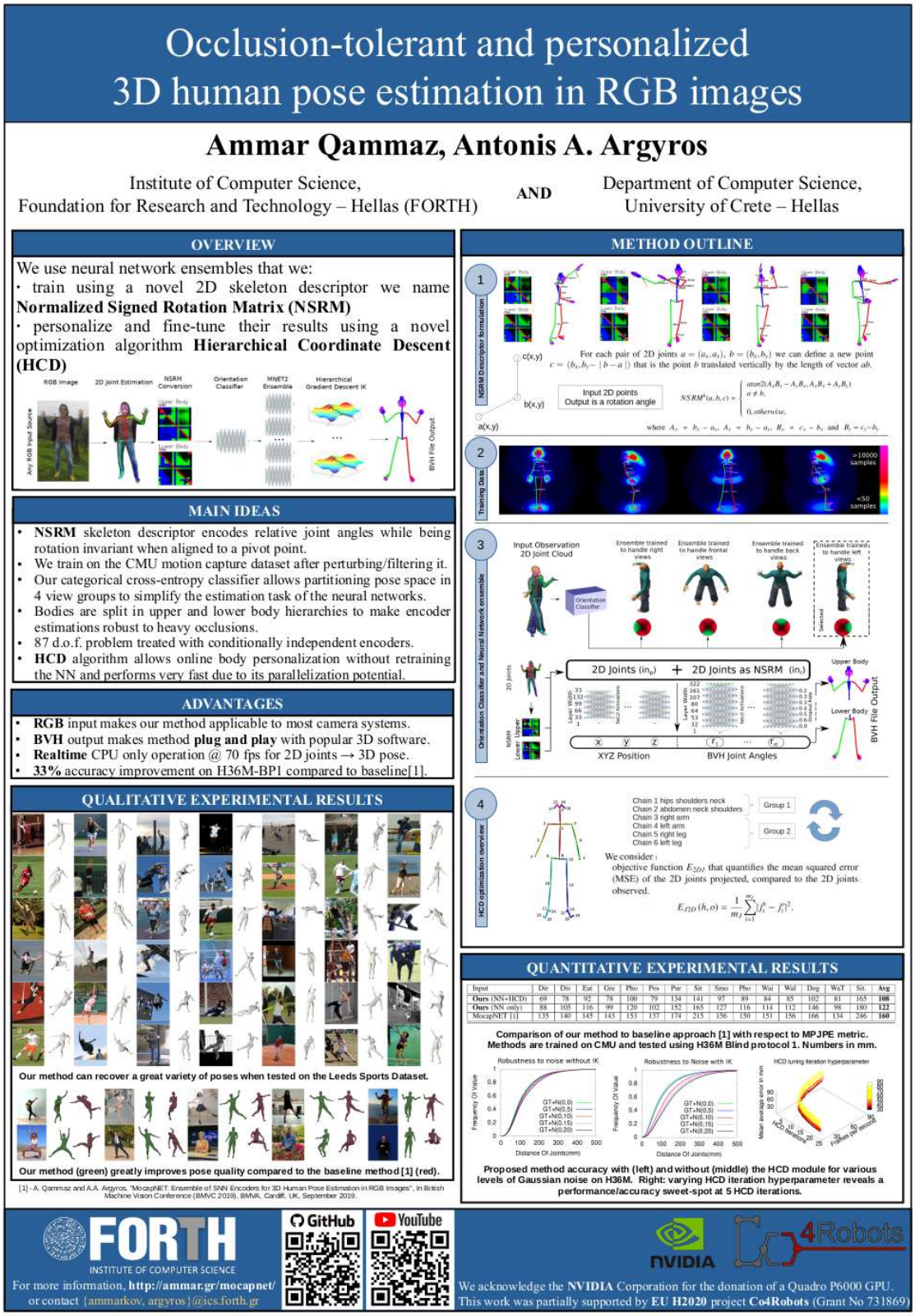


Figure A.2: ICPR 2020 Poster of MocapNET 2 [8]

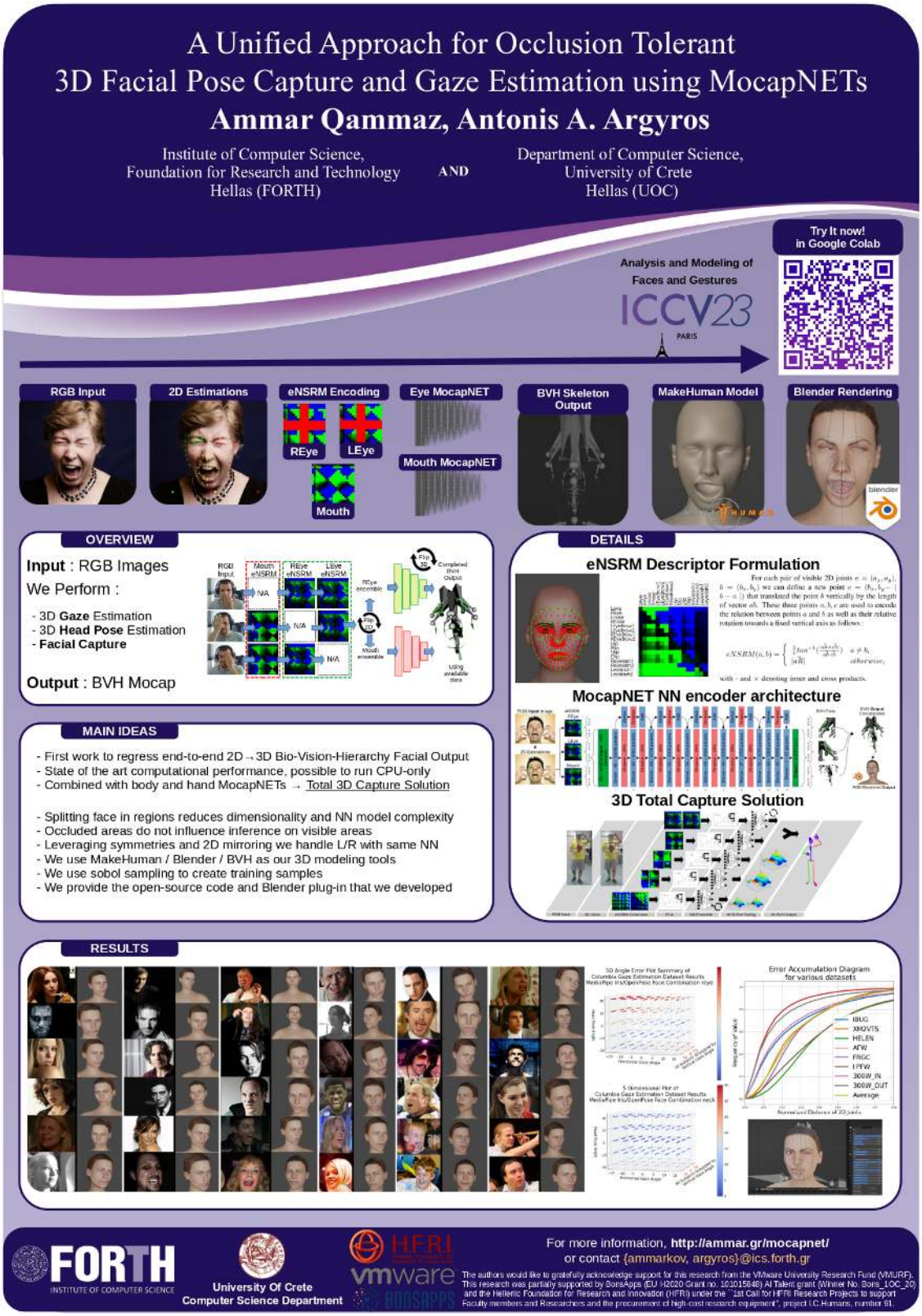


Figure A.3: ICCV 2023 Workshop Poster on MocapNET 4 [31]

A.2 Repositories

In the context of this thesis, the following discrete systems were developed :

- MocapNET development repository, hosted on the git repository of FORTH-ICS.
- MocapNET 1, published in [6] and hosted on the mnet1 branch of repository: github.com/FORTH-ModelBasedTracker/MocapNET
- MocapNET 2, published in [8] and hosted on the mnet2 branch of repository: github.com/FORTH-ModelBasedTracker/MocapNET
- MocapNET 3, published in [9] and hosted on the mnet3 branch of repository: github.com/FORTH-ModelBasedTracker/MocapNET
- AUTO-MNET, developed during 2022 and hosted on repository: gitlab.com/bonseyes/bonsapps/1st-support-programme/automotive/body-part-tracking/auto-mnet/bonsey
- MocapNET 4, published in [31] and hosted on the mnet4 branch of repository: github.com/FORTH-ModelBasedTracker/MocapNET
- Robot Operating System Module hosted on repository: github.com/FORTH-ModelBasedTracker/mocapnet_rosnode

A.3 Implemented Systems and Models

Their functionality involves:

- Real-time 3D Body Pose Estimation
- Real-time 3D Hand Pose Estimation
- Real-time 3D Eye Gaze Estimation
- Real-time 3D Facial Capture Estimation
- ESP32-CAM Streaming Demo
- WebRTC Streaming Mobile Phone Demo
- A BVH parser and CLI toolkit written from scratch in C
- A Linear algebra library written from scratch in C
- An OpenGL skinned mesh renderer written from scratch in C
- A MocapNET runtime written in C/C++
- A MocapNET training and runtime written in Python
- Blender plugins for body/face animation using Makehuman skinned models
- Integration with the BonsAPPS marketplace
- Integration with Robot Operating System
- Integration with arbitrary 2D Joint sources through CSV/JSON files
- Integration with Google mediapipe
- A variety of visualizations and data processing tools where developing that where used to produce the visualizations and graphs shown in this document.

Trained models are currently hosted on :
https://ammar.gr/mocapnet/mnet4/database.php although after the completion of this work a more permanent solution will need to be provided for their archival and preservation.

Appendix B
Applications

The development of this thesis had the good fortune to temporarily coincide with various research projects involving the Computer Vision and Robotics Lab (CVRL) of FORTH that required 3D human perception. These projects provided real-world experiments that gave useful feedback that helped shape the method, inspired many of its design characteristics and provided funding for the execution of its research. We will attempt to briefly list them while also providing some qualitative results from each project.

B.1 Ambient Intelligence, Confluence and Smart Environments

Having the privilege of conducting the presented work in the premises of the Ambient Intellicence (AmI) building [490] of the Foundation for Research and Technology (FORTH) in Crete, as well as studying Human Computer Confluence under the HY-569 course of professor C. Stefanidis as part of my PhD studies, the use of the presented work as a means to further and improve the capabilities of the AmI project was always a core consideration during its development. Even early versions of the method showed great promise [491] so using it to improve existing human sensing capabilities was an important priority.

The AmI Facility (Figure B.1) occupies a two-floor building of $3.000m^2$, comprising of simulated AmI-augmented environments and their support spaces, laboratories, offices and public spaces. The simulated environments constitute full-scale replicas of their real-life counterparts. The term "simulated" is employed since (a) these spaces are integrated in the overall AmI facility; (b) they are constantly monitored through various sensors, observation and logging mechanisms and (c) comprised of a mix of experimental prototype "sandboxes". An example of an AmI "Sandbox" (Figure B.2), is a laboratory space simulating a "home" of about $100m^2$ comprising of six rooms, in each of which various AmI technologies and applications are developed, tested, and exhibited.

The current solutions deployed for user-sensing are based on multiple modalities ranging from microphones, wearable devices, pressure, proximity and PIR motion/presence sensors to touch screens and "smart-surfaces" with cameras and computer vision algorithms formulated to track specific objects or motions to facilitate HCI. High-level 3D pose estimation is mostly conducted using RGBD sensors, that while providing accurate 3D observations, introduce a number of complications. First of all RGBD devices like ASUS-Xtion have availability issues and a high cost (300$ at the time of writing). Their resolution is VGA (640x480 @ 30Hz), their operation range is strictly between 0.5m and 4m, they use stale pose estimation software (OpenNI2.0 is now 11 years old), due to their USB controllers and driver stack they require USB cables with less than 5m between the sensor and the PC that performs local processing.

These requirements mean that typically each of the deployed RGBD cameras needs a dedicated computer to operate it, thus as many computers as the operating cameras are needed. This duplication translates to increased first purchase costs, administration and maintenance costs as well as electricity consumption for the operation of the systems, with the bulk of processing power available to support user activities being wasted when no users are present, since it cannot be re-diverted according to the needs of the user. These issues are highlighted on Figure B.3 showing a smart space in the main hall of the AmI building.

Our proposal for replacing the current 3D human pose estimation is summarized in Figure B.4, a net-

Figure B.1: The AmI building hosts a number of different environments to allow the study and development of technologies that allow automating or supporting daily life activities, as well as enhancing security and safety, health and communication.

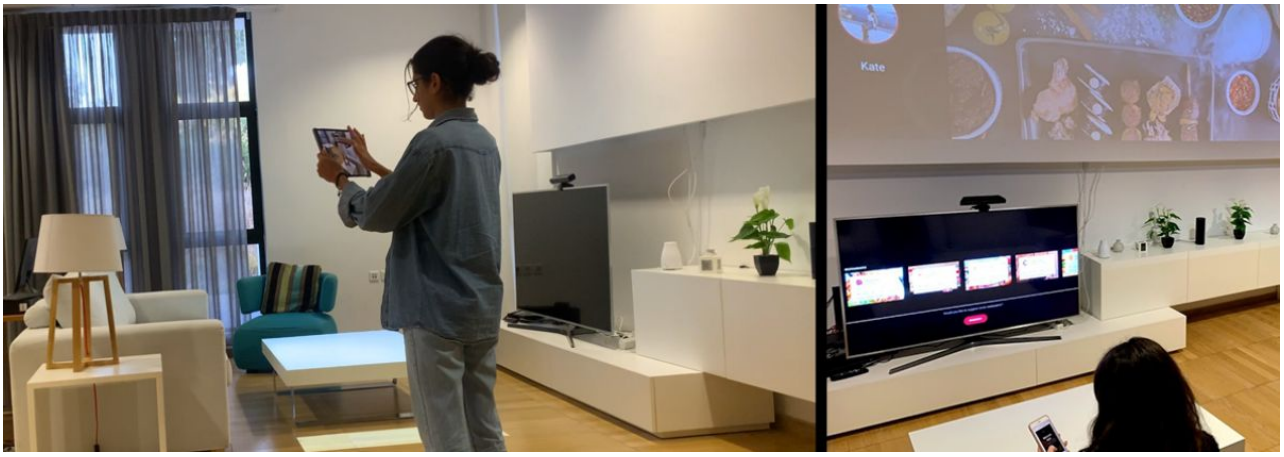


Figure B.2: AmI research efforts focus on specific simulation “sand-boxes” that allow intuitive and adaptable user interfaces which can be ubiquitous everywhere in the environment. Our presented work naturally facilitates user sensing allowing for richer and more immersive HCI experiences.

worked server architecture connected to a great number of relatively inexpensive IP cameras from various vendors. These can range from ESP32-Cam sensors that cost around 5$ (at the time of writing this thesis) and can provide WiFi streaming of JPEG images acquired using an OV2640 camera sensor of resolution of 1600×1200 @ 15 fps, SVGA @ 30fps or CIF @ 60fps, and up to higher end commercial IP cameras that feature pan and tilt motors and allow advanced video stream encoding standards that preserve network bandwidth.

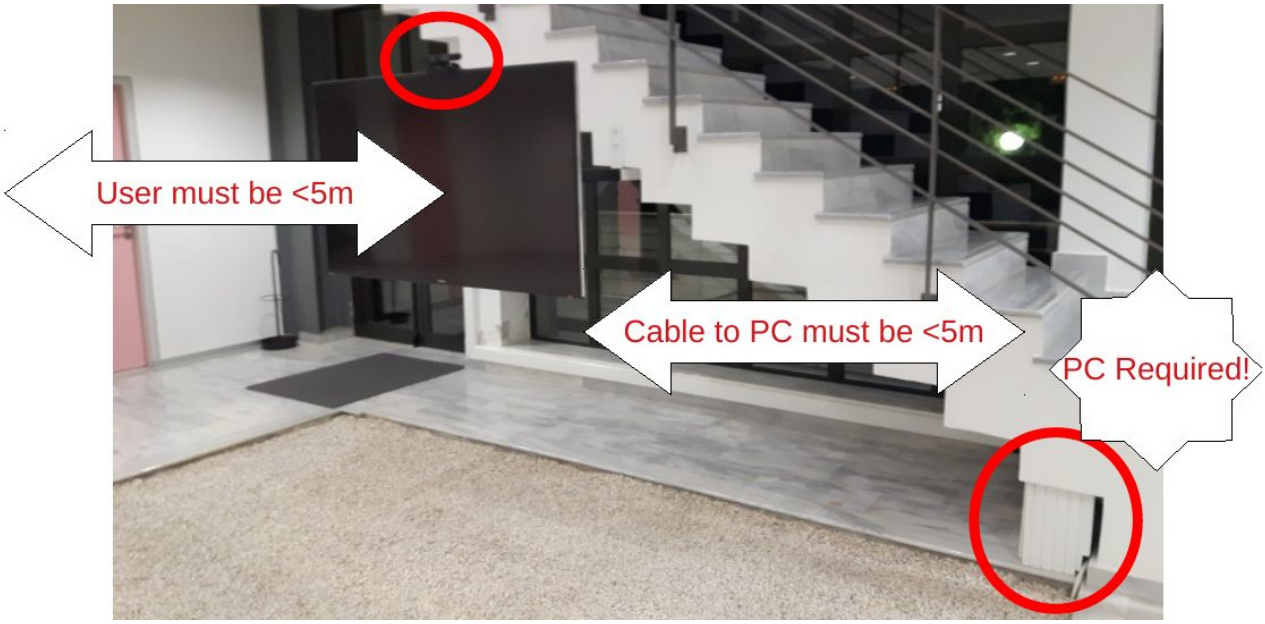


Figure B.3: Currently AmI human pose estimation is mostly performed using RGBD Sensors that introduce a number of aesthetic, cost, software and hardware constraints for their deployment.

IP cameras come from a great variety of manufacturers thus being easier to support, maintain, replace and avoiding vendor lock-in. They can accommodate both indoor and outdoor use cases since in contrast to RGBD sensors they do not rely on infrared structured light for depth perception. They can be adapted to different scenarios using different optics, are easier to conceal for aesthetic reasons due to their reduced size, while most importantly they completely disentangle image acquisition from computer vision since the bulk of the image transport task is handled by the TCP/IP or UDP/IP network (instead of USB). They use well defined open-standards and the perception stack can be centrally deployed on a single server. Upon updating the neural networks with new features, bug fixes and novel capabilities will be immediately equivalent to updating all of the different devices on a one by one basis, something that is very complicated with the current setup. Having all of the processing power available in a single machine for the vision tasks means combining the processing capabilities of each independent computer in the RGBD scenario of one computer per sensor, and "focusing" computations only on active streams, thus achieving true confluence of computation and human activity.

The prototype developed consisted of a user generated map with the location of each of the artifacts hardcoded into the system. Newly proposed methods like Gaussian Splatting [492] could facilitate acquiring a high quality 3D reconstruction, by moving the camera in the room before installation, while a Language embedded radiance field (LERF) [493] could combine a model like CLIP [57] to automatically segment and populate the 3D scene with all visible artifacts. Having a map of all of the available artifacts and areas of interest, the 3D pose estimation system projects the current 3D pose and resolves vectors for motion, attention, and hand pointing to the available artifacts in the scene, while a small number of gestures like clapping, t-posing, waving, circling, kicking, pushing, or crossing arms can be detected by thresholding the BVH files against preset configurations and then emitting high-level "activation" events to the artifacts that are involved.

Due to COVID-19, deployment of the method to the actual AmI building was not possible. A scaled

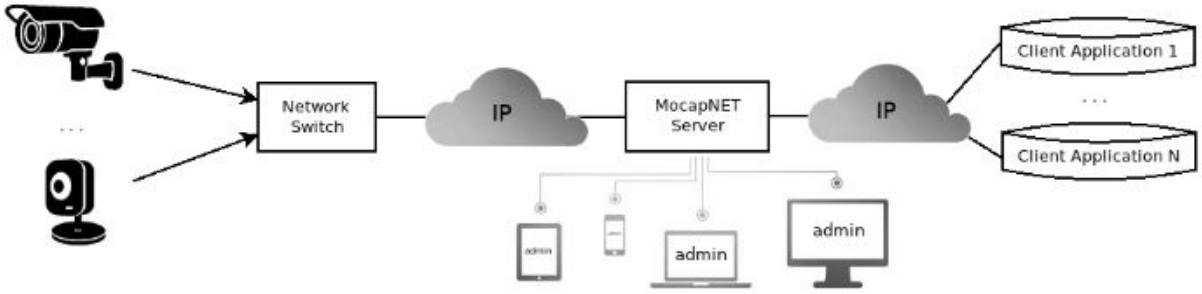


Figure B.4: Omni-MocapNET was proposed as a versatile 3D human pose estimation solution for AmI, to upgrade the existing RGBD based one. A network system of inexpensive wired and wireless IP Cameras would be connected to a centralized "MocapNET" server. Based on the presence of activity in the video streams, the server will apply its resources to pose estimation on active cameras, broadcasting 3D pose estimation to all active client applications. 3D human perception would dynamically shift from room to room to each specific cameras, while sharing the same computational resources and provide higher accuracy pose estimation at a fraction of the cost required with the current setup (Figure B.3), where each vision-capable "artifact" has its own dedicated computer.

down prototype was thus implemented in the author's home. Figure B.6 shows snapshots of video [48]. Two different lamps and an electric Fan get automatically activated based on the 3D location and orientation of the user to illuminate his path and provide him with cooling based on ambient temperature. After that, the user stands in front of a screen and looks at it, performs gestures to alternate on-screen images, performing a gesture he starts the music player which he also controls by waving his hands.

The proof of concept application of MocapNET to the AmI context was relatively successful, although in an admittedly limited scope. Although this effort did not directly lead to a publication, we believe that it shows a lot of promise for improving and upgrading the capabilities of the current systems. The Omni-MocapNET prototype was partially implemented by adding the ability to perform RGB to 2D regression in different machines than MocapNET, code for streaming JPG originating from ESP32 Camera (Figure 3.46) images, a system to define artifact maps and use them with the perception loop as well as an HTTP based server for publishing and subscribing to events. Although a fully-fledged system would need additional modules, we encountered no prohibitive factors for an actual Omni-MocapNET implementation.

B.2 Humanoid Experiments and ROS Integration

Having a decomposition of human motions in a hierarchical tree of joint rotations (Figure 1.4) in real-time, provided a unique opportunity to directly use the data as a control scheme for humanoid robots. Humanoid robots like NAO (Figure B.8) rely on a fixed dimension armature with links consisting of servo motors with well defined limits, torques and affordances. Humanoid robots mimic the human body, so our BVH armature naturally encodes pose compatibly allowing direct manipulation of the servos by the onboard controllers to achieve the poses observed by our vision system.

Due to prior experience, using our WACV18 work [215] and being able to successfully manipulate and control the upper-body of the robot while using a stabilization controller for the lowerbody so that the robot remains stable even in cases of rapid accelerations of the torso and hands, leading the robot out of balance (Figure B.9).

Figure B.5: During February of 2020 Amazon Go opened its first $1000m^2$ grocery store without cashiers in Seattle. This coincided with the COVID-19 pandemic and the limitation of human contact as a mitigation strategy to contain the virus, as well as the development effort for our MocapNET 2 [8] work. The supermarket application needs for human pose estimation and tracking are very similar to the needs of the AmI space. This development cemented our conviction on exploring this potential research direction.

Figure B.6: Omni-MocapNET prototype developed in the authors home due to COVID-19 quarantine. The application allowed the creation of different artifacts/interest regions and relayed signals to operate devices based on the user position, gestures, body orientation, and hand pointing direction, such as the lamps and fan in the illustration. Video available in [48].

The operation of the NAO robot had already been successful prior to this work. However in contrast to prior works this time, we created a ROS [494] wrapper that allowed seamless integration with robotics projects while providing our regressed armature using a TF-transform tree. This not only targeted a single use-case like the NAO robot but allowed 3D human perception from arbitrary robots, one of which was once again the NAO robot [495]. Instead of directly experimenting with the robot an intermediate simulator was employed to also allow for self-supervised stability optimizations at a later point. The simulator RAISIM (Figure B.11) was used towards this end since it featured real-time operation, many provided bipedal humanoid models, an easy programming interface and a long list of successful publications using it as their

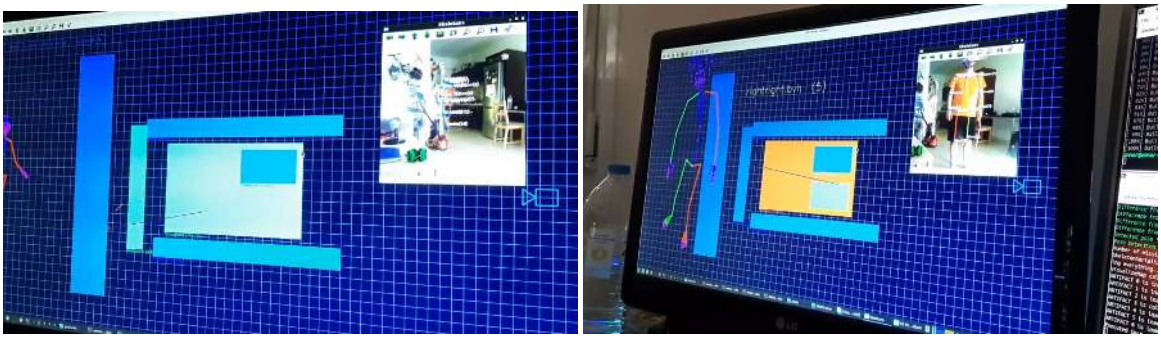


Figure B.7: Visualization of the Omni-MocapNET prototype “map” with 3D artifacts that are “selected” as the user walks, looks and points in the environment, triggering various actions that anticipate and facilitate his/her needs.

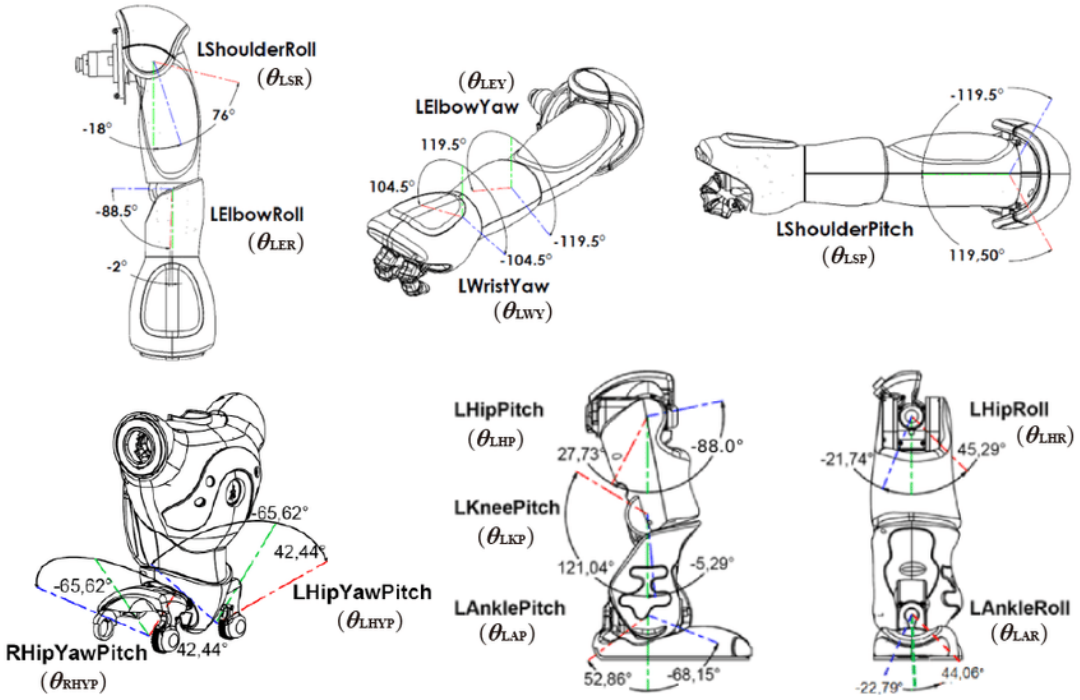


Figure B.8: NAO robot schematics listing the degrees of freedom of the robot along with their various joint angles limits.

physics engine.

Results using the RAISIM simulator where largely successful allowing MocapNET to act as a pose controller for the simulated robot while also allowing us to measure and record forces exerted between the robot and the floor. Sadly while the original plan was to use the SEROW motion planner developed by Stylianos Piperakis to learn a physics compliant bipedal motion controller using visual cues from human gait observations during the development of this idea Stylianos transitioned to a new position in Agility Robotics, which

Figure B.9: Tele-operation of NAO robot using visual data streamed from a laptop with a connected camera via WiFi in real-time.

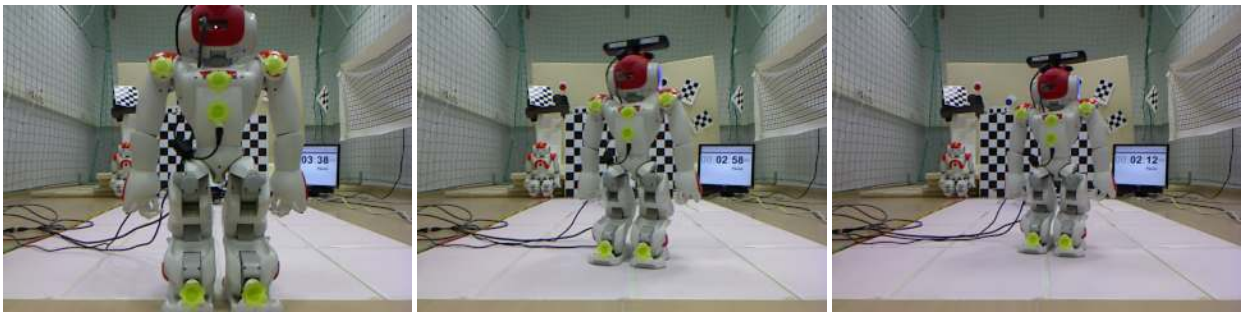


Figure B.10: Attempts to record actual ground truth on millimeter paper using a series of markers and a pair of active RGBD cameras for NAO experiments.

led to the abandonment of the effort of integrating physics and the direct robot teleoperation module to the MocapNET project.

B.3 Experimental Results from Mingei

Mingei was a Horizon 2020 EU project exploring the possibilities of representing and making accessible both tangible and intangible aspects of craft as cultural heritage (CH). Heritage Crafts (HCs) involve craft artifacts, materials, and tools that encompass craftsmanship as a form of Intangible Cultural Heritage. Intangible HC dimensions include dexterity, know-how, and skilled use of tools, as well as, tradition, and identity of the communities in which they are, or were, practiced. HCs are part of the history and have impact upon the economy of the areas in which they flourish. The significance and urgency to the preservation of HCs is underscored, as several are threatened with extinction. Despite their cultural significance efforts for HC representation and preservation are scattered geographically and thematically. Mingei strived to provide a means to establish HC representations based on digital assets, semantics, existing literature and repositories, as well as, mature digitization and representation technologies. These representations will capture and preserve tangible and intangible dimensions of HCs.

Central to craftsmanship is skill and its transmission from master to apprentice. To digitize this Mingei attempted to capture the motion and tool usage of HC practitioners, from Living Human Treasures and archive documentaries, in order to preserve and illustrate skill and tool manipulation and preserve European history.

A variety of techniques was used in the effort to digitize motion including a motion capture suite from

Figure B.11: Snapshots from the RAISIM simulator website. RAISIM is a closed-source cross-platform multi-body physics engine for robotics and AI used for multiple state of the art projects [49–53].

NANSENSE. The project documented three tasks, namely the craft of Silk weaving, Glass making and Mastic gathering. The presented method, was used to track and digitize 3D human motion of craft experts. The Silk weaving task was initially tackled using the MocapNET 1 method [6] while the Glass making coincided with MocapNET 2 [8] and Mastic trials during MocapNET 3 [9] development.

The particular project did not have real-time operation constraints and thus MocapNETs strong point, real-time speed did not play any role and satisfied no requirements. On the other hand, high accuracy of recorded motions was very important in the context of this project, providing motivation and prompting development towards this direction. The HCD algorithm usage with high fine-tuning budget was among various improvements conducted over the course of the thesis as a response to the challenges of this project.

B.4 Experimental Results from SustAGE

SustAGE was an EU-funded Horizon 2020 research project aiming to develop a smart person-centered solution with the goal of supporting employment and later retirement of older adults from work while promoting the concept of sustainable work for EU industries.

The project aimed towards a paradigm shift in human machine interaction, building upon seven strategic technology trends, IoT, Machine learning, micro-moments, temporal reasoning, recommendation systems, data analytics and gamification to deliver a composite system integrated with the daily activities at work and outside, to support employers and ageing employees to jointly increase well-being, wellness at work and productivity. MocapNET was used in the context of the project to accommodate markerless 3D human pose estimation in an industrial setting with the goal of tracking activity and also monitoring for anomalies and accidents, while also investigating if the recovered BVH configurations and the range of motion of workers could be used to predict and prevent repeated stress injuries from repetitive or extreme motions.

The use of our method with respect to the sustAGE project was successful, with the biggest problem being the large and repeated occlusions due to the interaction of the subjects with a variety of on-scene objects. As we can see in Figure B.16 while assembling a door the subject becomes severely occluded when behind the assembled door. Although the method regresses a plausible pose that fits our training set, a relatively straightforward to implement idea to further improve results would be to use multiple cameras from

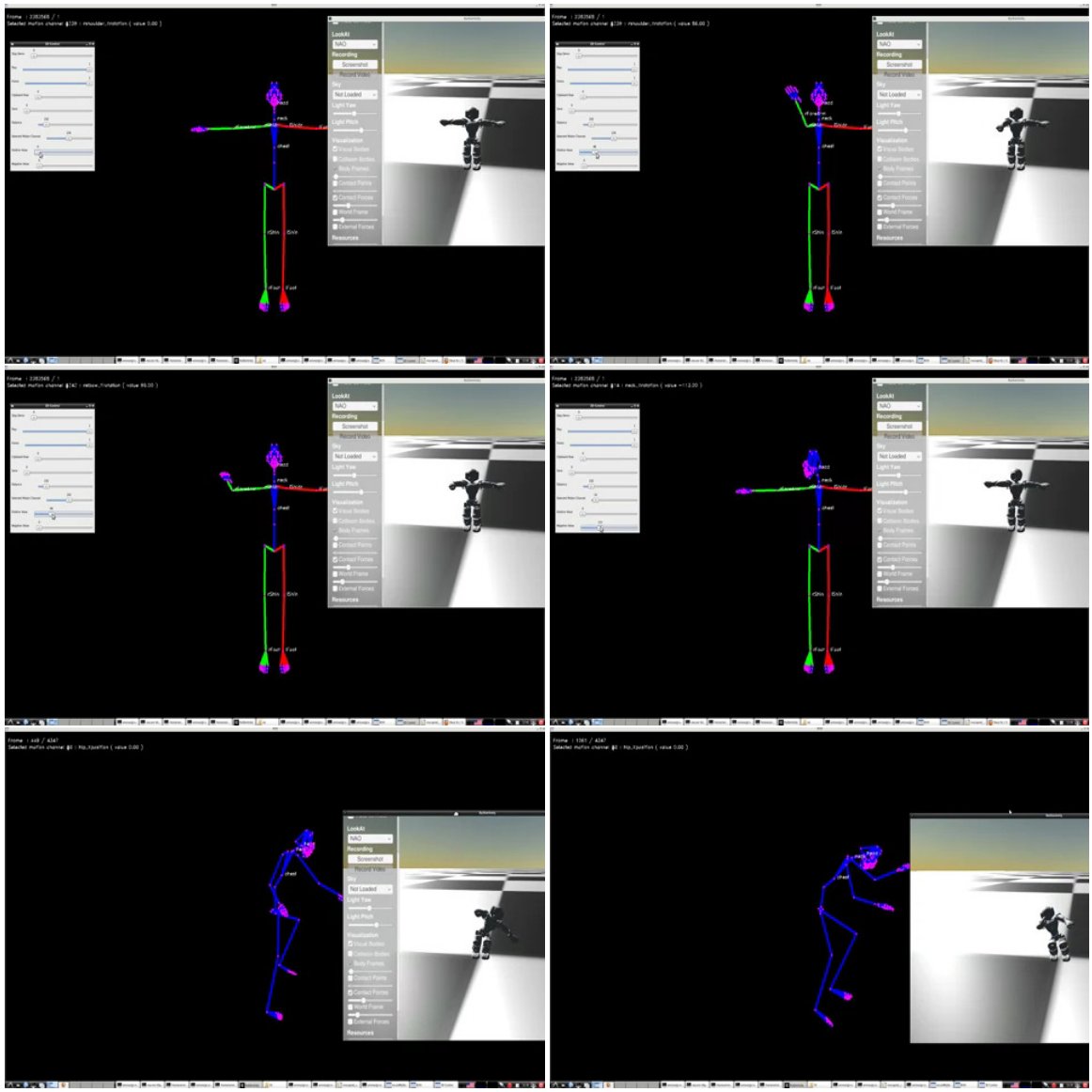


Figure B.12: Direct manipulation of the NAO Robot by mapping the MocapNET BVH armature to a simulated Robot using the RaiSim simulator.

different vantage points, each of which would have different occluded joints and either perform a visual hull analysis for the 2D/3D joints available revealing occluded joints, or even regress a BVH skeleton from each camera and then share the regressed degrees of freedom based on the 2D visibility of joints. For example a camera not seeing the right arm should not trust its 3D regressed output, while a camera not seeing the left leg should also ignore it. By combining the two BVH frames and keeping the "visible" information with low 2D reprojection errors one could systematically overcome this problem.

Figure B.13: Qualitative results from Mingei from Mastic harvesting videos recorded in Chios, Greece and tracked using **MocapNET 3** [9].

B.5 Experimental Results from SignGuide

The right to equal access to cultural-educational content is described in the European Charter of Human Rights. However, in practice the availability of museum cultural content that is accessible to the deaf is minimal to non-existent.

The object of the SignGuide project was to develop a prototype interactive museum tour system for deaf visitors using mobile devices that will be able to accept visitor questions in their native (sign) language in relation to the exhibits, and provide additional content also to sign language using avatars or videos, leveraging techniques from computer vision and machine learning.

The system was intended to be used in the Archaeological Museum of Thessaloniki (AMTH), with integration and commercial exploitation done through the Greek company MLS Informatics (MLS). The development of the monitoring system and the virtual tour guide was partly developed using the presented method to provide motion capture for the animated avatar guide.

The SignGuide project provided many lessons and inspired many important changes to our work. The RGB sign language datasets recorded by the consortium (Figure B.17) did not contain the legs on the camera view. This seemingly insignificant detail, highlighted an important problem of the original formulation of our MocapNET 1 [6] method that treated the whole body using a single NSDM descriptor. Having the lower-body occluded had very detrimental results for the ensemble as seen in Figure 3.15, prompting one of the basic design novelties of MocapNET 2 [8]. It also motivated the generalization of the body pose estimation method to the 3D hand pose estimation problem which was one of the core features of MocapNET 3 [9].

Other problems encountered during the project were the issue of 2D/3D symmetries, (Figure 3.13) leading to incorrect and unnatural hand orientations, despite our method overall following well the motions of the actor. The project also showcased real-world complexities of the use of MOCAP with existing tools/con-

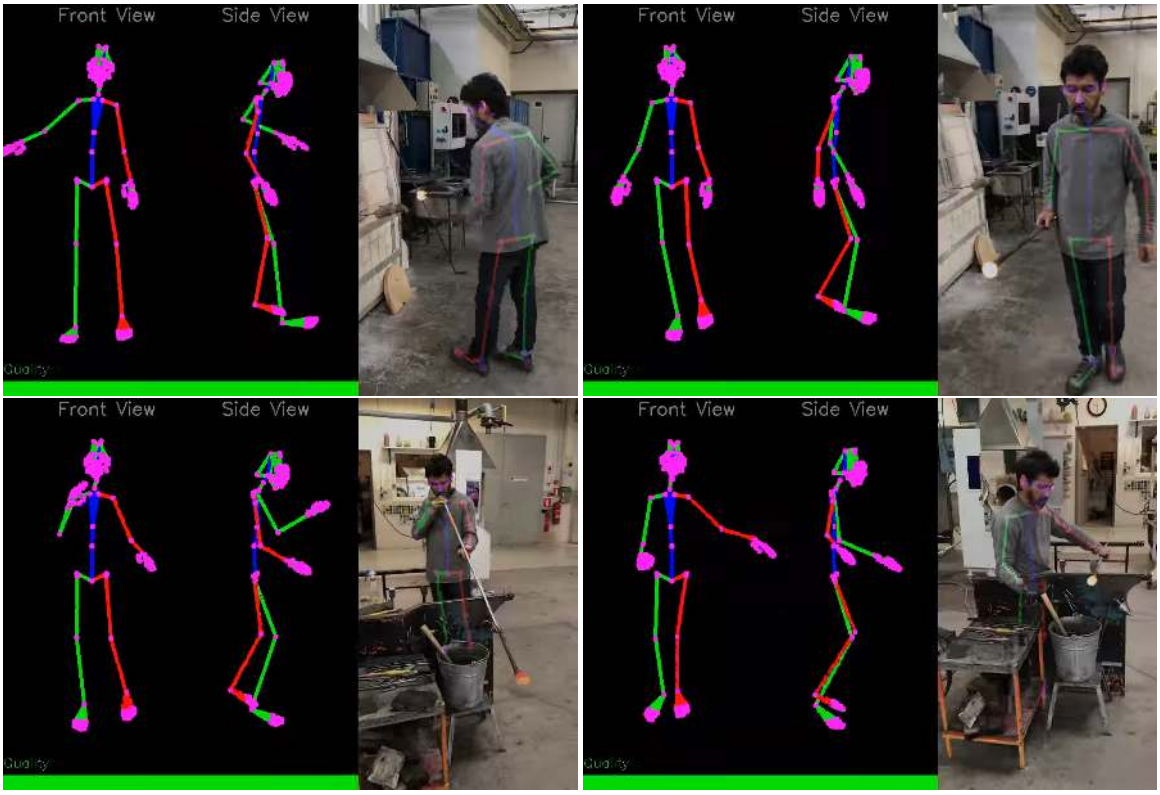


Figure B.14: Qualitative results from Mingei from Glass blowing videos recorded in Cerfav, France and tracked using **MocapNET 2** [8].

verters and graphics engines that can be understood better by examining the chain of steps required during the project.
To process videos from the SignGuide project, the following sequential steps were required:

1. Record a sign video.
2. Convert the video into individual images.
3. Pass each image through a 2D estimator (OpenPose [4]/BlazePose [34]) and convert each RGB/JPG image into a JSON and then CSV file with normalized 2D coordinates.
4. Run MocapNET and extract a BVH file for each frame.
5. Run the Hierarchical Coordinate Descent Algorithm for each regressed pose for any improvements with regards to the 2D observation provided by the 2D estimator.
6. Concatenate the BVH frames to a BVH output file.

At this point the BVH file is ready to use and can be consumed by many applications, while we also provided our own rendering solution using a combination of Makehuman and BLender as seen in this video [496]. However, due to other requirements of the project the Unity environment was used while also

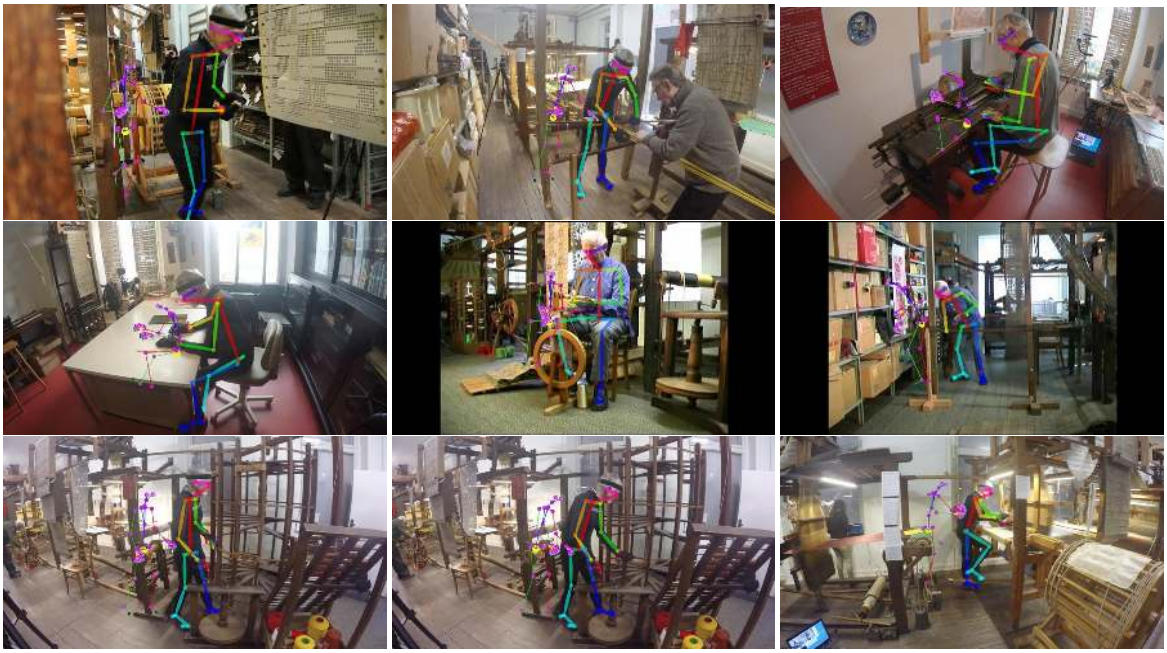


Figure B.15: Qualitative results from Mingei Silk heritage videos recorded in Krefeld, Germany and tracked using **MocapNET 1** [6].

requiring additional steps which were not immediately obvious since Unity is a closed source graphics engine and most of its contracted libraries as well, so detailed specifications are not easy to find for appropriate data conversions.

After a long series of testing we the following steps where required to properly import the BVH file onto Unity:

7. Convert the MocapNET BVH file to a cut-down BVH armature derived from reverse engineering Unity compatible FBX files available online which was a combination between the middle and rightmost armatures seen in Figure 3.18.
8. Reading the recalculated BVH file in Unity
9. Running a C# script developed by a colleague to convert the BVH file to Unity's internal skeleton representation
10. Exporting the animation from Unity's internal representation to an FBX file (ASCII/Binary)
11. Re-importing the FBX file after restarting Unity
12. Final rendering of animation with Unity 2021.3.10f1 from FBX file with a digital avatar created with Reallusion Character Creator 3.44

During conversions, we encountered the following problems, which we will list, highlighting the stage at which we recognized them:

- During stage (3) the 2D detector OpenPose and BlazePose often output noisy or incorrect 2D coordinates for body parts and especially fingers. As the MocapNET method is based on 2D coordinates,

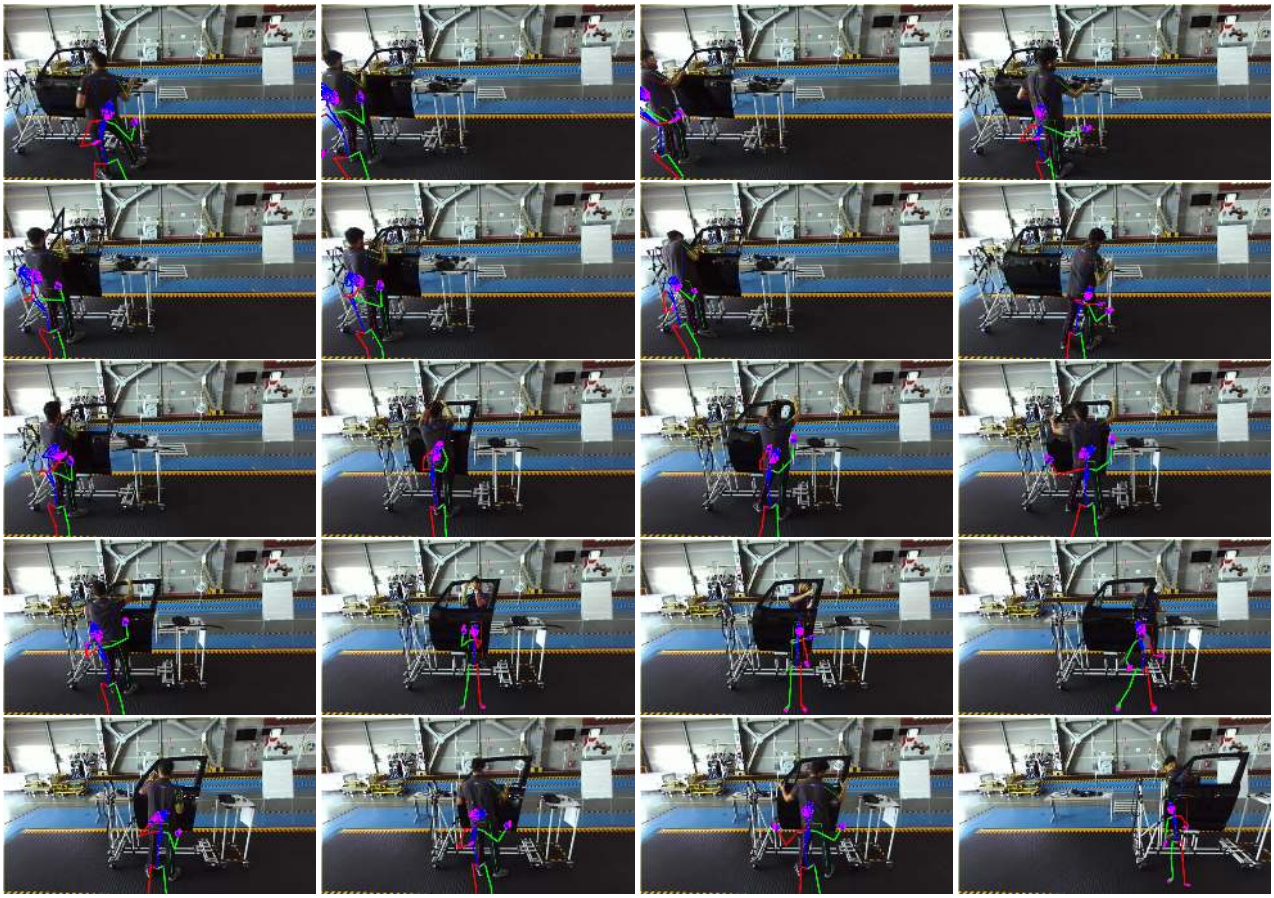


Figure B.16: **MocapNET 2** qualitative results on a SuSTAGE dataset monitoring a factory floor and an engineer performing assembly of a car door.

wrong input produces wrong output. In our effort to improve these problems we introduced constraints on the body, hand and face kinematics and a Butterworth filter to smooth the noise where it is possible.

- During stage (4) there was ambiguity between 2D arrangements which could come from both a forward extended arm and a backward extended arm. Mathematically, the input was the same, so both solutions where equally "correct". This problem was exacerbated as the CMU BVH dataset conversion used to train the network appeared to contain poses where the arms bend backwards. These poses where identified, isolated from training and are reported in this thesis in Section 3.3.2. To mitigate the symmetry issue we experimented with a loss penalty heuristic during HCD for solutions that moved hands backwards (since for the sing-language tasks hands are typically in front of the torso). Unfortunately, these penalties in turn cause shaking of the limbs in Nash Equilibrium points where the penalty balanced out with the actual loss of the solution. An additional improvement was adding history of previous solutions to further reduce this problem.

- During the conversion (7) the final BVH file has fewer bones (e.g. in the thumb) resulting in a natural loss of the original accuracy of the regressed representation due to a lower capacity to represent poses. Trying to intervene with the Blender 3D editor on this BVH file (Figure B.17) we notice that

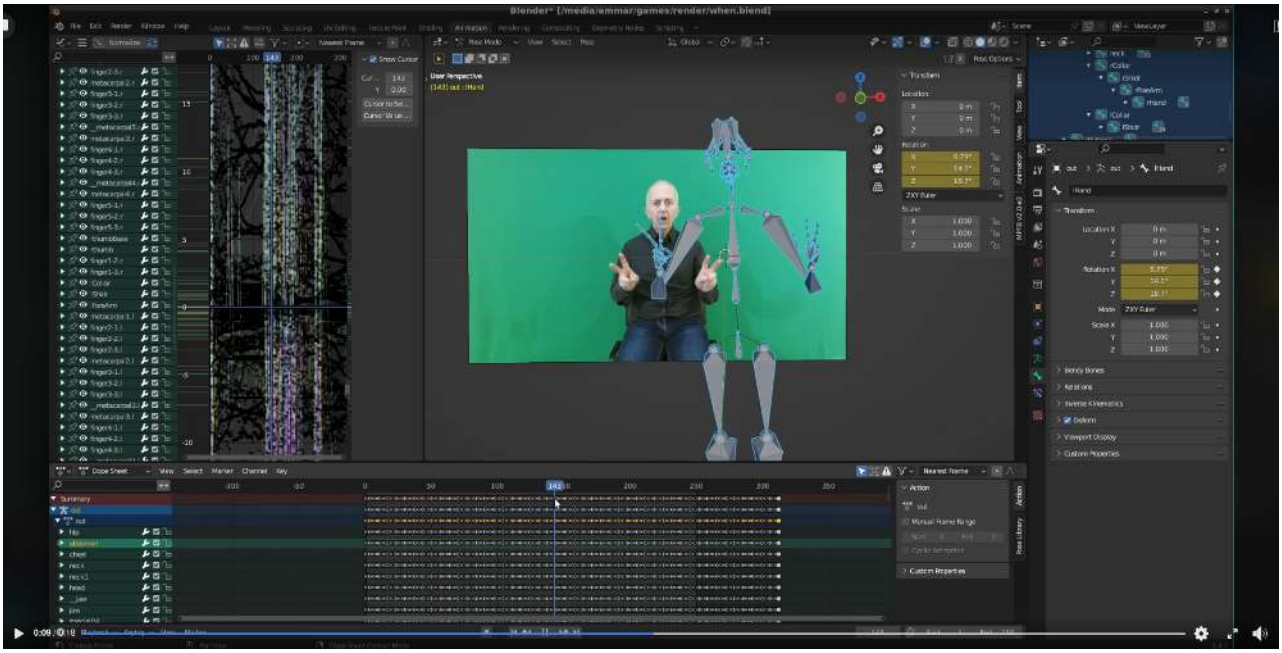


Figure B.17: **MocapNET 3** results from SignGuide loaded on Blender.

the coordinate system changes resulting in further problems during the next steps. It is worth noting that looking at Blender's BVH import code we also found problems that we reported 6 months ago (https://projects.blender.org/blender/blender-addons/issues/104549) unfortunately without receiving a response at the time of writing these lines.

- During conversion (9) we visually observed a small movement of the points and a slightly different movement than in the previous BVH representation. Because of the closed source implementation of Unity we could not trace where this discrepancy originated from and the procedure with which the internal skeleton is created and calculated.

- ASCII/Text FBX files need proprietary AutoDesk closed-source loaders. They are not directly compatible with 3D editors like Blender. By downloading the AutoDesk FBX-SDK-2020-0, it was possible to manually convert an ASCI FBX to Binary FBX, import into Blender, however after re-exporting from Blender the file no longer was importable into Unity.

- During the final playback, after successfully performing Steps 1-12 we notice that accumulated discrepancies by all of the above significantly alter and degraded the movement produced by the final model.

B.6 Experimental Results from BonsAPPS

The BonsAPPs project had the ambitious goal of creating an AI marketplace to allow researchers and developers to create AI solutions to solve real-world industry challenges, submit their code to an end-to-end containerized, ready-to-integrate and streamlined system inter-operable with the AI4EU platform. Stakeholders, industrial partners and AI consumers could then have direct access to the provided AI solution thus benefiting all parties involved.

The open call of BONSAPPS (https://bonsapps.eu/) for AI talents received 126 proposals from 31 EU countries. Out of these proposals, 30 were actually accepted. Out of the 30 accepted BONSAPPs projects

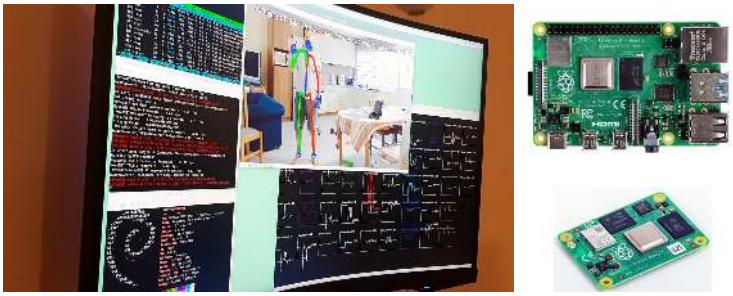


Figure B.18: Left: PCA compressed **MocapNET 3.5** body tracking results running on a Raspberry Pi4 with 2GB RAM (Right) using the TF-Lite runtime.

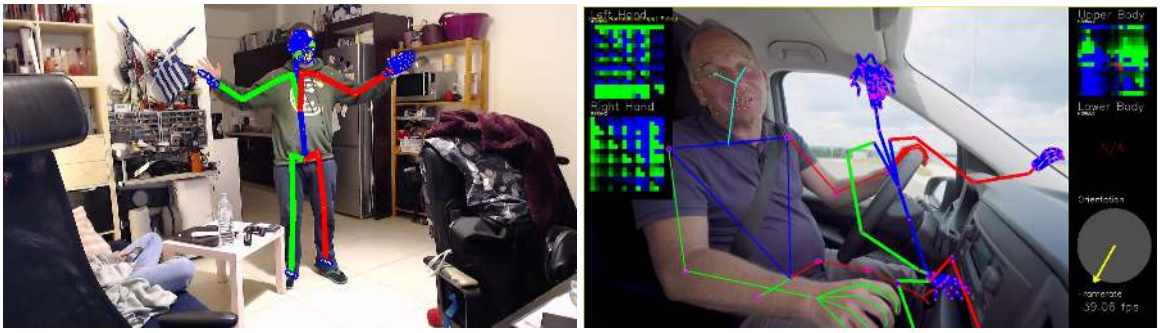


Figure B.19: Qualitative instances of PCA compressed **MocapNET 3.5** body tracking results up, experiments with body+hands down, since this implementation was meant to be used in automotive scenarios to monitor the driver.

after a contest the top-10 where chosen to complete the program, with the proposed work being chosen as one of the top 10 finalists. The call targeted automotive applications for passengers, thus the specific "flavor" of MocapNET, deployed as part of the platform was coined AUTO-MNET.

B.7 Oculus/Meta Quest VR Tracking

During the first year of development of the presented work, Oculus (now Meta) on May 21, 2019 released the first-generation of the Quest VR headset. The device was revolutionary being able to work tether-less and provide immersive virtual reality experiences while also performing hand tracking using the on-board cameras. Its successor Quest 2, was released on October 13 2020 and provided better hardware allowing for improved framerates and user experience.

After decades of research, this with affordable tether-less VR headsets purchased by millions new users due to the COVID-19 pandemic lock-downs prompted us to study use of our method as an auxiliary sensing modality to complement the Quest. By using a mobile phone, laptop or PC with a regular webcam running MocapNET, BVH full body+hand poses could be streamed to the headset to improve the VR-avatar. Full-body pose estimation is a widely criticized [497] still a lacking feature on the Quest platform as well as hand estimation when hands are outside the relatively narrow view of headset cameras forcing popular new VR platforms like VRChat [498] to adopt bulky and expensive active tracking systems [499].

Our work during MocapNET 3 [9] and after considered the Quest platform as one of the core use-cases. To provide proof of concept we conducted many experiments successfully tracking VR activities with some

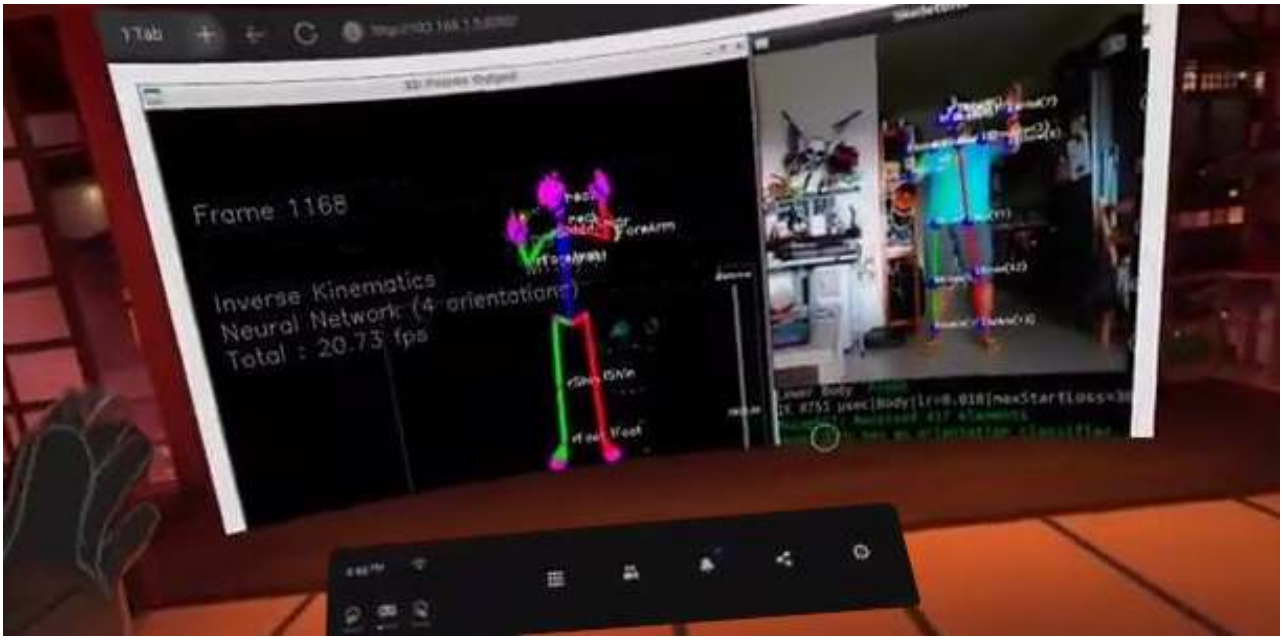


Figure B.20: **MocapNET 3** running on a desktop PC and streaming pose data to Oculus Quest 2.

Figure B.21: **MocapNET 3** qualitative tracking snapshots. Left: Oculus Quest 2 hand activity, view from the headset. Right: A PC observing the scene with a camera with MocapNET running extracting BVH regression and overlaying the 3D skeleton and skinned model next to the input.

indicative results shown here.

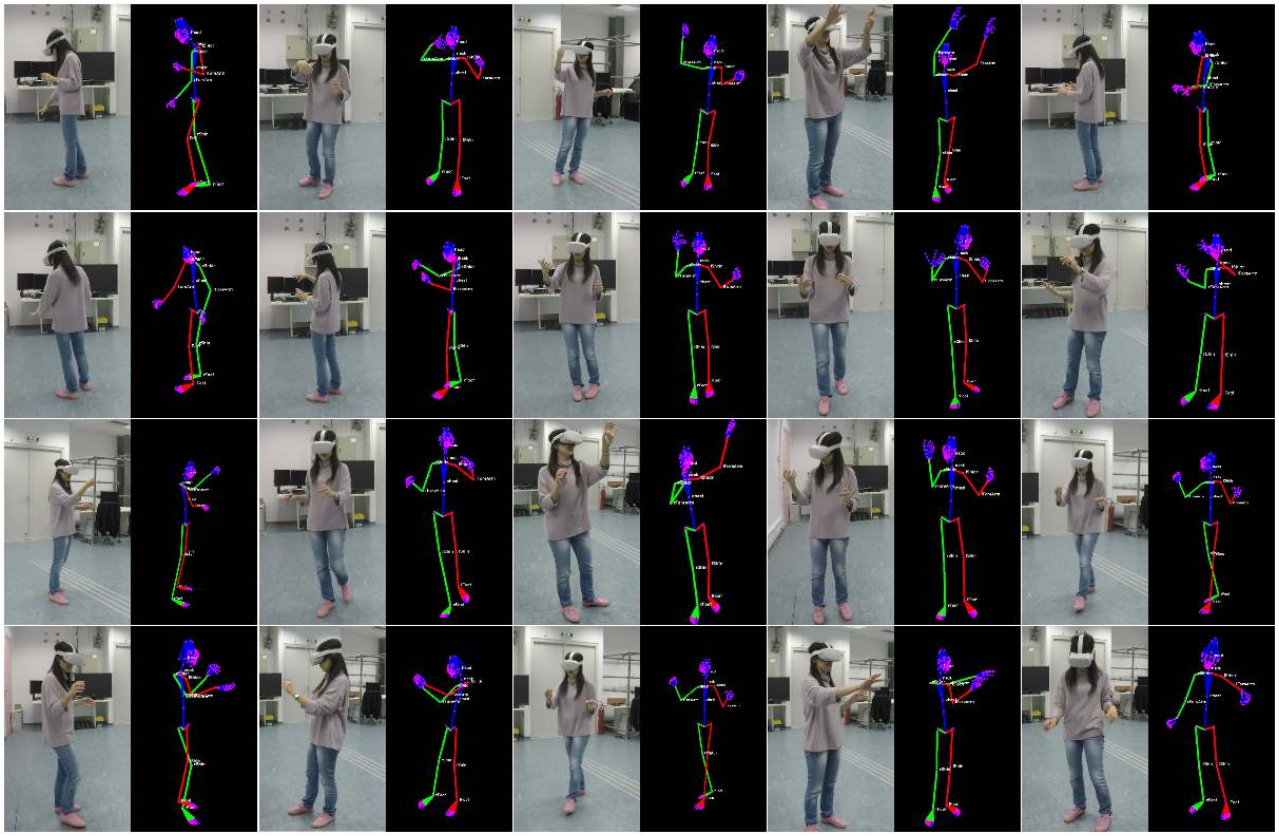


Figure B.22: **MocapNET 3** body+hand tracking results on the “Elixir” application running on Oculus Quest 2.

Appendix C
Acronyms

2D Two Dimensional
3D Three Dimensional
FPS Frames Per Second
AR Augmented Reality
VR Virtual Reality
IK Inverse Kinematics
DOF Degrees of Freedom
HOI Hand Object Interaction
MPJPE Mean Per Joint Position Error
CPU Central Processing Unit
GPU Graphics Processing Unit
GPGPU General Purpose Graphics Processing Unit
SIMD Single Instruction Multiple Data
RPI4 Raspberry Pi 4 Computer
CMU Carnegie Mellon University
FORTH Foundation for Research and Technology, Greece
ICP Iterative Closest Point
PSO Particle Swarm Optimization
HCI Human Computer Interaction
HPE Human Pose Estimation
BVH Bio Vision Hierarchy motion capture format
CSV Comma Seperated Values
RGB Red Green and Blue, color image
RGBD A Red Green and Blue color image accompanied by a Depth map
SIFT Scale-Invariant Feature Transform
SURF Speeded up robust features
SDE Stochastic Differential Equation
GASF Gramian Angular Summation Field
GADF Gramian Angular Difference Field

MAE Mean Average Error
MSE Mean Squared Error
MQE Mean Quad Error
NSDM Normalized Signed Distance Matrix
NSRM Normalized Signed Rotation Matrix
eNSRM enhanced Normalized Signed Rotation Matrix
RNN Recurrent Neural Network
LSTM Long short-term memory
PDF Probability Density Function
BPTT Back Propagation Through Time
OHME Online Hard Example Mining
PoA Price of Anarchy
AmI Ambient Intellicence
LERF Language embedded radiance field
MAC Multiply-Accumulate
MAD Multiply-Add
NaN Not a Number
LR Learning Rate
RL Reinformenent Learning
SL Supervised Learning
PCA Principle Component Analysis
HCD Hierarchical Coordinate Descent
MOCAP Motion Capture
YOLO You Only Look Once
LLM Large Language Models
AI Artificial Intelligence
MLP Multi-Layer Perceptron
ViT Vision Transformer
NN Neural Network
PTN Polar Transformer Networks
STN Spatial Transformer Networks
CNN Convolutional Neural Network
GAN Generative Adversarial Network
NLP Natural Language Processing
API Application Programming Interface
ONNX Open Neural Network eXchange
SaaS Software as a Service
EU European Union