Rheinland-Pfälzische
Technische Universität
Kaiserslautern
Landau

Monocular Visual Human Pose Parameter Estimation under Partial Observations

Thesis approved by
the Department of Computer Science
University of Kaiserslautern-Landau
for the award of the Doctoral Degree

Doctor of Engineering (Dr.-Ing.)

to

Mr. Pramod Narasimha Murthy

Date of Defense: 24.01.2025

Dean: Prof. Dr. Christoph Garth, RPTU

Reviewer: Prof. Dr. Karsten Berns, RPTU
Reviewer: Prof. Dr. Didier Stricker, RPTU

DE-386

Pramod Narasimha Murthy

Monocular Visual Human Pose Parameter Estimation under Partial Observations

Dissertation, February 24, 2025

Abstract

Human pose estimation, a crucial step for understanding body postures and movements in visual data, faces challenges in monocular camera setups due to inherent limitations. These include the lack of depth information, susceptibility to occlusions, and sensitivity to viewpoint changes. Despite these challenges, advancements in computer vision, especially through deep learning techniques, continue to enhance pose estimation robustness. This thesis contributes to extending human pose parameter extraction in monocular cameras, addressing challenges related to varying camera viewpoints and partial body observation. It explores pose estimation in exo-centric and ego-centric views, offering approaches to handle self-occlusion and partial visibility.

Initially, we focus on accurate estimation and synthesis of articulated human pose. Utilizing a data-driven approach, joint limits are learned from 3D motion capture datasets, facilitating the generation of synthetic datasets for training neural network discriminators. These discriminators serve as priors for human pose estimation and motion synthesis. A novel deep learning module for 3D joint angle regression is introduced, utilizing swing-twist representation. This approach, more generic than estimating 3D joint positions, allows for predicting twists around body segments, enhancing 3D human body shape estimation.

The thesis further introduces DiveNet, a diving sports motion analysis framework, specifically designed for accurate localization and segmentation of complex diving motions in videos. It includes a diving action localization network and a dive pose regression model, achieving high accuracy in trajectory and body joint angle estimation.

Furthermore, we address a car driving scenario with a static camera to monitor the driver behavior. A driver pose estimation dataset collected with common activities along with pose data. The dataset is generated in a driving simulator, encompasses diverse data types and annotations, contributing to improved accuracy in human pose estimation. A PointConvNet neural network consisting of local feature aggregation is trained on recorded dataset to demonstrate a robust 3D driver pose estimation.

Lastly, the research introduces a framework for 3D egocentric pose estimation using a head mounted fisheye camera. We introduce a triple-branch network that integrates the Waterfall Atrous Spatial Pyramid module along with effective loss functions. This network is designed to facilitate simultaneous learning of joint rotations and confidence estimation, enhancing the accuracy of both 2D and 3D pose predictions.

Overall, these contributions advance the field of human pose estimation by identifying and addressing challenges in monocular camera setups for diverse applications, and improving accuracy through innovative approaches and datasets.

Acknowledgement

I want to express my sincere gratitude to Prof. Dr. Didier Stricker for giving me an opportunity to be part of Augmented vision department. I was fortunate to learn from my excellent colleagues and students working at the department. I also want to express my sincere gratitude to Prof. Dr. Bertram Taetz for his valuable advice, motivation, guidance, and immense knowledge transfer. His mentorship helped me carry out the research work diligently and successfully. His valuable comments on my thesis work immensely improved the thesis draft.

I am grateful to all members of Augmented Vision Department for their kind support and assistance. I am also thankful for the successful and fruitful collaborations with all the colleagues and students during my tenure. I feel fortunate to have worked with Siddharth Ingle, Ankit Kumar, Arpit Lekhra, Asmita Mittal, Xue Yang, Shubham Gupta and other wonderful students who contributed towards the presented research work. My special gratitude for my friends Alex Schafer, Torben Fetzer, Tewodros Amberbir Habtegebrial, Jason Raphael Rambach, Jilliam Maria Diaz Barros and Mohammed Selim for their boundless support till the end of defense.

I also wish to extend my sincere appreciation to Mr. David Hasse. His technical expertise along with timely resolution of computing infrastructure issues and system dependencies were invaluable in ensuring a smooth research process. I am also grateful to Dr. Thomas Köthe from Institute for Applied Training Sciences (IAT), Leipzig. His guidance helped us in curation of better diving dataset. His valuable insights of high performance training of competitive diving sports helped us improve the experimental results.

I want to thank all my dear friends, Sandesh Hiremath, Arun Alapati, Shivanand Batane, Aniket Kolhapurkar, Khyati Shah, Harshal Karanjkar, Tanay Deshmukh, Avinash Dande, Navneet Singh, Gautam Gala and Milan Agrawal, for their unwavering support and encouragement throughout my academic journey. Their friendship and motivational words have been a constant source of inspiration and helped me overcome my challenges along the way.

Last but not least, I want to thank all my family for their unwavering support throughout my studies. This achievement would have been unimaginable without their constant encouragement.

Declaration

I hereby certify that I have independently authored the work with the title "*Monocular Visual Human Pose Parameter Estimation under Partial Observations*". I have completely specified the sources and tools used, and I have included the places of work - including tables and Illustrations - which have been taken from other works or the Internet in the wording or meaning, indicating the source as borrowing. Except wherever specific reference is made to the work of others, the contents of this thesis are original. I have not submitted my work for consideration for any other degree in this, or any other university. This dissertation is my work and contains which is the result of work carried out in collaboration with others as specified in the footnotes, text, and references.

Kaiserslautern, February 24, 2025

Pramod Narasimha Murthy

This work is dedicated to my parents, who have always been my source of inspiration and support....

Contents

Introduction 1

1 Introduction 1
1.1 Motivation . . . 2
1.2 Challenges . . . 4
1.3 Thesis Outline . . . 6

I Preliminaries 11

2 Human Pose Estimation Overview 15
2.1 Overview . . . 15
2.2 Literature Review . . . 17
2.3 Human Pose Representation . . . 18
2.3.1 2D Pose Representations: . . . 18
2.3.2 Kinematic Models: . . . 20
2.3.3 Human Body Models . . . 22
2.3.4 3D Skeletal Joint Rotation Representation . . . 25
2.4 Problem Formulation . . . 26
2.5 Uncertainty Analysis . . . 27
2.5.1 Monte Carlo Dropouts . . . 28
2.5.2 Deep Ensembles . . . 29
2.6 Conclusion . . . 30

3 Learning 3D Human Joint Constraints from Mocap Datasets 31
3.1 Overview . . . 32
3.2 Related Works . . . 32
3.3 Methodology . . . 34
3.3.1 Swing-Twist Decomposition . . . 34
3.3.2 Look-Up Maps for Synthetic Dataset . . . 36
3.3.3 Training Discriminators . . . 37
3.4 Results and Discussion . . . 38
3.5 Conclusion . . . 41

4 Deep 3D Human Joint Rotation Regression 43

4.1 Introduction . . . 43
4.2 Overview . . . 44
4.3 Related Works . . . 45
4.4 Methodology . . . 46
4.4.1 HeatmapReg . . . 47
4.4.2 VectorReg . . . 49
4.5 Implementation . . . 50
4.6 Evaluation Results . . . 50
4.6.1 Quantitative Results . . . 51
4.6.2 Qualitative Results . . . 53
4.7 Additional Experiments . . . 55
4.7.1 Effect of Using 3D Joint Angle Constraints . . . 55
4.7.2 Baseline 3D Pose estimation . . . 57
4.8 Conclusion . . . 59

II Exocentric view: Self Occlusion 61

5 Diving Sports Pose Regression 67
5.1 Introduction . . . 67
5.2 Related Works . . . 68
5.3 Diving Sports Datasets . . . 69
5.4 DiveNet Pose Regression Network . . . 71
5.5 Evaluation Results . . . 72
5.6 Efficient Architectures for Diving Sports Pose Estimation . . . 76
5.6.1 BlazePose . . . 77
5.6.2 MobileHumanPose . . . 78
5.6.3 Model Evaluation . . . 79
5.6.4 Ablation Studies . . . 80
5.7 Conclusion . . . 82

6 Diving Action Localization 83
6.1 Overview . . . 84
6.2 Related Works . . . 84
6.3 Methodology . . . 87
6.4 Implementation Details . . . 89
6.5 Evaluation Results . . . 90
6.5.1 Classification Accuracy . . . 91
6.5.2 Localization Latency . . . 91
6.6 Ablation Studies . . . 92
6.6.1 Importance of Balanced Sampling . . . 92
6.6.2 Backbone Feature Extraction Network . . . 92

6.6.3 Number of Input Frames . . . 94
6.6.4 Weighted Cross Entropy Loss . . . 94
6.6.5 Temporal Context . . . 94
6.6.6 TCN vs GRU . . . 95
6.7 Conclusion . . . 95

7 Diving Motion Physical Parameters Extraction 97
7.1 Overview . . . 97
7.2 Center of Mass Trajectory . . . 98
7.3 Peak Dive Height Estimation . . . 99
7.3.1 Mathematical Derivation 1 . . . 100
7.3.2 Mathematical Derivation 2 . . . 102
7.4 Proposed Diving Plane Homography Estimation . . . 103
7.5 Body Angular Rotation . . . 108
7.6 Evaluation of Physical Parameters . . . 109
7.7 Demonstrator Tool . . . 113
7.8 Conclusion . . . 114

III Exocentric view: Partial Visibility 117

8 In-Cabin Car Driver Pose and Activity Monitoring Dataset 121
8.1 Introduction . . . 121
8.2 Related Works . . . 124
8.3 Driving Datasets . . . 124
8.3.1 DREICAP-3D . . . 126
8.4 Automated Mocap Annotation . . . 129
8.5 Implementation . . . 132
8.6 Conclusion . . . 134

9 Driver Pose Estimation using Compressed Volumetric Heatmap 135
9.1 Introduction . . . 135
9.2 Methodology . . . 138
9.2.1 Volumetric Heatmap Autoencoder . . . 138
9.2.2 Code Predictor Network . . . 140
9.2.3 Pose Refiner Network . . . 142
9.3 Implementation . . . 143
9.3.1 Post-processing . . . 144
9.4 Evaluation Results . . . 145
9.5 Ablation Study . . . 149
9.5.1 Influence of Varied Voxelization Dimensions . . . 150
9.5.2 Improvement through Pose Refiner Network . . . 150
9.6 Conclusion . . . 150

10 Driver Pose Estimation using Point Clouds **153**
10.1 Overview . . . 153
10.2 Motivation . . . 153
10.3 Proposed Methodology . . . 154
10.3.1 Sampling Techniques . . . 155
10.3.2 Network Structure . . . 156
10.4 Experimental Setup . . . 158
10.5 Evaluation Results . . . 159
10.5.1 ITOP Dataset . . . 159
10.5.2 DREICAP-3D Dataset . . . 160
10.6 Conclusion . . . 164

IV Egocentric view: Self Occlusion and Partial Visibility 167

11 Egocentric Human Pose Estimation with Head Mounted Camera **171**
11.1 Overview . . . 171
11.2 Related Works . . . 173
11.3 Methodology . . . 173
11.3.1 xR-Ego Network . . . 174
11.3.2 EgoPoseNet Network . . . 175
11.4 Implementation Details . . . 179
11.5 Evaluation Results . . . 179
11.6 Ablation Studies . . . 180
11.7 Uncertainty Estimation . . . 182
11.8 Conclusion . . . 183

V Conclusion 189

12 Conclusion **191**
12.1 Summary . . . 191
12.2 Future Work . . . 194

Appendices 197

A Deep Learning Modules **199**
A.1 Stacked Hourglass . . . 199
A.2 MobileHumanPose Model . . . 200
A.3 SPIN . . . 202
A.4 VIBE . . . 203
A.5 SOMA Model . . . 204
A.6 PointNet++ . . . 207

A.7 EgoPoseNet . . . 209
A.8 WASP . . . 209

B Datasets **211**

C Joint Rotation Representation **213**
C.1 Rotation Matrix . . . 213
C.2 Euler angles . . . 213
C.3 Unit Quaternions . . . 214
C.4 Axis angle . . . 214
C.5 Exponential Map . . . 215
C.6 6D Rotation Representation . . . 215

D Evaluation Metrics **219**
D.1 Mean Pixel Error (MPE) . . . 219
D.2 Percentage of Correct Key-points (PCK) . . . 219
D.3 Mean Per Joint Position Error (MPJPE) . . . 220
D.4 Mean Per Joint Position Error after Procrustes Alignment (PAMPJPE) . . . 220
D.5 Angle error . . . 220
D.6 Inference Time . . . 220

Backmatter 221

References **221**

List of Figures **245**

List of Tables **253**

Curriculum Vitae **257**

List of Publications **259**

Supervised Student Works **260**

Acronyms

AAM	Active Appearance Model
ADAS	Advanced Driver Assistance Systems
ASM	Active Shape Model
ASPP	Atrous Spatial Pyramid Pooling
BVH	Bio Vision Hierarchy
CNN	Convolutional Neural Network
COM	Center of Mass
CPM	Convolutional Pose Machines
CRS	Continuous Relaxation based Sampling
DNN	Deep Neural Network
FCN	Fully Convolutional Network
FFS	Forward Facing Seat
FOV	Field of View
FPS	Frames Per Second
GCN	Graph Convolutional Network
HCI	Human-Computer Interaction
HDRI	High Dynamic Range Imaging
HMD	Head-Mounted Display
HPE	Human Pose Estimation
IAT	Institut für Angewandte Trainingswissenschaft
IDIS	Inverse Density Importance Sampling
KNN	K-Nearest Neighbors
LCN	Locally Connected Network
LFA	Local Feature Aggregation
LSE	Local Spatial Encoding
LSTM	Long Short-Term Memory
mAP	Mean Average Precision
MPE	Mean Pixel Error
MRG	Multi-Resolution Graph
MSE	Mean Squared Error

MSG	Multi-Scale Guidance
NIR	Near-Infrared
ONNX	Open Neural Network Exchange
PAF	Part Affinity Field
PCA	Principal Component Analysis
PCK	Percentage of Correct Keypoints
PCKH	Percentage of Correct Keypoints with normalized Head Length
PDAN	Pyramid Dialated Attention Network
PGS	Policy Gradient based Sampling
PVE	Per Vertex Error
RFIS	Rear Facing Infant Seat
RNN	Recurrent Neural Network
ROI	Region of Interest
SCAPE	Shape Completion and Animation of People
SHN	Stacked Hourglass Network
SMPL	Skinned Multi-Person Linear Model
SOMA	Solving Optical Marker-based mocap Automatically
SSM	Synchronized Scan and Markers
TCN	Temporal Convolutional Network
VPD	Video Pose Distillation
WASP	Waterfall Atrous Spatial Pyramid

Mathematical Notations

I_{RGB}	RGB Image
I_{IR}	IR Image
I_{Depth}	Depth Image
P_{2D}	Pose represented using 2D Keypoints
P_{3D}	3D Pose represented using 3D keypoints
P_{3DA}	3D Pose represented using 3D skeletal joint rotations
μ	Mean 3D Pose represented using 3D keypoints
σ	Standard deviation of the uncertainty estimated in 3D Pose

Introduction

1

> *"Every movement reveals us."*
>
> — **Michel de Montaigne**
> french philosopher

Understanding human motion has been a longstanding area of interest and research with its roots in biomechanics and physiology. Researchers in these fields have sought to understand how the human body moves, how muscles and joints work together, and how different movements impact overall health.

Early pioneers like Eadweard Muybridge used photography to capture motion sequences, contributing to our understanding of human locomotion [1]. His work played a significant role in the early understanding of human and animal locomotion and had a profound influence on the fields of photography, art, science, and even popular culture. The best known early motion studies, where the capture sequences of images depicted the movement of people and animals in a highly detailed and precise manner. He conducted a series of experiments with the goal of settling a bet regarding whether all four of a horse's hooves leave the ground during a gallop. This work had a profound impact on both the scientific and artistic communities. Scientists and researchers used his photographs to study locomotion, anatomy, and biomechanics.

The understanding of human motion through pose estimation has undergone a remarkable evolution over the years. The sensitivity to biological motion, documented by psychologist Gunnar Johansson in 1973 using point light displays (PLDs) [2], revealed that people recognize actions when the PLD is in motion. The 1970s and 1980s witnessed the emergence of early computer vision research in human pose estimation. Johansson's work inspired modern PLD technology, utilizing multiple cameras and computers for 3D representations.

Primarily, the advent of motion capture technology for animation used in the film and entertainment industry. The techniques evolved by attaching markers to an actor's body and using multiple cameras to record movements for realistic animations [3]. Simultaneously, biomechanics researchers began employing film and video analysis to study human gait and movements, providing insights into how humans walk and perform various activities [4]. Researchers began to develop algorithms that utilized simple geometric models and feature points to estimate body positions [5]. In the world of computer graphics and animation, the concept of kinematic chains and articulated models gained prominence [6]. These

developments influenced the creation of pose estimation methods that incorporated the kinematic structure of the human body to improve accuracy [7, 8, 9].

The discovery of mirror neurons in 1992, initially in the premotor cortex, renewed interest in biological motion [10]. This insight has driven extensive research in various fields, exploring the connection between visual information, motor actions, and their implications on action perception in neuroscience, language, motion capture, and artificial intelligence. During the late 20^{th} century and early 21^{st} century, contour-based techniques, such as the Active Shape Model (ASM) [11, 12] and Active Appearance Model (AAM) [13], took center stage. These methods were centered on detecting the outline of the human body to estimate pose and facial expressions.

The 2010s marked a revolution in human pose estimation with the introduction of deep learning, especially convolutional neural networks (CNNs) [14, 15, 16]. Researchers harnessed the power of CNN-based methods to significantly enhance the accuracy of pose estimation, enabling real-time detection of keypoints in video. The recent decade and the present era have witnessed a shift from 2D pose estimation to 3D pose estimation, involving the estimation of both the positions of keypoints in images and their three-dimensional coordinates in physical space [17, 18, 19, 20, 21, 22].

Human pose estimation is now ubiquitously applied in various domains, including fitness tracking [23, 24], healthcare [25], animation [26, 27], virtual reality and human-computer interaction [28]. Furthermore, real-time and multi-person pose estimation has become feasible due to advances in hardware and algorithmic efficiency, paving the way for applications in augmented reality, gaming, and security [29, 30, 31, 32, 33]. These applications underscore the practical significance and increasing relevance of human pose estimation in contemporary society.

The historical evolution of human pose estimation showcases a remarkable journey from the pioneering work of Eadweard Muybridge to the current era of deep learning based human pose applications. This evolution trajectory reflects the gradual transition from early photographic experiments to sophisticated computer vision techniques, setting the stage for further advanced research and innovation in this dynamic field.

1.1 Motivation

Human pose estimation (HPE) plays a crucial role in applications such as computer vision many downstream task such as person tracking and gesture recognition. It is vital in Human-Computer Interaction (HCI) for enabling natural and intuitive interactions between humans and computers [34]. This can include gesture-based controls, virtual reality, and augmented reality applications. In the entertainment industry, pose estimation is used for creating realistic character animations in video games and movies. It also has applications in motion capture for character design.

In the field of sports, pose estimation is used for tracking the movements of athletes. This data can be used for performance analysis, injury prevention, and improving training

regimes [35]. Pose estimation can also be used in healthcare for tracking the movements of patients. This is useful in rehabilitation, monitoring the elderly, and other medical applications [36]. Autonomous robots and vehicles benefit from pose estimation to understand and interact with the environment [37], especially in the context of self-driving cars [38] and delivery robots.

Pose estimation can be applied to understand customer behavior in stores. It helps in tracking how customers move through a space and interact with products [39]. It also plays a vital role in security and surveillance by tracking and analyzing human movements for applications like crowd monitoring, anomaly detection, and facial recognition. Virtual try-on solutions often utilize pose estimation to allow customers to see how clothing items will fit and look on them [40].

In the realm of computer vision, human pose estimation involves determining the spatial positions of key body joints or landmarks in a single image or a sequence of frames from a video. The goal of human pose estimation is to precisely identify and locate the positions of various body parts within a specific posture. By accurately determining the spatial coordinates of key joints or landmarks, it enables computers to interpret and understand the body pose and movements of humans depicted in visual data. However, human pose estimation in monocular cameras presents several challenges due to the limitations inherent in using a single 2D image for analysis. Monocular cameras capture only 2D information, making it challenging to discern the depth or distance of body parts from the camera. This ambiguity can lead to inaccuracies in estimating the actual three-dimensional pose [41].

Occlusions occur when body parts or joints are obstructed by other objects or body parts. Monocular cameras may struggle to accurately estimate poses in the presence of occlusions. Monocular cameras are sensitive to changes in viewpoint. Variations in camera angles or perspectives can affect the appearance of body parts, making it challenging to maintain accurate pose estimation across different viewing conditions [42].

Marker-based human motion capture systems excel in providing highly accurate recordings of lifelike human movements [43, 44, 45]. While these systems are extensively used in animated films to imbue characters with realistic motion, they are also employed in monitoring sports athlete performance through multi-view camera-based setups. Nevertheless, these systems face limitations and difficulties in tracking optical markers in an uncontrolled, real-world complex environments.

To address these limitations, other motion capture technologies, such as markerless motion capture (using computer vision techniques) [46], inertial motion capture (using sensors) [47, 48], or hybrid approaches [49, 50] have been developed. Markerless motion capture systems although offer a promising alternative, overcoming many of limitations of marker-based systems [41]. They also contend with inherent issues, including depth ambiguities and partial observations. These technologies aim to provide more robust solutions for motion capture in diverse real-world scenarios, but each has its own set of advantages and limitations. Despite these limitations, advancements in computer vision

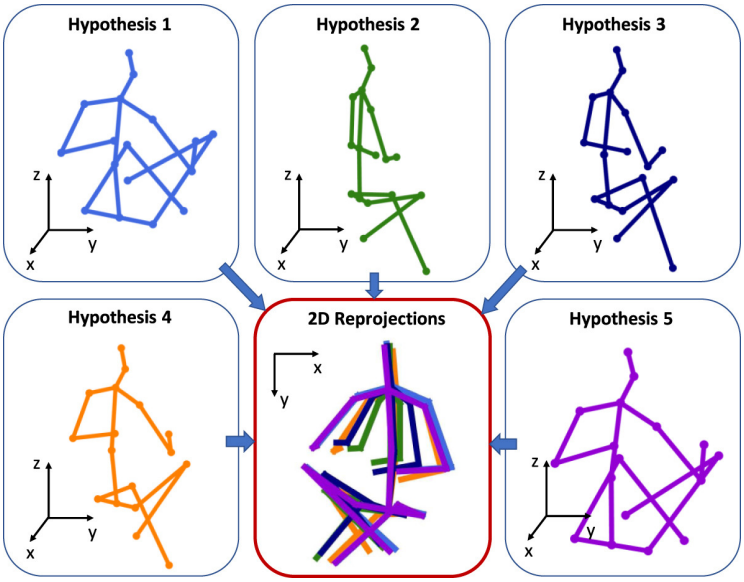


Figure 1.1.: Ill-posed problem of 3D Human Pose Estimation. The figure shows multiple 3D pose hypothesis to have same 2D pose reprojection [51]

continue to improve the robustness of pose estimation in diverse applications. Researchers address these challenges through machine learning algorithms, often incorporating deep learning techniques and leveraging additional contextual information.

In next section, we delve into the challenges for markerless monocular human pose estimation.

1.2 Challenges

The 3D human pose estimation from a single view 2D pose is ambiguous, as one 2D pose can have multiple plausible 3D poses. The multiple different 3D poses can be re-projected into the same 2D pose, resulting in an ill-posed problem. The Figure 1.1 illustrates five different 3D pose re-projections into identical 2D pose. This could arise due to various factors such as occlusion (body parts hidden due to a particular view point), ambiguous visual cues, variations in lighting, or the inherent complexity of human body pose [41].

The detailed explanation of these challenges towards 3D Human Pose estimation are as follows:

- **Self-Occlusion:** There are multiple reasons for occlusions to occur. It can be due to some object or one joint hiding the other joint. Complex poses and flexible body configurations can also lead to self-occlusion as shown in Figure 1.2 (top left).
- **Complex environment:** The visual scene can contain occlusion or truncation in the camera view or occlusion due to persons nearby, as represented in the last column of Figure 1.2.

- **Perspective Distortion**: Perspective distortion occurs when the position and angle of the camera impact the perceived size and shape of objects in an image. Tilting the camera upward or downward can lead to objects in the scene appearing either stretched or compressed in the final image (top right image of Figure 1.2). Consequently, objects farther from the camera may seem smaller than their actual size, while those closer may appear larger. In the context of ego-centric pose, it may result in the upper body appearing larger in comparison to the lower body. Additionally, fish-eye camera images must address additional spherical distortion.

- **In-variance factors**: Computer vision algorithms encounter problems because of inconsistencies in elements like lighting, views, and backgrounds. In the instance of HPE, the images of people may include poses with self-similar body parts and people wearing various types of apparel as shown in the Figure 1.2.

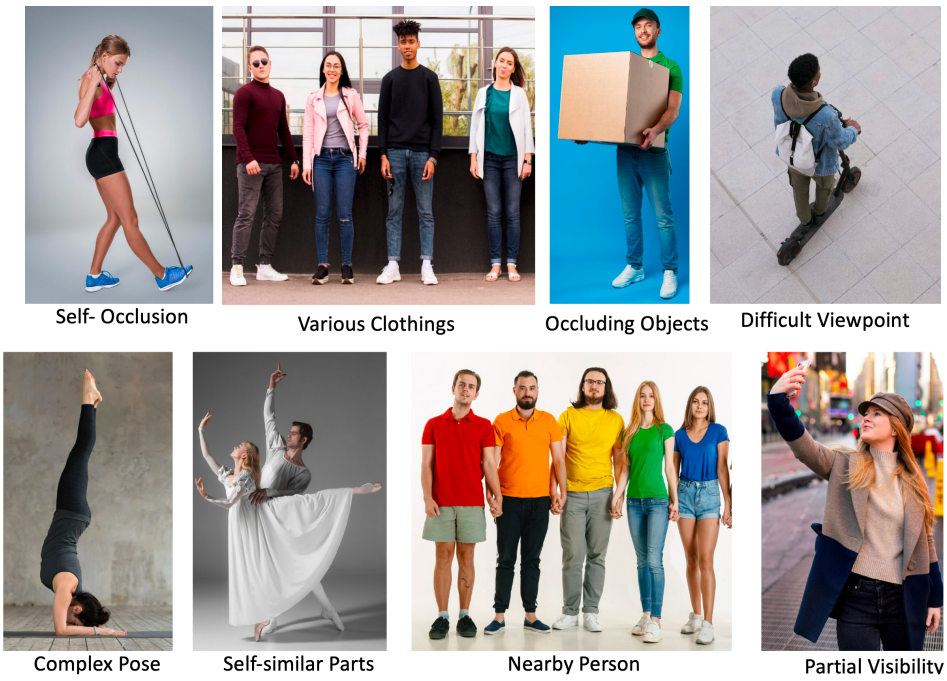
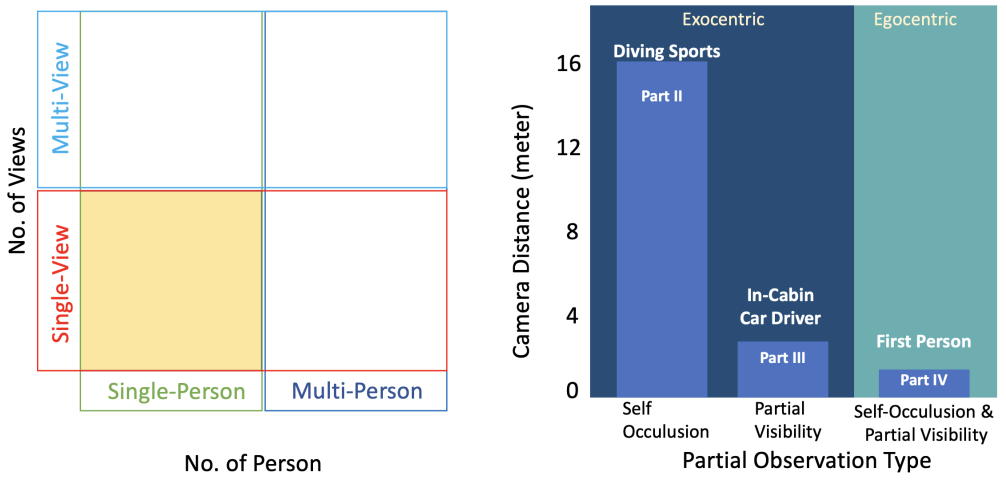


Figure 1.2.: Various challenges for human pose estimation in monocular images and videos. (Images: [52])

The accuracy of pose estimation is also significantly influenced by the distance from the camera, with several key factors to consider [53]. Firstly, as the distance increases, the resolution of the image decreases, posing challenges for algorithms to detect and localize key body points accurately . Additionally, variations in distance alter the scale of the human body within the frame, impacting relative positions and proportions of body parts, potentially leading to inaccuracies. Moreover, changing distances introduce different perspectives,



(a) Thesis scope **(b)** Thesis Overview

Figure 1.3.: The scope of thesis is to focus on single-person pose using monocular view images (left). The thesis parts (right) showing the different application scenarios and the partial observation type in each scenario. The horizontal bars show the possible range of camera distance in meters for each part.

resulting in different appearance of body parts and necessitating adjustments in pose estimation algorithms. Furthermore, increased distance can raise noise levels in the image and heighten occlusion probabilities [54]. Lastly, variations in camera parameters, such as focal length and lens distortion, also influence 3D pose accuracy, necessitating calibration and adaptation strategies to mitigate these effects and improve overall performance.

To enhance pose estimation accuracy despite varying distances from the camera, several strategies can be implemented. Utilizing multiple cameras from diverse viewpoints offers additional information, aiding in overcoming challenges linked to distance and perspective. Calibration of cameras, considering intrinsic and extrinsic parameters, can enhance accuracy across different distances [55]. Additionally, preprocessing techniques such as image denoising, normalization, and perspective correction can elevate image quality and alleviate distance-related challenges [56, 57]. Furthermore, adapting pose estimation models through training or fine-tuning on data captured at different distances with augmentation techniques can bolster their robustness and generalization capabilities [58, 59]. In summary, while distance can impact pose estimation accuracy, utilizing different deep learning strategies can effectively mitigate the challenges and enhance overall performance.

1.3 Thesis Outline

The current thesis work concentrates on addressing the problem of partial observations, which can result from own body (self-occlusion) or limited visibility of human body (partial visibility). The primary objectives of this thesis is to recover human body pose parameters under these two conditions of partial observation. The thesis scope encompasses the exploration and development of innovative solutions for single-person human pose estima-

tion under the constraints of monocular camera setups Figure 1.3a. The research spans diverse applications, including in-cabin car driving simulation, diving sports analytics, and egocentric perspectives, aiming to provide generic insights applicable to various scenarios.

In this dissertation, we aim to investigate the challenges to enable robust human pose estimation under varying camera-view points and partial observation of human body. Thus, the primary focus is to model pose estimation methods under three different application context as shown in Figure 1.3b.

The camera viewpoint variability is classified broadly into two cases: exo-centric view and ego-centric view. An exocentric viewpoint refers to a camera or observer perspective located outside of the scene being observed. This viewpoint is typically stationary and provides an external, objective view of the scene. It captures the spatial relationships and interactions among objects or subjects within the frame. Exocentric viewpoints are commonly used in traditional cinematography, surveillance systems, and computer vision tasks such as scene understanding, object tracking, and environment modeling.

Conversely, an egocentric viewpoint refers to a camera or observer perspective aligned with the viewer's own point of view. This viewpoint effectively simulates the first-person experience, capturing the scene as it would be perceived by an individual immersed within it. Egocentric viewpoints often involve dynamic camera movements corresponding to the viewer's actions and gaze direction, making them particularly suitable for immersive experiences in virtual reality, augmented reality, first-person video games, and wearable camera applications [60]. They facilitate a sense of presence by enabling users to explore and interact with virtual or real-world environments.

The thesis is structured into five distinct parts, each focusing on specific aspects of the problem as described in the following section. Each part of thesis are explained along with its research questions guiding the exploration:

Part I: Preliminaries

The Chapter 2 to 4 covers the preliminaries part by laying the groundwork for the subsequent parts. The emphasis is on achieving realistic estimates and synthesizing human motion while adhering to anatomical joint angle constraints. To this end, a data-driven strategy is devised, utilizing 3D motion capture datasets to understand human joint limits. A key aspect involves training neural network discriminators with synthetic datasets to distinguish between valid and invalid joint rotations. The discriminators, once trained, serve as valuable priors for subsequent human pose estimation and motion synthesis. Furthermore, a deep learning module is presented to learn 3D joint angle regression using monocular RGB images. This approach goes beyond traditional 3D joint position estimation, offering a more generic representation of learned 3D poses. Notably, it enables the prediction of twists around body segments, contributing to enhanced 3D human body pose and shape estimation. Further, we investigate the better 3D joint rotation representation to train deep learning models.

The following are overarching research questions for Part I of the thesis:

1. How can constraints for 3D joint rotation be learned from data in an intuitive way, so as to identify the validity of 3D rotation by checking its valid range of motion ?
2. Which 3D joint rotation representation provide better learning representation for deep learning networks ?

Part II: Exocentric View: Self-occlusion

Chapters 5 to 7 narrow the focus to self-occlusion challenges within the exocentric perspective. We concentrate on the extraction of pose parameters in the context of Diving Sports Training Videos. The chapters cover an introduction to challenges in diving sports analytics, diving action localization in continuous video, and a comprehensive framework for diving motion parameter extraction.

The research questions guiding Part II are as follows:

1. How different deep learning methodologies effectively confront self-occlusion challenge in estimating diving poses ?
2. How to accurately demarcate diving motion using deep learning models in a continuous dive training video stream ?
3. What broad framework can be devised for the extraction of diving sport key motion parameters to benchmark performances ?

Part III: Exocentric View: Partial Visibility

Chapters 8 to 10 delve into the challenges of handling partial visibility in exocentric perspectives, specifically the driver pose estimation in a in-cabin car driving simulation environment. These chapters introduce a novel 3D driver pose dataset, propose an end-to-end pipeline to estimate 3D driver pose estimation using different input modalities.

The research questions guiding Part III are as follows:

1. How to accurately capture 3D pose data in a partially visible in-cabin car driving environment ?
2. How deep learning strategies can improve the accuracy of 3D driver pose estimation by using diverse input modalities, such as infrared (IR) or depth images ?

Part IV: Egocentric View: Self-occlusion & Partial Visibility

Chapter 11 presents a framework for 3D egocentric pose estimation from a monocular perspective using a head-mounted fisheye camera. The study uses deep learning model to predict 2D pose, 3D keypoints, and 3D joint rotations, investigating the significance of

each branch. Furthermore, it also analyzes model uncertainty while handling of occlusions in fishyeye RGB images.

The research questions guiding Part IV are as follows:

1. How deep learning methodologies perform under the combined challenge of self-occlusion and partial view of the body in a dynamic egocentric camera views ?

2. How certain the deep learning models are in predicting pose parameters for frames consisting of partial observation of body ?

Part V: Conclusion

In the concluding part, Chapter 12 concludes the dissertation by synthesizing findings on human pose parameter estimation under partial observations. The chapter discusses critical issues in modeling self-occlusion and partial visibility using deep learning, offering insights for both exocentric and egocentric camera views. The study acknowledges observed limitations and suggests paths for future research.

Part I

Preliminaries

Overview

Overview

The Preliminaries part constitutes of the foundational segment of this thesis, comprising three crucial chapters.

- Within the introductory **Chapter 2**, an overview of essential background information is presented, providing a comprehensive introduction to the primary literature and fundamental concepts to model human pose.
- Subsequently, **Chapter 3** introduces a data-driven approach aimed at acquiring constraints for human joint rotations. This involves the training of discriminators for each joint and an investigation into various representations of joint rotations.
- Lastly, **Chapter 4** conducts a thorough examination of 3D joint rotation representations, along with deep learning methodologies pertinent to a generic human pose learning framework.

Collectively, these chapters within the Preliminaries section lay the groundwork for the comprehensive exploration and analysis that ensues in the subsequent sections of the thesis.

Related publications

[1] Pramod Murthy, Hammad T Butt, Sandesh Hiremath, Alireza Khoshhal, and Didier Stricker. “Learning 3D joint constraints from vision-based motion capture datasets”. In: *IPSJ Transactions on Computer Vision and Applications* 11.1 (2019), pp. 1–9.

[2] Siddharth Ingle. “3D Human Pose Estimation in the Wild”. Master’s thesis. RPTU Kaiserslautern-Landau, Sep 2019.

[3] Ankit Kumar. “3D Human Pose Estimation with Uncertainties”. Master’s thesis. Informatik, RPTU Kaiserslautern-Landau, Dec. 2020.

Human Pose Estimation Overview

2

> *"The universe is under no obligation to make sense to you"*
>
> — **Neil deGrasse Tyson**
> Astrophysicist

In the current chapter, we offer a concise introduction to prevalent methodologies utilized in human pose estimation. Subsequently, we delve into a comprehensive review of pertinent literature and explore representations of pose in both 2D and 3D dimensions. Additionally, the chapter presents a succinct introduction to generic pose estimation using deep learning models, alongside an exploration of uncertainty modeling within these deep learning models.

2.1 Overview

A significant progress has been achieved in human pose estimation (HPE) tasks through the application of deep learning techniques. Notably, 2D HPE from images and videos with 2D pose annotations has yielded impressive results, particularly in single-person scenarios [61]. The 2D HPE are broadly classified into two common paradigms: the top-down approach and the bottom-up approach [32]. The top-down approach follows a sequence of steps, beginning with person detection to identify human candidates in the image, using a human detector (see Figure 2.1). Subsequently, within the bounding box of each detected person, it estimates the pose by recognizing key body parts. Although top-down methods offer the advantage of leveraging global context and structural information, they can be slower and more computationally intensive, especially in crowded scenes. On the other hand, the bottom-up approach consists of keypoint detection, initially identifying all key points of the targets in the image, followed by an association stage that associates the detected key points with corresponding targets to form complete poses during optimization (refer Figure 2.2). Bottom-up methods are more robust to occlusion and complex poses, and they are generally more efficient as they involve a single processing stage. However, they may struggle with handling complex poses and can lead to false positives.

The recent shifted towards addressing the complexities of multi-person HPE in heavily occluded environments presents additional hurdles due to the difficulty in obtaining accurate 3D pose annotations, especially in uncontrolled settings. While motion capture systems

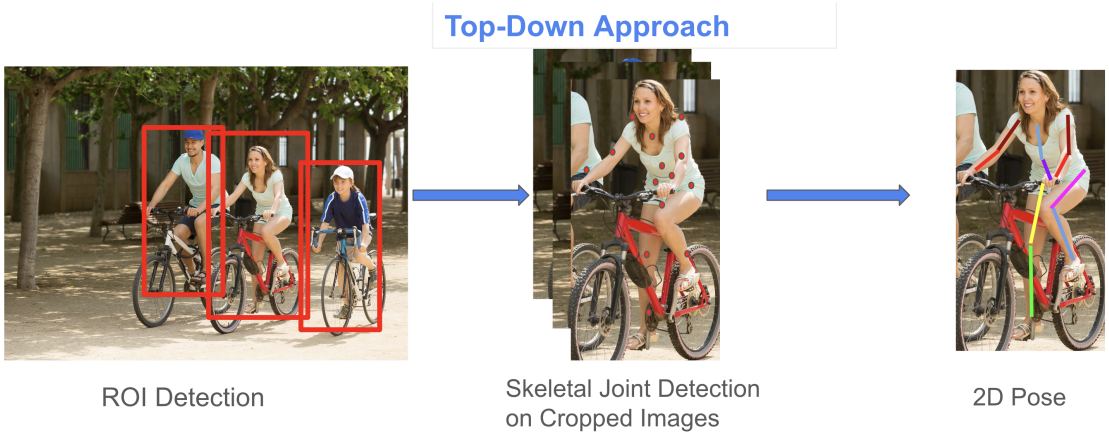


Figure 2.1.: The figure illustrates the top-down approach for pose estimation. The Region of Interest(ROI) is detected using a person detector. The cropped person images is processed to detect joint keypoints. The keypoints are combined to output the skeletal 2D pose. (Image source: [52])



Figure 2.2.: The figure illustrates bottom-up approach to pose estimation. The input image is processed to detect all the joint keypoints in an image. The keypoints are later processed to form a person's skeleton. (Image source: [52])

offer precise 3D pose data in controlled environments, they fall short in real-world scenarios [41].

Additionally, in multi-view settings, the association of viewpoints poses a significant issue. Although some approaches utilize specialized sensors like depth sensors or inertial measurement units (IMUs), they often come with cost and hardware requirements [48]. In the case of 3D HPE, it can directly yield 3D coordinates for key points from the original input data or initially predict 2D key point coordinates before undertaking a conversion into the 3D domain. Furthermore, the output of 3D HPE is not limited to merely the 3D coordinates of key points but can also encompass a comprehensive 3D model of the human body.

Given the dynamic landscape of HPE research, the next section provides insights into current state of deep learning-based 2D and 3D HPE research.

2.2 Literature Review

For single-person 3D human pose estimation tasks, Mehta et al. proposed VNet [62], which takes a single 30 Hz RGB video as input, firstly gets out the bounding box of the current frame by the key points of the previous frame, then feeds the cropped RGB picture into a fully convolutional CNN network to regress the 2D and 3D positions of the joints, and finally uses a kinematic skeleton fitting method to reconstruct the 3D motion. This network can estimate the 3D pose of the human body in real time and the accuracy of the prediction results is close to that of the real-time RGB-D method. The human pose can also be viewed as a graph consisting of joints and bones, so it is also possible to estimate a 3D pose from a 2D pose using Graph Convolutional Network(GCN) [63, 64]. However, GCN approaches do not fully exploit the structural relationships and dependencies between human body joints, which are crucial for accurate pose estimation [65]. To address some drawbacks of GCN, Ci et al proposed Locally Connected Network(LCN) [66], which can represent inter-joint relationships more directly and flexibly and with higher generalization ability. In addition to convolutional structures, transformer architectures for natural language processing can also be applied to human pose estimation. Inspired by visual transformers, Zheng et al proposed PoseFormer [67], a spatial-temporal transformer architecture. It contains a total of three modules, the spatial transformer module extracts features by considering the relationship between joints, the temporal transformer module focuses on the global dependence of the sequence, and the regression module finally regresses the 3D coordinates of the central frame.

For multi-person 3D human pose estimation tasks, top-down or bottom-up approaches are generally used. Reddy et al proposed TesseTrack [68], an end-to-end framework that uses a top-down approach to estimate multi-person 3D poses from single or multiple camera views. TesseTrack [68] consists of three parts: person detection module, person tracking module and pose estimation module. First detect the bounding box of a person in 3D voxel space, then use 4D CNN to generate a spatio-temporal cube called tesseract, and finally use deconvolution layers to get 3D pose. Mehta et. al. [69] proposed XNect that uses a bottom-up approach to capture multi-person motion with a single RGB camera in real-time. XNet [69] consists of three steps. In the first step, 2D and 3D pose features of all visible joints are estimated using the SelecSLS network architecture. The second step reconstructs a complete 3D pose for each individual through a fully connected network. The third step uses the spatio-temporal skeletal model to optimize the 3D pose and return the full skeletal pose for each person's joint angle. Cheng et al proposed a network that integrates top-down and bottom-up approaches [70]. The network can solve both the problems of wrong detection of people and wrong grouping of joints in the top-down approach and the problem of sensitivity to scale changes of multiple people in the bottom-up approach. The top-down module is used for fine-grained 3D pose estimation, the bottom-up module is used to generate a global camera-centric 3D pose, and the pose integration module matches and fuses the two poses to obtain the final 3D pose.

The 3D human pose estimation problem has been addressed in various contexts, including the analysis of single-view RGB images [71, 72] and the utilization of multi-view camera setups [73]. Methodologies employed in this domain can further be categorized into single-person pose estimation or multi-person pose estimation. Notably, employing a multi-view camera setting has demonstrated superior performance compared to a single-view setting, primarily due to the enhanced depth estimation capabilities inherent in multi-view configurations. Research pertaining to multi-view scenarios often involves amalgamating 2D positions from multiple perspectives, utilizing them as input for predicting 3D human pose estimation [74, 75]. The works of Iskakov et. al [76] leverage volumetric triangulation methods by combining multiple 2D views and employing a 3D convolutional neural network to regress 3D human pose.

Furthermore, researchers continually strive to address several challenges. Firstly, the variability in human poses, encompassing different orientations and articulations, poses a significant hurdle for accurate detection and estimation across various images [42]. Background clutter can interfere with accurate detection, while real-time performance demands efficient algorithms to adapt robustly for applications like robotics. Ensuring generalization across datasets and environments, coping with limited labeled data, and enhancing robustness to noise are ongoing challenges. Researchers tackle these obstacles through algorithmic innovations, dataset curation, augmentation techniques, and advancements in model architectures and training methodologies. However, occlusions exacerbate the learning difficulty as obscured body parts hinder complete pose inference. In the current thesis, we selectively focus on occlusion challenges to estimate pose estimation under scale variation, ambiguous pose representations in diverse scenarios.

The interested reader can refer [29, 30, 31, 32, 33] for a detailed survey of pose estimation approaches using deep learning methods. In the next section, we discuss the commonly used pose representation as an output representation for deep learning methods.

2.3 Human Pose Representation

In deep learning-based human pose estimation, keypoints, heatmaps, and skeleton graphs play crucial roles in representing and predicting the spatial configuration of a person's body in images or videos. Let's delve deeper into each of these components:

2.3.1 2D Pose Representations:

Keypoint Coordinates: Keypoints are specific anatomical locations on the human body, such as joints or body landmarks. Common keypoints include those for the head, shoulders, elbows, wrists, hips, knees, and ankles. Each keypoint is represented by a pair of cartesian coordinates (x, y), denoting its location in the image. Keypoint coordinates are fundamental for capturing the precise spatial arrangement of body parts. In deep learning models, the task often involves predicting these coordinates from input images.



Figure 2.3.: The figure shows four different 2D pose representations. The 2D pose representations can be represented in keypoint coordinates, heatmap regression, part affinity fields and skeleton graphs (from left to right).

Heatmaps: Heatmaps are 2D grids that represent the likelihood or intensity of the presence of a keypoint at each pixel location in an image. Each keypoint has an associated heatmap, where high intensity indicates a higher probability of the keypoint's presence. The model predicts heatmaps for each keypoint, and the final pose is determined by identifying peak locations in these heatmaps. Heatmaps provide a continuous and probabilistic representation of keypoint locations.

Part Affinity Fields: Part Affinity Fields (PAFs) [77] play a pivotal role in modeling the spatial relationships between joint key points. The representation involves key points or joints, affinity fields, heatmaps, and vector fields. For each key point, a corresponding heatmap is generated, representing the confidence of its presence at each pixel. Affinity fields are introduced to encode the relationships between pairs of key points. These affinity fields are represented as vector fields, offering directional information on the connections between body parts. The network architecture encompasses two branches: one for predicting individual key point heatmaps and another for predicting the Part Affinity Fields. The synergy of information from these branches allows the model to estimate both the locations of individual body parts and the interconnections between them. This comprehensive approach enhances the algorithm's ability to understand the pose and structure of the human body within an image.

Skeleton Graphs: Skeleton graphs, also known as pose graphs, represent the connectivity between keypoints by defining edges between them. The result is a graphical structure that mimics the human skeleton. Edges connect pairs of keypoints, forming a graph that represents the anatomical relationships and structural connections between body parts. Skeleton graphs are especially useful for modeling the articulated structure of the human body. They help capture dependencies between keypoints and guide the learning process in deep neural networks. Graph convolutional networks (GCNs) are often employed to process information from these graphs in a way that respects the inherent structure of human poses.

2.3.2 Kinematic Models:

Kinematic models represent the human body as a connected system of joints and limbs, often incorporating biomechanical constraints. These models are prevalent in biomechanics and motion analysis, helping to understand and simulate human motion.

An articulated human pose can be defined by a kinematic model. A kinematic model is defined by a tree structure (called as Kinematic tree) which consists of bones and joints as shown in Figure 2.4. A human body is divided into different segments, and bones represent these segments of fixed length. These bones connect each other through joints. One of the joint (usually pelvis) is considered as a root joint which defines the global position and orientation of the human body. A kinematic model has J joints and $J-1$ bones with $\{t\}_{j=1}^{J-1}$ bones lengths [78]. Each human body has a specific bone length and varies among different human bodies by a scale factor s. The scale and bone lengths are defined as apriori.

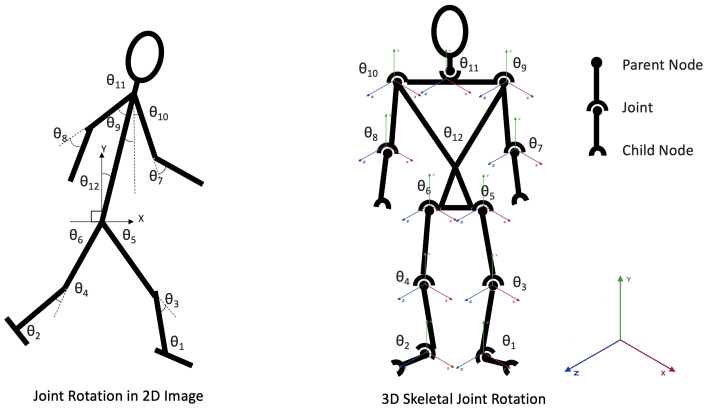


Figure 2.4.: Kinematic models in a tree structure of human body representation to measure joint rotation in 2D (left) and 3D rotation(right) with relative rotations [79].

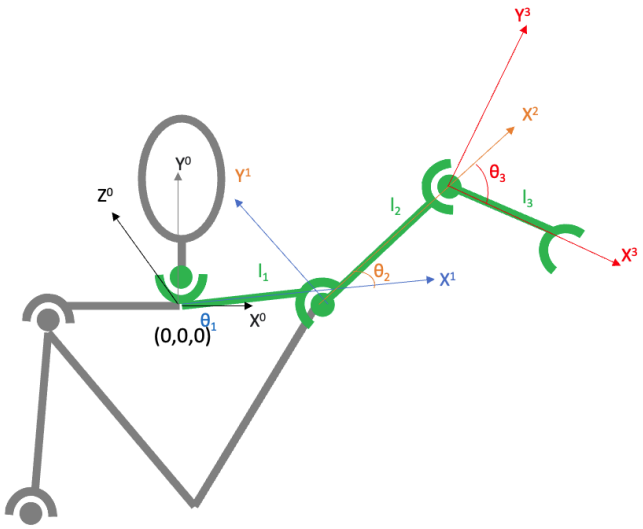


Figure 2.5.: An example of 3D kinematic chain representing left arm.

The Forward Kinematic function defines the mapping from the motion parameter space to the joint location space. Let's consider a simple case of a human arm in Figure 2.5. Let the joints have 3 degrees of freedom (DOFs), where each joint is represented by an angle θ_j (for $j = 1, 2, \ldots, n$).

The forward kinematics determines the position and orientation of the all joints represented in an skeleton based on the joint angles over a fixed skeleton kinematic structure. Mathematically, we can represent the forward kinematics of the human arm using a function FK that maps joint angles $\Theta = [\theta_1, \theta_2, \ldots, \theta_n]^\top$ to the 3D joint position represented by $\mathtt{P} = [p_1, p_2, \ldots, p_n]$. Each joint pose is represented as $\mathbf{p_j} = [\mathbf{x}, \mathbf{y}, \mathbf{z}]$ and $\theta_\mathbf{j} = [\alpha, \beta, \gamma]^\top$, where x, y, and z represent the 3D position coordinates, and α, β, and γ represent the orientation (usually expressed in terms of roll, pitch, and yaw angles).
Forward kinematics (FK) for human pose usually refers to the process of computing the pose $\mathbf{P}$, with the rest pose template $\mathbf{T} = \{t_j\}_{j=1}^{J}$ and the relative rotations $\mathbf{\Theta}$ Mathematically, the forward kinematics function F can be expressed as:

$$\mathbf{P} = \mathtt{FK}(\mathbf{\Theta}, \boldsymbol{T})$$

Formally, the relative 3D geometry between two successive bones can be expressed through a 3D rigid transformation τ, where τ belongs to the Special Orthogonal group $SO(3)$. This transformation can be decomposed into a rotation and a translation. Specifically, τ relates the local coordinate systems of two consecutive bones and can be represented as a 4×4 matrix:

$$\tau = \begin{pmatrix} \mathrm{R} & \mathbf{t} \\ 0 & 1 \end{pmatrix}$$

Here, R is a 3×3 rotation matrix, and $\mathbf{t}$ is a 3D translation vector. To illustrate, consider a joint with coordinates $\mathbf{x} = (x, y, z)^\intercal$ in coordinate system j. These coordinates $\mathbf{x}' = (x', y', z')^\intercal$ relative to coordinate system $j - 1$ can be obtained using the transformation θ_j:

$$\begin{pmatrix} \mathbf{x}' \\ 1 \end{pmatrix} = \begin{pmatrix} \mathrm{R}_j & \mathbf{t}_j \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \mathbf{x} \\ 1 \end{pmatrix} = \tau_j \begin{pmatrix} \mathbf{x} \\ 1 \end{pmatrix}$$

In a simplified scenario, where each joint is responsible for changing the position and orientation of the end joints in a straightforward manner (e.g., rotational joints along specific axes), the forward kinematics function can be decomposed into a series of transformation matrices representing the transformations from one joint to the next, until the end joints are reached in a kinematic chain. This can be represented as a series of homogeneous transformation matrices, typically denoted as τ_j, where each matrix represents the transformation from joint j to joint $j + 1$ (or from the base to joint j).

$$P_j = (\prod_{n \in N(j)} \theta_j) P_r \tag{2.1}$$

where, $N(j)$ are the set of parent joints nodes of joint j in a given the kinematic tree, P_r represents the pose of the root of the skeleton in the homogeneous coordinates.

where $\mathbf{P}_0$ represents the position and orientation of the initial pose when all joint angles are zero (often known as T-Pose or default pose). Each τ_j matrix is typically is represented by the rotation angles and translation of the bones in homogeneous coordinates.

This is a simplified explanation of the mathematical formulation of forward kinematics with pose angles Θ. The exact formulation may vary depending on the defined skeletal kinematic structure, specified joint coordinate definition and joint rotation axis order considered. Typically, the pelvis joint is selected as root of the skeleton instead of neck joint.

2.3.3 Human Body Models

This section briefly discusses about the models which are used in the research community to encompass the 3D human pose. Human body is a non-rigid object with many characteristics such as kinematic structure (explained in section 2.3.2), body shape, surface texture, body joints [80]. Human body can be represented by different body models, mainly categorized into skeleton-based, contour-based and volume-based as shown in figure 2.6.

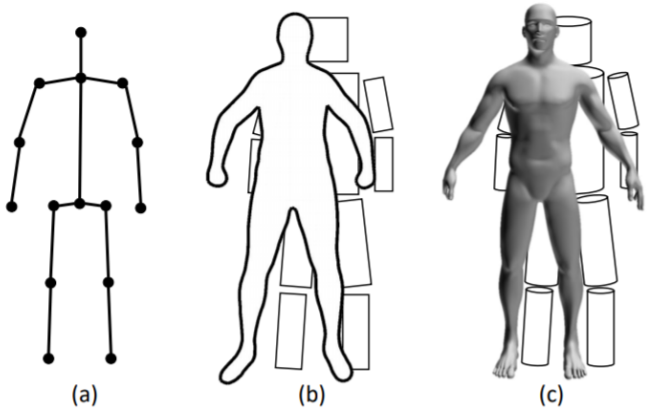


Figure 2.6.: Commonly used human body models:(a) skeleton-based (b) contour-based (c) volume-based. [80]

1. *Skeleton-based model*: These are represented as the stick-figures which commonly has joint locations and corresponding limb orientations. This representation is simple and is widely used in 2D and 3D HPE. However, the drawback of using this representation is that, it fails to capture the constraints of the body movements visually. This model is discussed in detail in section 2.3.2.

2. *Contour-based model*: The body parts of human is represented by silhouette. The most famous models are Active Shape Models (ASMs) [11]. However these models are not in use in recent research.

3. *Volume-based model*: One of the rich representation to model human body in 3D is in mesh form, which are captured with 3D scans. Generally utilized volume-based models incorporates Shape Completion and Animation of People (SCAPE) [81] and Skinned Multi-Person Linear model (SMPL) [82]. The recent body model called A Sparse Trained Articulated Human Body Regressor (STAR) [83] which is more compact than SMPL and also has better generalization capability.

Deformable Models: Deformable models, such as Active Shape Models (ASMs) or Active Appearance Models (AAMs), represent the human body as a set of deformable contours or textures. These models are applied in medical image analysis, where they can adapt to the shape of anatomical structures in different individuals.

Statistical Shape Models (SSMs)

Description: SSMs capture the statistical variability of shapes within a population. These models are often built from annotated datasets of 2D or 3D shapes, like human silhouettes or organ contours. Application: SSMs are widely used in medical imaging for tasks such as organ segmentation, where they provide a compact representation of shape variations.

Skinned Multi-Person Linear (SMPL) :

The SMPL body model has significantly contributed to advancing research and applications related to human body understanding in computer vision. SMPL has found applications in computer graphics, animation and virtual reality. It is widely used for tasks such as virtual character animation, human pose estimation, and clothing simulation. Its parametric nature and efficiency make it a valuable tool for a wide range of tasks involving the synthesis and analysis of human body shapes and poses.



Figure 2.7.: Illustration of generated body shapes using SMPL body models [82].

SMPL model encodes 3D mesh of a human body [82] as shown in Figure 2.7. This is a vertex-based model based on standard skinning method. SMPL model consists of extensive body shape with variety of human poses. SMPL body model as shown in Figure 2.8 consists of:

1. Mean template shape in the zero pose, $\bar{\mathbf{T}} \rightarrow \mathbb{R}^{3N}$, where N=6890 is number of vertices. (See Figure. 2.8 **a**)
2. A set of blend weights, $W \rightarrow \mathbb{R}^{N*K}$.
3. A blend shape function, $B_S(\overrightarrow{\beta})$ which captures variation in human shape, and different blend shape function correspond to different body shape.
4. SMPL predicts joint location of body, represented by white dots as a function of mesh vertices, denoted by $J(\overrightarrow{\beta})$. Figure **b** of Figure. 2.8
5. From 3D scans of people, pose blend shape function $B_P(\overrightarrow{\theta})$ is learned, which captures how real human body differs from skinned bodies.
6. Finally SMPL computes a linear combination of these blend shapes $M(\overrightarrow{\beta}, \overrightarrow{\theta})$ and produce realistic parametric body deformations, which are dependent on poses of human body (refer Figure 2.8)

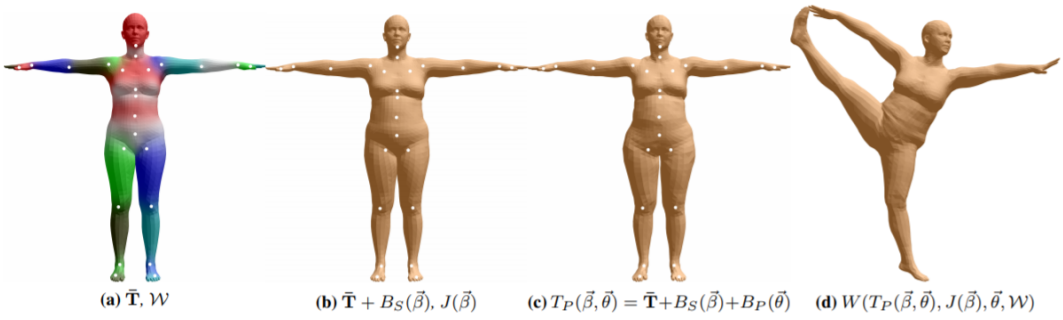


Figure 2.8.: The different phases describing the development of SMPL model [82].

Extensions of SMPL, known as SMPL-X and SMPLify-X, have been developed to address limitations and improve the model's accuracy. SMPL-X extends the model to handle diverse body shapes and sizes, while SMPLify-X refines the model's parameters using 2D images and 3D joint annotations.

SCAPE (Shape Completion and Animation of People): SCAPE is a statistical model designed for capturing human body shape variations [84]. It combines a low-dimensional pose-dependent shape space with a high-dimensional pose-independent space. It has been used for realistic shape completion and animation of human bodies. The SCAPE (Shape Completion and Animation of People) method is a data-driven approach for building a human shape model that encompasses variations in both subject shape and pose. It is based on a representation that incorporates articulated and non-rigid deformations. The

method learns a pose deformation model that derives non-rigid surface deformation as a function of the pose of the articulated skeleton, as well as a separate model of variation based on body shape. These models can be combined to produce 3D surface models with realistic muscle deformation for different people in different poses, even when they do not appear in the training set. The SCAPE method can be used for shape completion, generating a complete surface mesh given a limited set of markers specifying the target shape. It has applications in partial view completion and motion capture animation, capable of constructing a high-quality animated surface model of a moving person with realistic muscle deformation using just a single static scan and a marker motion capture sequence of the person [84].

The SCAPE and SMPL models differ primarily in their strategies for representing body shape and pose-dependent deformations. SCAPE breaks down body shape into identity-dependent and non-rigid pose-dependent components, while SMPL adopts a similar decomposition but employs a vertex-based skinning method with corrective blend shapes. Moreover, SMPL relies on blend skinning, ensuring compatibility with established rendering engines, and is trained using a vast dataset of thousands of 3D body scans. In contrast, SCAPE is a generative model tailored to image silhouettes, making it less directly compatible with graphics pipelines and game engines. Additionally, SMPL exhibits greater accuracy than SCAPE when trained on the same data [17, 85].

SMPL is a realistic 3D human body model based on skinning and blend shapes, learned from an extensive collection of 3D body scans. It employs a skinned vertex-based approach, accurately representing various body shapes in natural poses. The original 300-dimension shape space of SMPL and related models are available upon request [17, 86].

To sum up, both models address body shape and pose-dependent deformations, but SMPL utilizes a vertex-based skinning method with corrective blend shapes, ensuring compatibility with existing rendering engines. In contrast, SCAPE is a generative model tailored to image silhouettes and lacks direct compatibility with graphics pipelines and game engines. Furthermore, SMPL demonstrates superior accuracy compared to SCAPE when trained on identical data.

2.3.4 3D Skeletal Joint Rotation Representation

In contemporary 3D human pose estimation, regressing 3D joint positions stands as the prevailing method, widely employed by state-of-the-art pose estimators such as those discussed in studies by Zhou et al. [87] and Sun et al. [88]. However, an alternative approach involves the estimation of 3D joint rotations, as explored by Pavllo et al. [89] and Kanazawa et al. [90]. Notably, differences arise when considering the estimation of 3D human pose through the regression of joint positions versus the regression of joint angles or rotations.

The direct regression of 3D joint positions poses challenges, leading to potential errors in bone lengths and body symmetry. Zhou et al. [87] address this by introducing a "geometric

constraint loss" to maintain consistent limb lengths and ensure anthropomorphically realistic configurations. In contrast, regressing joint rotations with a predefined skeleton minimizes the risk of unrealistic bone lengths and asymmetrical body configurations. However, the challenge with 3D rotation parametrizations is to have a continuous representation space because it helps deep neural network frameworks to converge faster and produce optimal results.

While joint angle regression is more complex due to its less homogeneous nature [91], it offers advantages in recovering realistic poses. Joint rotations enable the estimation of factors like twists around body segments, contributing to a more accurate pose. Additionally, the regression of a human body surface in 3D space, as explored by Kanazawa et al. [90], highlights the limitations of joint positions alone in solving such problems.

Joint angles, integrated with a parameterized skeleton, are applied along a kinematic chain, preserving parent-child relationships for joints. This mitigates issues related to non-constant bone lengths and asymmetry. Opting for joint rotations allows for data-driven constraints, enhancing the realism of the final human pose. This approach also enables the imposition of learned pose priors derived from joint angle limits, restricting poses outside the articulation range.

Nevertheless, estimating joint rotations has challenges, particularly in correctly regressing joint angles for the torso region. The averaging of loss formulations may overlook the weighted impact of error-prone joints, such as the hips, compared to end-effector joints.

Defining the pose of a human body involves various perspectives, from orientation to key points like anatomical joints. The pose parameter at different times is expressed in terms of translation and rotation with respect to a global frame. The interested reader can read the brief description of all the commonly used 3D rotation representation in Appendix C.

2.4 Problem Formulation

Pose estimation is the problem of estimating the 2D and 3D positions of joints (or keypoints) of a human body from an input 2D image or a set of 2D images. Additionally, the estimation of 3D joint rotation can be used as a target, which in turn can be used to obtain 3D keypoints for a predefined skeleton using forward kinematics. The problem can be formulated mathematically as an optimization problem. The Algorithm 1 shows a general training procedure for a pose regression deep neural network.

Given N RGB input images $I_{RGB} \in \mathbb{R}^{\mathbb{N}\times 3\times \mathbb{H}\times \mathbb{W}}$ with height H and width W. Also, a pose dataset consists of 2D and 3D ground truths $P_{2D} \in \mathbb{R}^{N\times K\times 2}$ and $P_{3D} \in \mathbb{R}^{N\times K\times 3}$ respectively, where K is the number of joints. The ConvNet1 extracts learned features from the I_{RGB} RGB images. Typically, the features are run through a FullyConnectedNetwork (FCN) to regress 2D positions of human pose P_{2D}. The 2D pose P_{2D} is then lifted using another ConvNet2 to estimate 3D pose P_{3D}. The line 5 shows additional step of a forward kinematics used to obtain P_{3D} 3D positions using estimated 3D joint rotations $\widehat{P}_{3d_{\text{rot}}}$.

Algorithm 1 Generic Human Pose Network Training

Input: RGB images I_{RGB}, 2D ground truths P_{2d}, 3D ground truths P_{3d}, 2D pose prediction network $h_{2D}(\cdot)$ with ConvNet1 and FCN layers, 3D pose prediction network $h_{3D}(\cdot)$ with ConvNet2 layers, and number of iterations N

Output: Predicted 2D keypoints $\widehat{P}_{2D}$, Predicted 3D keypoints $\widehat{P}_{3D}$

for $n = 1$ to N **do**
  Features $\leftarrow$ ConvNet1$(I_{RGB}; \theta_1)$
  $\widehat{P}_{2d}^{(n)} \leftarrow$ FCN(Features; θ_2)
  $\widehat{P}_{3d}^{(n)} \leftarrow$ ConvNet2$(\widehat{P}_{2d}^{(n)}; \theta_3)$
  $\widehat{P}_{3d}^{(n)} \leftarrow$ Forward Kinematics$(\widehat{P}_{3d_{\text{rot}}}^{(n)})$ {// Optional: only for 3D joint rotation regression models}
end for
{//Loss computation during training}
$Loss_{2D} \leftarrow \frac{1}{N}\sum_{n=1}^{N}(\widehat{P}_{2d}^{(n)} - P_{2D}^{(n)})^2$
$Loss_{3D} \leftarrow \frac{1}{N}\sum_{n=1}^{N}(\widehat{P}_{3d}^{(n)} - P_{3D}^{(n)})^2$
Update network parameters: $optimizer.step(Loss_{2D}, Loss_{3D})$
return $\widehat{P}_{2D}$, $\widehat{P}_{3D}$

2.5 Uncertainty Analysis

The primary concern lies in evaluating the reliability of predictions made by deep learning models. Intelligent models require a scoring mechanism to gauge their confidence levels. The challenge intensifies in the domain of 3D egocentric pose estimation from monocular images, characterized by substantial self-occlusion and perspective transformations induced by the ego-centric view through a fish-eye camera. Furthermore, ambiguity in 3D poses becomes evident, with multiple solutions for 3D poses corresponding to the same 2D pose.

While Bayesian models like Gaussian Processes address model uncertainty, their computational demands and limitations with large-scale data make them less practical in real-time applications [92, 93]. Uncertainty in deep learning is classified into epistemic uncertainty and aleatoric uncertainty [94, 95].

Epistemic uncertainty encapsulates the model's uncertainty and confidence in predictions. It emerges from a lack of data, and resolution involves augmenting the dataset for enhanced learning. This type of uncertainty proves invaluable in scenarios prioritizing user safety, enabling the model to discern new data from training data. It is particularly useful in situations with limited training datasets. Strengthening epistemic confidence is achieved by presenting sufficient data, imposing a distribution across the weights of deep neural networks to model epistemic uncertainty.

Aleatoric uncertainty, on the other hand, pertains to uncertainties inherent in the data, capturing anomalies such as occlusions or distorted visual features, often encountered in applications integrating diverse sensors.

Two major approaches for modeling uncertainty estimation in deep neural network frameworks [96] are sampling-based and sampling-free. Bayesian deep learning methods, such as Monte Carlo Dropout [93], involve sampling model predictions from the trained

network over multiple forwards passes during inference. This approach, also known as stacking, computes variance for the accumulated model predictions. Another sampling-based method proposed by [97] relies on the ensemble of neural networks. Various networks are trained on the same dataset, and predictions are averaged to compute the variance. While effective, both methods can be computationally expensive, a topic further explored in the subsequent sections.

2.5.1 Monte Carlo Dropouts

Monte Carlo (MC) Dropouts is a method introduced by Gal and Ghahramani [93] for quantifying uncertainty in a deep neural network. Their work showed that employing a dropout layer after every fully connected layer is mathematically equivalent to an approximation to the probabilistic deep Gaussian Process. Dropout layers are traditionally used for regularization and to limit overfitting. Randomly dropping weights during training reduces the network's generalization error. But normally during inference/testing time dropout layers are deactivated. However, for uncertainty estimation purposes, dropout layers are utilized both during training and testing phases. The Algorithm 2 presents the various steps involved in estimating uncertainty for a given deep learning model $h(\cdot)$ using MC Dropout method.

Algorithm 2 Monte Carlo Dropout for Uncertainty Estimation

Input: Test Input x_{test}, prediction network $h_(x)$ with dropout layers, dropout probability p, number of forward passes to sample T

Output: Prediction mean $\widehat{y}_\mu$, Prediction variance $\widehat{y}_\sigma$

/* Enable dropout during inference */
$h(\cdot)$.eval() {Set the model to evaluation mode}
for $t = 1$ to T **do**
    $\widehat{y}_{(t)} \leftarrow h(x_{\text{test}}, p)$ {T Forward pass with dropout probability p }
end for
/* compute the epistemic uncertainty */
$\widehat{y}_\mu \leftarrow \frac{1}{T}\sum_{t=1}^{T} \hat{y}_{(t)}$ { Mean estimate $\widehat{y}_\mu$ for input x_{test}}
$\widehat{y}_\sigma \leftarrow \frac{1}{T}\sum_{n=1}^{T}(\hat{y}_{(t)} - \hat{y}_\mu)$ {Uncertainty prediction $\widehat{y}_\sigma$ for input x_{test}}

The framework proposed by [93] involves executing multiple (T) forward passes with dropout enabled for identical input data. These multiple passes are demonstrated to be equivalent to Monte Carlo sampling. The outcome of these forward passes results in T prediction samples. The mean of these T prediction samples serves as the model output, while the variance of the T prediction samples provides an estimate of the model uncertainty. The dropout rate, a hyperparameter, necessitates careful tuning. A small dropout rate diminishes the utility of Monte Carlo sampling, while a high dropout rate may lead to divergence or, at the very least, require more iterations to achieve convergence.

In essence, MC dropout is more suited for estimating epistemic uncertainty because it pertains to the model's uncertainty about the data, which can be reduced with more data or better modeling. Aleatoric uncertainty, which relates to the inherent noise or variability

in the data, may still be present but is not directly targeted by MC dropout. Nevertheless, this uncertainty can be useful in various applications, such as in safety-critical systems, where understanding the confidence or uncertainty of a model's predictions is crucial. For example, in autonomous vehicles, knowing when the model is uncertain can help trigger further exploration of the environment or alert driver for human intervention.

2.5.2 Deep Ensembles

The complete training algorithm of Deep ensembles uncertainty estimation method can be found in Algorithm 3.

Algorithm 3 Training Deep Ensembles

Input: training data x, ground truth data y, network parameters θ, number of model iteration M, n number of batch samples

Output: prediction networks $h_1(\cdot)$, $h_2(\cdot)$,..., $h_M(\cdot)$

1: Initialize θ_1, θ_2,..., θ_M randomly
2: **for** $m = 1$ to M **do**
3: Sample data points n_m randomly for each net
4: $l(\theta_m, x_{n_m}, y_{n_m}) \leftarrow \frac{\log \sigma_\theta^2(\mathbf{x})}{2} + \frac{(y-\mu_\theta(\mathbf{x}))^2}{2\sigma_\theta^2(\mathbf{x})} +$ constant
5: $h_m(.) \leftarrow$ Minimize $l(\theta_m, x_{n_m}, y_{n_m})$ w.r.t. θ_m
6: **end for**
7: **return** $h_1(\cdot)$, $h_2(\cdot)$,..., $h_M(\cdot)$

In the context of a typical pose regression task, the network is fed an image or a set of feature vectors as input and generates a single pose prediction. This output is computed based on the model parameters acquired during training. Lakshminarayanan et al. [97] suggest that by assuming the real data represents a probabilistic distribution, the model can be trained to predict not only a single output value but also the parameters of a probability distribution..

Therefore, instead of predicting a single deterministic output value y, the model learns to predict a normal distribution with parameters $(\mu(x), \sigma^2(x))$, depending on the input x. To take into account the distributional variance along with the mean value, the loss function for training needs to be updated and the commonly used mean squared error (MSE) loss (equation 2.2) cannot be used. This is accomplished by using a maximum likelihood approach by using the negative log-likelihood function of the normal distribution as a loss function (equation 2.3).

$$L_{MSE} = \frac{1}{N}\Sigma_{i=1}^{N}(\hat{y}(x_i) - y_i)^2 \tag{2.2}$$

$$L_{NLL} = \frac{\log \sigma_\theta^2(\mathbf{x})}{2} + \frac{(y - \mu_\theta(\mathbf{x}))^2}{2\sigma_\theta^2(\mathbf{x})} + constant \tag{2.3}$$

Each model predicts a continuous value for a given test input, and the final prediction is obtained by averaging the predictions of all models. This method is particularly useful

when the individual models have similar performance and are expected to make unbiased predictions as shown in Algorithm 4.

Algorithm 4 Evaluating Deep Ensembles

Input: test data x^*, prediction networks $h_1(\cdot)$,.., $h_M(\cdot)$, number of model iteration M
Output: prediction μ_* , uncertainty σ_*^2

1: **for** $m = 1$ to M **do**
2: $\mu_{\theta_m}, \sigma^2_{\theta_m} \leftarrow Ensemble(h_m(x^*))$
3: **end for**$\{//prediction\}$
4: $\mu_* \leftarrow \frac{1}{M}\sum_{m=1}^{M} \mu_{\theta_m}$ $\{//uncertainty\}$
5: $\sigma_*^2 \leftarrow \frac{1}{M}\sum_{m=1}^{M}(\sigma^2_{\theta_m} + \mu^2_{\theta_m}) - \mu_*^2$
6: **return** μ_*, σ_*^2

2.6 Conclusion

In summary, the current chapter covers a comprehensive exploration of the foundational elements crucial for modeling human pose estimation. We delved into the current research landscape surrounding monocular human pose estimation, exploring diverse methodologies and approaches. The chapter encompasses an examination of different representations of the human body as employed in existing literature. A presentation of a generic deep learning training process is included, shedding light on fundamental aspects of model development. Additionally, we discussed two widely recognized deep learning uncertainty estimation algorithms. In the subsequent chapter, our focus shifts towards a data-driven approach aimed at acquiring insights into human joint constraints, a pivotal aspect in the pursuit of realistic human pose learning.

Learning 3D Human Joint Constraints from Mocap Datasets

3

> *"The most important thing in science is not so much to obtain new facts as to discover new ways of thinking about them."*
>
> — **Sir William Bragg**
> an English physicist, chemist, mathematician

This quote emphasizing the paramount importance of scientific perspective and the continuous pursuit of innovative ways to conceptualize information. In this chapter, we look the best representations to model 3D joint rotations constraints, while using deep learning models. While traditional parameterizations such as Euler angles, Quaternions, and Axis-Angles have conventionally served as straightforward representations, their limitations in accurately capturing anatomical joint limits [98, 99, 100]. This is particularly crucial given the wealth of information provided by large motion capture datasets, highlighting the necessity to not merely obtain pose but to understand the validity of pose by capturing the range of motion for each joint.

An enormous corpus of human motion capture (mocap) datasets have emerged to train deep neural networks [101, 102, 103, 71, 104, 105]. Recently, Gül Varol [106] has even employed a synthetic human dataset for better training of deep neural networks. But synthesis of *realistic* human motion for long run is an open problem itself [107]. In the past, a *'penalty term'* for exceeding joint limits was used, but human joint constraints have been worked out from more limited datasets [108, 109, 110]. However, no recent attempt has been reported to obtain more realistic and statistically valid joint limits from publicly available mocap datasets to the best of our knowledge.

The artifacts and outliers in the mocap data (ground truth) can be pruned using joint constraints inferred directly from respective datasets. Also, the imposition of accurate joint constraints as *priors* is valuable for 3D human pose estimation, realistic human motion animation and generation of *synthetic* human motion datasets. But, a data-driven learning of joint constraints from mocap datasets is a challenging task. The presence of false outliers, voids and sparse data makes it imperative to choose a representation of joint space, which is robust to such artifacts. Moreover, the instances of class representing *invalid joint rotations* are missing in motion capture datasets and needs to be synthesized for balanced learning.

3.1 Overview

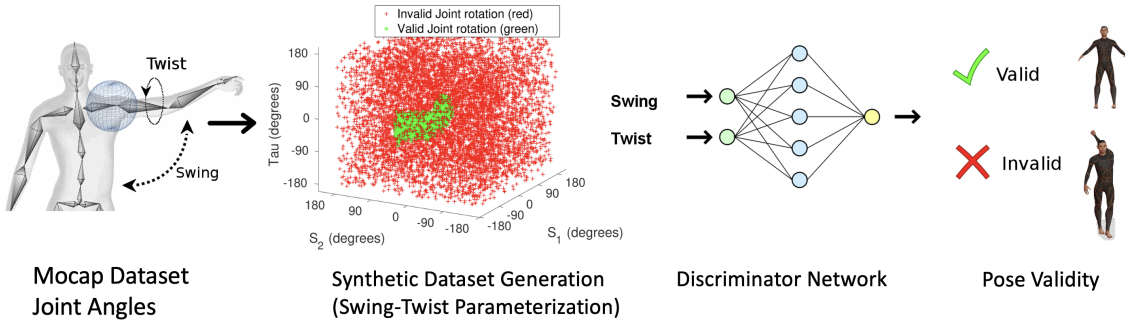


Figure 3.1.: The joint angles from motion capture datasets are transformed into (s_1, s_2, τ) swing-twist representation. Then look-up maps are prepared to train a discriminator on joint constraints to classify valid and invalid poses.

The overview of the work presented in the current chapter is depicted in the Figure 3.1. The figure shows a generic methodology of transforming motion capture (mocap) dataset joint angle values to Swing-Twist parametrization. A synthetic dataset is generated using look-up maps derived from the transformed data. A set of discriminators are trained on synthetic dataset to ascertain the validity of a given joint pose based on the dataset's characteristics. Notably, the proposed technique offers the advantage of accommodating more personalized or activity-specific constraints as opposed to generic constraints. For instance, the constraints of joints learned on athlete's data may not be directly applicable to a construction worker.

Further, our three-parameter representation is unambiguous and free of anomalies, which makes it easier to learn a joint rotation manifold. Thus, neural network discriminators trained on our representation yield a good accuracy and qualitative precision with very few parameters.

The main contribution of our work are as listed below:

- A new formulation (s_1, s_2, τ) of swing-twist decomposition of a joint rotation based on exponential map.
- Discretized look-up maps based on the above parameterization to synthesize labels for both valid and invalid instances in joint space.
- A set of neural network discriminators trained on these datasets for different joints.

3.2 Related Works

The parameterization of joint constraints is dependent on the choice of representation for joint rotations. The simplest representation of joint rotation is in terms of euler angles.

A straightforward approach to implement joint constraints is to assume minimum and maximum limits on euler angles (box-model). However, this approach is too crude to capture realistic anatomical joint limits. Also, euler angles lead to discontiguous cluster of valid poses in joint space [111] and often causes the problem of gimbal lock [100]. Due to these limitations, the box-model based joint limits on euler angles though intuitive are not an optimal choice. The swing-twist decomposition of joint rotation is devised [100, 112, 100, 113] in order to simplify the parameterization and reduce computation in order to determine validity of a given pose. However, interdependence of swing and twist (intra-joint dependence) has not been satisfactorily addressed in earlier works.

In a data-driven approach, Lorna Herda [114] proposed a novel parameterization in terms of an implicit surface on joint trajectory data in quaternion space. The authors extend their method to deal with inter-joint dependencies but it is computationally expensive and is not intuitive to interpret the joint limits. Further, the imposed limits are directly on captured data, which might lead to unrealistic limits especially in the presence of voids and sparse mocap data. Since, the vector part of quaternion is only used in [114], it clearly ignores information encoded with the scalar part leading to ambiguity in rotation angle of joint. Also, the quaternion has dual representation (+q and -q) for a given joint rotation. Thus, a preprocessing step is necessary over motion trajectory in quaternion space to eliminate this duality.

Recently, Akhter et al. [109] learned pose-conditioned joint angle limits from a more controlled mocap dataset performed by trained gymnasts and athletes. The authors encode joint limits for swing of a body segment in a discretized binary occupancy map and defined a table-look-up function to check for validity of given pose. The occupancy map is defined using local spherical coordinates of the joint on a unit sphere. The presented work also conditioned pose of a child joint based on its immediate parent in a kinematic chain, but it ignores intra-joint dependency[1].

Aforementioned limitations of previous works form the motivation of our research. Instead of quaternion or Euler angles, we use exponential map parameterization to represent joint rotations in present work. The exponential map representation avoid singularities as well as non-contiguous clusters [111] of mocap data. It also avoids the duality problem in representation as described earlier vis a vis quaternions or axis-angle. We derive our joint constraints using more comprehensive Human 3.6 Million(H36M) dataset [115] and compared these to Akhter et al. [109]. We also encoded lower and upper bounds of twist angle τ on occupancy map (s_1, s_2) of swing, as an additional parameter to be learned. This addresses the intra-joint dependency to overcome limitation of previous work.

[1] Inter-joint dependency is defined between the parent joint and the subsequent children joint angles in a kinematic chain, while Intra-joint dependency exists between swing and twist of the same joint.

3.3 Methodology

The constraints of a 3D human joint rotation for routine activities are learned by using Human3.6M (H36M) dataset [115]. Our joint constraints parameterization is based on swing-twist formulation. Initially, we decompose the joint rotation (expressed in quaternion form) into swing and twist parts. Later, we express each of the swing and twist in exponential map and arrive at (s_1, s_2, τ).

In comparison, Akhter et al. [109] learn joint rotation constraints on a motion capture dataset based on extreme range of joint movements. Moreover, they ignore twist around axis as they parameterize joint rotation from local 3D Cartesian coordinates to the spherical coordinates (ϕ, θ) representing swing of body segment. In current work, we compare our joint limits (swing part) derived from a range of normal activities performed by human subjects (in H36M) with extreme range limits provided by [109].

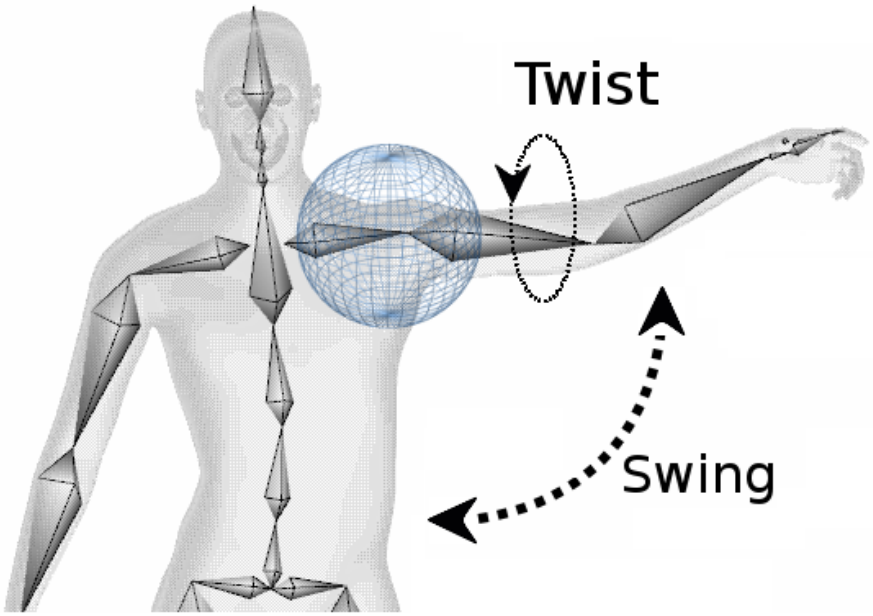


Figure 3.2.: Swing-Twist joint angle rotation decomposition

3.3.1 Swing-Twist Decomposition

The swing-twist decomposition of joint rotation is devised to be more intuitive and reduce computation. Moreover, the parameterization addresses the intra-joint dependence (dependency between swing and twist of the same joint) which has not been satisfactorily addressed in previous related research. The twist parameter is learned to account for the intra-joint dependency.

Initially, the 3D joint rotations values are represented in the form of quaternions. The quaternion rotation representation enables further decomposition into swing and twist parts [98]. The resulting swing and twist components are finally transformed into exponential map form to achieve concise three-parameter representation i.e. (s_1, s_2, τ). The Swing-Twist (s_1, s_2, τ) decomposition of joint rotation into orthogonal components i.e. swing and twist, is achieved by defining the twist with respect to an axis of intrinsic (rotated) frame as shown in Figure 3.2.

Let $\eta := \hat{\boldsymbol{\theta}}\mathbf{w}$, with $\hat{\mathbf{w}} := (w_x, w_y, w_z)$, $\|\hat{\mathbf{w}}\| = 1$ represents the angle-axis representation of a 3D rotation with $q_s := \cos(\theta/2)$, $q_x := w_x \sin(\theta/2)$, $q_y := w_y \sin(\theta/2)$, $q_z := w_z \sin(\theta/2)$. Let the corresponding quaternion $q := (q_s, q_x, q_y, q_z)$ can be decomposed as $q_{\text{twist}} \cdot q_{\text{swing}}$, where $q_{twist} := (c_y, 0, s_y, 0)$, $q_{swing} := (c_{zx}, s_x, 0, s_z)$.

Let $\eta_{twist} := (0, \tau, 0)$ and $\eta_{swing} := (s_1, 0, s_2)$ as respective exponential map counterparts.

Then by using the following quaternion properties:
(i) $\|q_{twist}\| = 1$,
(ii) $\|q_{swing}\| = 1$ and
(iii) $q = q_{twist} \cdot q_{swing}$

we get,

$$\tau = 2 \cdot \arctan\left(\frac{s_y}{c_y}\right),$$
$$s_1 = 2 \cdot \arctan\left(\frac{\sqrt{s_x^2+s_z^2}}{c_{zx}}\right) \cdot \frac{s_x}{\sqrt{s_x^2+s_z^2}},$$
$$s_2 = 2 \cdot \arctan\left(\frac{\sqrt{s_x^2+s_z^2}}{c_{zx}}\right) \cdot \frac{s_z}{\sqrt{s_x^2+s_z^2}}$$

where,

$$c_y = \left(\frac{q_s}{\sqrt{q_s^2+q_y^2}}\right), s_y = \left(\frac{q_y}{\sqrt{q_s^2+q_y^2}}\right), c_{zx} = \left(\frac{(q_s^2+q_y^2)}{\sqrt{q_s^2+q_y^2}}\right)$$
$$s_x = \left(\frac{(q_s \cdot q_x - q_y \cdot q_z)}{\sqrt{q_s^2+q_y^2}}\right), s_y = \left(\frac{(q_s \cdot q_z + q_x \cdot q_y)}{\sqrt{q_s^2+q_y^2}}\right)$$

Using the above re-parameterization, we get swing components (s_1, s_2) and the twist component τ of a joint rotation as exponential map. The exponential map form provides a superior parameterization for enforcing rotational joint limits by offering a spherical subspace encapsulating all orientations generated. Admissible orientations are confined within this sphere, and detecting out-of-bounds orientations involves checking containment within this volume. Joint limits can be effectively imposed by mapping out-of-bounds orientations to the nearest admissible orientation inside the volume using the Euclidean metric [98, 116].

The representation is convenient as it: (i) is unambiguous (no duality), (ii) avoids non-contiguous regions/jumps in joint trajectory and (iii) requires no additional constraints like unit norm. Consequently, it provides optimal joint representation for learning valid/invalid poses.

The human pose in euler angle representation provided by H36M and [109] dataset are transformed into (s_1, s_2, τ) representation, as described in previous section. The resulting joint motion space is continuous, contiguous and well-behaved for discriminative learning.

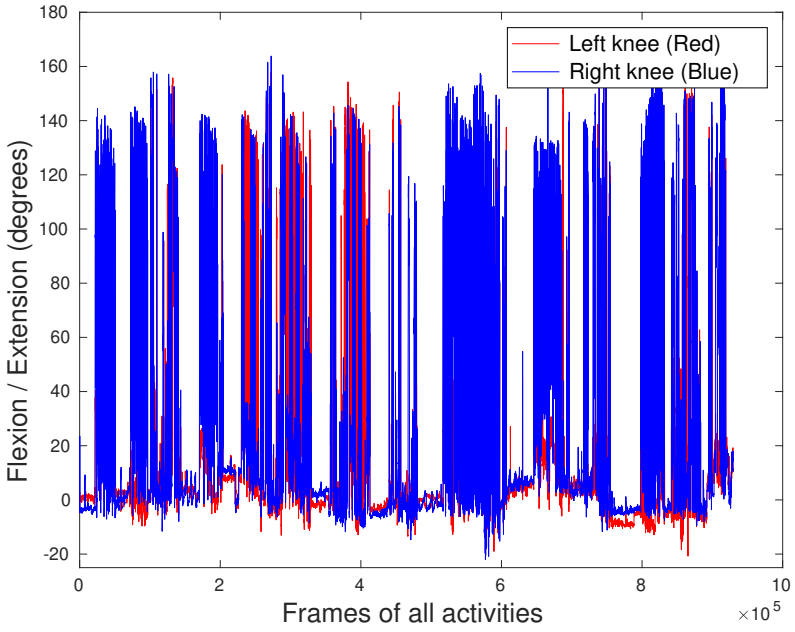


Figure 3.3.: The knee joint rotation anomalies in Akhter et al. dataset[2]

Considerable outliers are observed in the motion capture (mocap) dataset provided by Akhter et al. [109]. For instance, left and right knee flexion of all subjects performing various activities is depicted in Figure 3.3. The figure clearly shows impractical knee extensions $(upward)$ of up to -20° for certain motion sequences in dataset. Moreover, the limited set of mocap data presents a challenge to learn inter-joint dependency. Specifically, the insufficient instances of swing of the child limb exists in order to condition upon a prior swing of parent limb.

Further, we compare swing limits of right hip joint (Right Up leg) angles in the dataset provided by Akhter et al. [109] and H36M in (s_1, s_2) space (see Figure 3.4). The limits obtained using H36M though more restrictive, nevertheless lie well within the range of extreme joint movements. Furthermore, these joint limits are representative of human motion involved in routine activities. Therefore, we restrict our learning to H36M datasets than extreme range of motion.

3.3.2 Look-Up Maps for Synthetic Dataset

The joint range of motion in our (s_1, s_2, τ) space derived from H36M dataset is discretized into one degree bins on each axis. Thus, the dimensionality of the discretized joint space

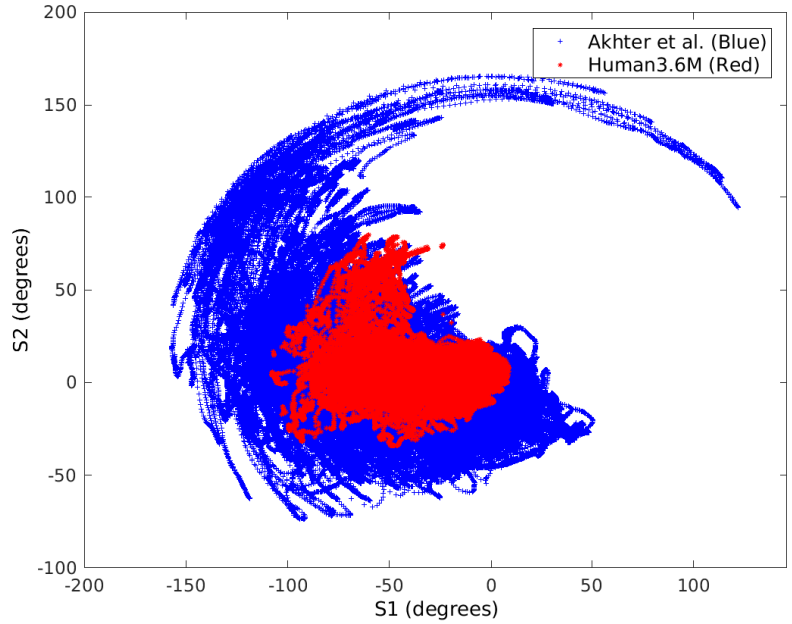


Figure 3.4.: The right hip joint's range of swing rotation

is of the order $(360 \times 360 \times 360 = 46.656\ Million)$. If a mocap sample is present at (s_1, s_2) bin, its occupancy is marked as valid or otherwise is considered to be invalid.

$$\text{Occ}_{swing}(s_1, s_2) = \begin{cases} 1 & \text{if } \exists (s_1, s_2) \in \{mocap\} \\ 0 & \text{else} \end{cases} \tag{3.3}$$

With each valid occupancy bin, we further associate minimum and maximum τ limits, based on all the instances of mocap data occurring in that bin.

$$\text{Lookup}_{twist}(s_1, s_2) = \begin{cases} \tau_{min}, \tau_{max} & Occ_{swing}(s_1, s_2) = 1 \\ undefined & \text{else} \end{cases} \tag{3.4}$$

In order to avoid noisy twist angle bounds on occupancy map we apply smoothing kernel of 3×3 prior to the generation of synthetic ground truth with valid/invalid bins in (s_1, s_2, τ) grid as follows:

$$\text{isValid}(s_1, s_2, \tau) = \begin{cases} 1 & Occ_{swing}(s_1, s_2) = 1 \\ & \text{and } \tau_{min} \leq \tau \leq \tau_{max} \\ 0 & \text{else} \end{cases} \tag{3.5}$$

3.3.3 Training Discriminators

A large portion of the complete joint rotation space is invalid region (for any given joint). Therefore, we randomly sample to obtain $30\%/75\%$ mix of valid and invalid instances

respectively. This obtains equally spaced 'labeled' training instances in entire joint space. An example of such sampled joint space for right up leg (right hip joint) is as shown in Figure 3.5.

A balanced (50%/50%) subset is selected from these sampled valid/invalid instances using random sampling. The subset is again divided into training, validation, and test samples in a ratio (70%,15%,15%). For comparison, we also tested over-sampling of valid class and under-sampling of invalid class to obtain balanced subset independently. However, the sampling proposed above leads to least variation of performance, when evaluation is performed on multiple test sets. The neural network discriminator are trained for each joint, using an equal number of valid/invalid instances in each batch. We train an ensemble of 20 discriminators for each joint to evaluate performance.

We also qualitatively evaluate the ability of neural network to discriminate valid/invalid joint rotations by means of an interactive validation tool. The tool also helps us to ascertain any anomalous examples found in motion capture datasets contradictory to the learned joint limits.

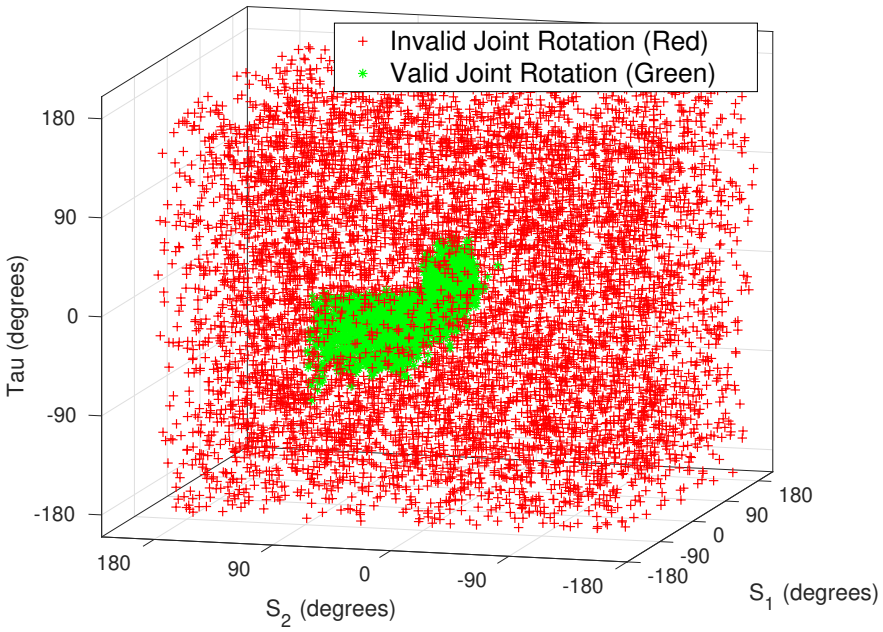


Figure 3.5.: Generated synthetic dataset for right hip joint with valid (green) and invalid (red) samples

3.4 Results and Discussion

The learning of a smooth implicit function by neural network represents the valid range of motion for a specific joint i.e valid swing and twist component of joint rotation. We achieve over 95% accuracy on all joints using only one hidden layer of 10 neurons in Fully Connected Neural Network (results shown in Table 3.2). There was no improvement

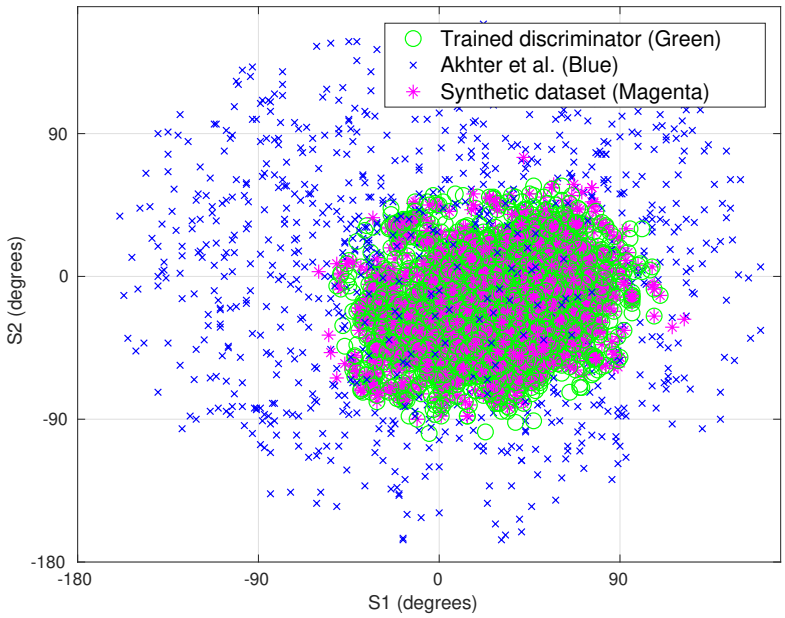


Figure 3.6.: The valid left shoulder joint rotation of (i) Trained discriminators output (green) (ii) Akhter et al. dataset (blue) [109] and (iii) Synthetic dataset (magenta)

found in accuracy with an increase in number of neurons or hidden layers. In our view, this is only feasible due to the choice of joint rotation parameterization, which is free of ambiguities or discontinuities over a joint rotation manifold (compare Figure 3.5, Figure 3.10a and Figure 3.10b). However, the accuracy on 'synthetic' ground truth is not an absolute figure of merit, as the synthetic dataset generation from occupancy maps might result in generation of data false outliers, voids or sparse data. The corresponding outliers in *synthetic* dataset which are learned as invalid by discriminator are shown in Figure 3.7. The output of discriminator also shows that learning is smooth and effectively accounts for outliers in the data. Hence, the precision of valid/invalid classification joint rotation in a given pose is further evaluated qualitatively using an interactive tool. The output of our interactive pose validation tool is as depicted in Figure 3.8. Further, we also present a dataset motion visualizer as shown in the Figure 3.9. The visualization includes the 2D Pose overlaid on the image and the corresponding 3D skeleton on the bottom-left. On the right side of the interface, Swing-Twist look-up maps (in green) for each joint, learned over the Human3.6M dataset, are displayed. This tool facilitates the examination of joint validity across various sequences in the dataset.

Furthermore, the *intra-joint* (parent-child) dependency of range of motion in a hierarchy of joints [109, 114] is also implicitly learned in our approach. Our approach emphasizes the coupling of parent-child arising from twist itself. For example, a constraint on the twist of upper limb avoids *unrealistic* pose of the lower limb by twist propagation along its kinematic chain. Thus, presented with any (s_1, s_2, τ) parameterized joint rotation, our trained neural network(s) are able to decide for the valid or invalid joints independently.

The Table 3.2 shows that discriminators perform over 95% consistently for all the joints. The Table 3.1 shows a discriminative comparison matrix on test poses of left shoulder using Akhter et al. [109] and our trained network. All poses classified as valid by our network are also found valid by [109]. However, since [109] does not take into account twist, the joint rotations classified as invalid by our network on account of twist are still valid in [109]. The Figure 3.6 further highlights that Akhter et al. isValid() function do not constrain the left shoulder swing very well and almost entire swing space is valid.

These two factors account for high instances (36.65%) of left shoulder poses classified as invalid by our network while still valid by [109].

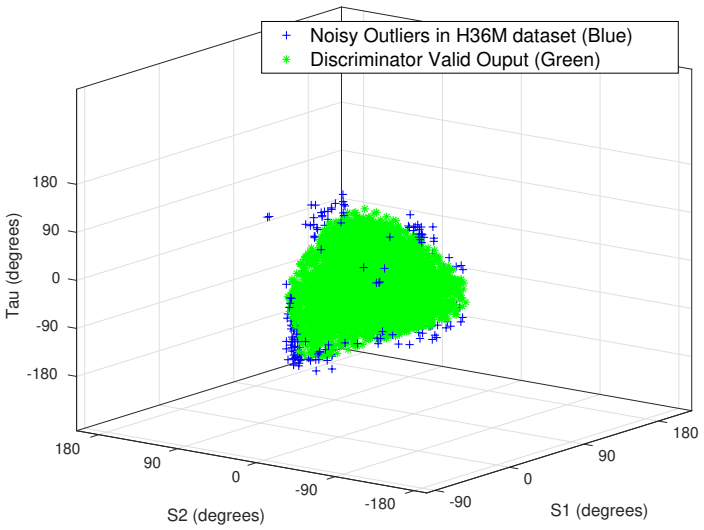


Figure 3.7.: Valid discriminator network output of left shoulder joint in (s_1, s_2, τ) space

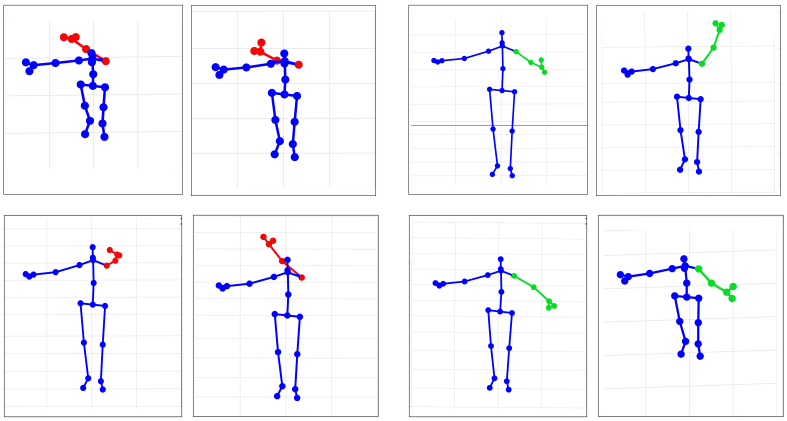


Figure 3.8.: Interactive validation tool showing invalid left arm poses in red color and valid left arm poses in green color

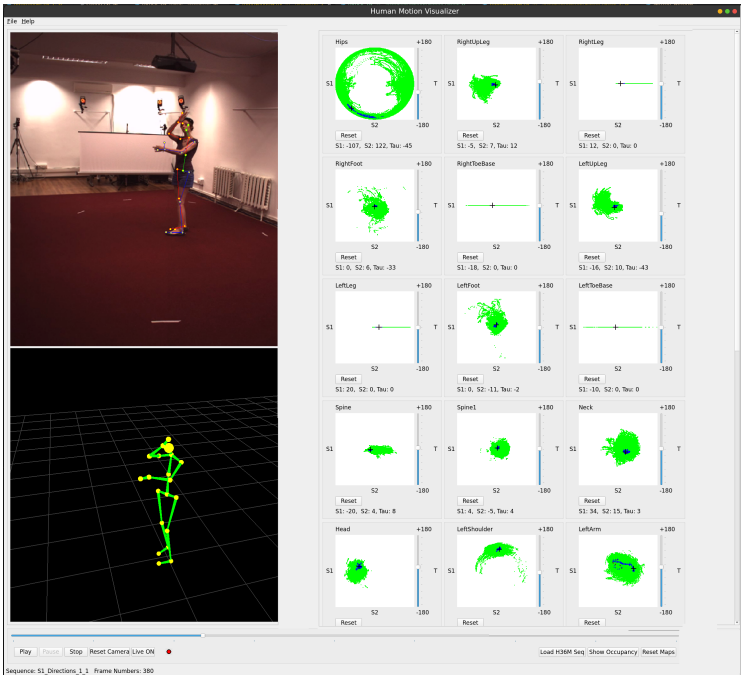


Figure 3.9.: Interactive Dataset Analyzer tool showing 3D pose and all the look up maps learnt for each joint for the Human3.6M dataset [102]

		Akhter et.al [109]	
		Valid %	**Invalid %**
Ours	Valid	99.92%	0.08%
	Invalid	36.65%	63.35%

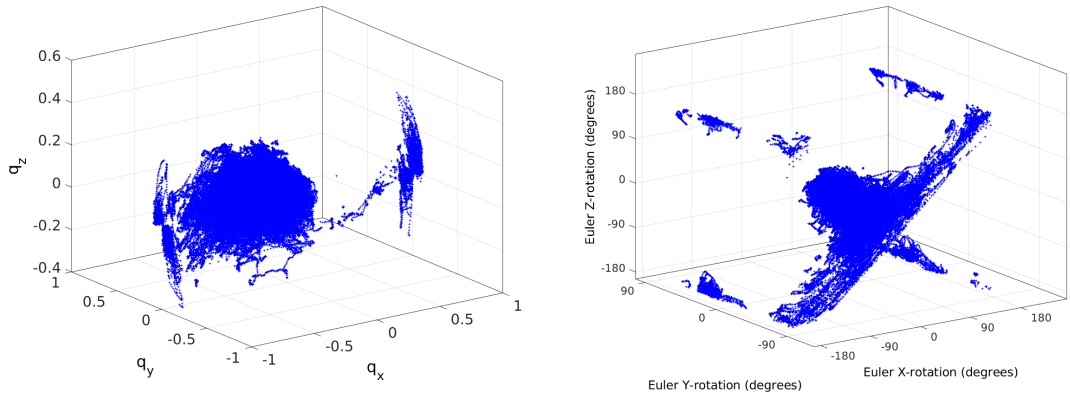
Table 3.1.: Classification comparison results between learned discriminator (ours) and Akhter et al. [109] on generated left shoulder synthetic dataset

3.5 Conclusion

The articulated movement of human body is constrained by the limits on 3D joint rotations. We present a formulation of swing and twist components of joint rotations in exponential maps to learn joint limits by data-driven approach. Our representation is used to encode valid poses in an intuitive and physically meaningful way. A discretized look-up map is derived to label valid/invalid rotations in parametrized joint space. And we train a set of neural network discriminators to classify valid and invalid rotations for different joints. Our trained discriminators used as priors will lead to accurate learning of 3D human pose estimation and motion synthesis. The learned joint constraints are crucial in representing normal range of human motion with meaningful joint limits for social affordance applications like vision-based human-robot interaction or monitor a patient's rehabilitation with joint injuries. Since the learning is performed using a data-driven approach, the quality of learned constraints depends on the range of motion available in the mocap data. Therefore, the

Joint Name	Accuracy %	False Valid %	False Invalid %
RightupLeg	98.7	0.3	1.0
RightFoot	97.7	1	1.3
LeftUpLeg	98.9	0.3	0.8
LeftFoot	98.2	0.6	1.2
Spine	99.4	0.1	0.5
Spine1	99.4	0	0.6
Neck	94.4	2.3	3.3
Head	97.7	0.9	1.4
LeftShoulder	97.9	0.7	1.5
LeftArm	96.8	0.5	2.7
LeftHand	94.7	2.1	3.3
RightShoulder	95.1	1.5	3.3
RightArm	95.5	1.63	2.9
RightHand	95	2	3

Table 3.2.: The accuracy of the discriminator networks for different joints on synthetic test dataset generated using look-up maps of Human3.6M dataset [102]



(a) (b)

Figure 3.10.: The Figure (a) shows the limitation of Quaternion (vector part) space of Right Hip (Right-Up-Leg) used by Herda et al. [114] shows duality of valid joint rotation. Anti-podal points +q & -q represent same rotation. Figure (b) illustrates artifacts in Euler Angle space of Right Hip (Right-Up-Leg) shows non-contiguous clusters of Valid Joint Rotation. Also boundary jumps at 180 degrees (for Z and X) and 90 degrees for Y rotation are visible

current method can be extended by capturing more data or using more publicly available motion capture datasets. Our visualization tool also guides collection of motion data close to constraints. A more comprehensive qualitative study to validate synthetic invalid poses with the help of human evaluation would further result in better ground truth generation for any motion capture dataset.

Deep 3D Human Joint Rotation Regression

4

> *"The key to artificial intelligence has always been the representation."*
>
> — **Jeff Hawkins**
> Neuroscientist

The structuring and representing information in the realm of artificial intelligence is of foundational significance. The representation refers to how knowledge, data, or features are modeled and organized to enable machines to understand, reason, and make decisions. The choice of representation plays a critical role and significantly influences the performance and capabilities of an AI system. A well-designed representation facilitates the extraction of meaningful patterns, relationships, and dependencies in the data, enabling the system to reason and generalize better.

4.1 Introduction

Recent approaches towards 3D human pose focus on estimating 3D joint positions instead of 3D joint rotations [117, 118, 119, 120]. Even the ones that use a 3D joint rotation based kinematic model do so implicitly [18, 121]. These approaches falls short in addressing pose ambiguity due to two key reasons. First, the bone lengths may vary and need to be regularized. Second, the twist around a body segment is not explicitly learned. When the twist angle information is lacking or imperfect, the dense pose or mesh rigged to that body part is no longer realistic.

Furthermore, the intra and inter-subject variations and range of activities make the problem even harder. Despite these confounding variables, the temporal context and constrained spatial relationship between adjacent body parts make 3D human pose estimation still tractable. However, the scalability and real-time implementation remains a major concern. Therefore, the prediction of a realistic 3D human pose from monocular images often requires additional constraints. Thus, we diverge from the common practice of predicting 3D joint positions; instead, we train a model to estimate the 3D joint rotations for a given standard 3D skeleton model.

The choice of regressing 3D joint rotation representations for joint kinematics is justified on following grounds:

- Deep neural networks (DNNs) are now widely used to conduct rotation regression for different computer vision tasks such as skeletal motion synthesis [122], structure from motion, pose estimation from images [123, 124] and point clouds [125]. The 3D rotations can be expressed in variety of ways including 3D or 4D representations such as quaternions, axis-angles, and euler angles [126]. These representations, however are insufficient for network regression, when the entire rotation space is required. According to empirical findings, converged networks do exhibit significant errors at specific rotation angles, pointing to more serious topological problems related to the continuity of the rotation representations [127].

- The focus of recent deep learning approaches [117, 118, 119, 120] is to recover 3D joint positions which provide sparse and insufficient pose information for many applications like motion re-targeting, virtual avatar/character animation and human-robot social affordance. [128] shows that when training is only supervised by 3D joint position loss, the human mesh models visualized have twisted body segment surfaces. Although the joints positions are predicted accurately, the joint rotation parameters are erroneous resulting in invalid poses[128].

- To this end, the human model based generative approaches that use a 3D joint rotation based kinematic model [18, 121] to improve upon 3D joint position only do so implicitly by using a re-projection loss of estimated 3D to 2D joint positions.

- The 3D joint rotations are scale invariant and independent of dataset. Naturally the 3D orientations of limbs are what completely differentiate one pose from another. Thus decoupling the 3D orientations from limb length ratios till post processing, is a sound proposition to improve 3D human pose estimation.

4.2 Overview

The primary focus of our work is to investigate how best the 3D human joint rotations can be recovered from monocular images. We present two different 3D angle regression models to estimate 3D joint rotations based on final output representation of the swing space. The training process involves employing a deep stacked hourglass network within a supervised learning setup, aiming to accurately regress the 3D joint rotations (see Figure 4.1).

Our unique approach involves estimating 3D human pose using a representation based on intuitive swing-twist decomposition for 3D joint rotations. Leveraging the anatomical constraints inherent in human joint rotations, our swing-twist decomposition readily incorporates these constraints. The result is a set of estimated human poses that consistently align with anatomical realism, ensuring that they never exceed the joint's natural range of motion.

Our novel contributions are as follows:

- We present a novel deep 3D human pose model regressing directly 3D joint angles in swing-twist representation.
- We integrate data driven joint constraints in intuitive swing-twist space, thereby ensuring estimated pose are always realistic and never exceed the joint range of motion.

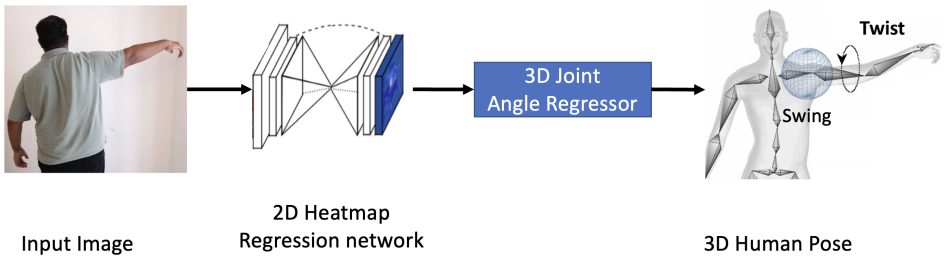


Figure 4.1.: Illustration of Deep 3D human pose regression using swing-twist decomposition for a right shoulder joint.

The estimated 3D pose in terms of joint rotations can be used directly for motion re-targeting to virtual avatars in virtual reality environments, character animation or for human action recognition. The social affordance in human-robot interaction and especially collaborative robotics (cobots) applications can also directly benefit from it without any need of inverse kinematics.

4.3 Related Works

The deep learning methods unite the strength of the large human motion datasets [18, 121, 129, 118, 119, 130, 131, 132, 133, 134] and hierarchical feature learning. In essence, the estimation of 3D human pose from monocular images (or image sequences) is dealt with using both discriminative [135, 118] and iterative fit [18] methods. The discriminative approaches try to learn the direct regression of 3D human pose from 2D images using latent pose space. The iterative fit approaches learn the 3D pose as an re-projection fit on the 2D image, starting with a good guess that may come from former approach. In either case, a parameterized kinematic layer or model [135, 18] may be used to incorporate structured prediction and constraints, but the learning is end-to-end and the intermediate kinematic vector remains latent.

However, the objective in the past has been to decrease the error metrics on 3D joint positions of human body. The inclusion of a kinematic layer/module helps to learn a realistic 3D human pose meeting anatomical constraints. The error itself is however minimized on output 3D joint positions and not the latent kinematic pose (which encode joint rotations in SE(3) space). The kinematic poses defined on SE(3) are not always unique to 3D joint positions. This many-to-one ambiguity often leads to erroneous pose, especially in terms of twist around a body segment. The anomaly is not apparent in a stick based visualization of 3D human pose, but would nevertheless lead to shape distortions around joints, if a

surface mesh is rigged to the body part. The twist motion (along kinematic chain) is also essential to predict the correct 3D orientation of body parts at extremity such as hands, head or feet. The twist error is not reflected in the 3D joint position error and often goes unreported in the current literature.

The literature in computer graphics (CG), deals with 3D human pose in more perceptive manner. The character animation and realistic motion synthesis requires that human body meets the anatomical constraints on its articulation. These constraints are often represented in swing-twist formulation of human pose. The work by [109] uses only the swing part and incorporate joint limits as pose priors in estimation of human pose. The pose synthesis by [107] includes a kinematic layer, but the error is again optimized on output 3D positions. The recent work [136] has emphasized the limits on joint swing-twist space to discriminate valid human poses from invalid ones. In this chapter, we develop a deep learning model to regress the 3D pose in terms of swing-twist representation of joint rotations, while incorporating the anatomical limits on articulation. *Note that, the focus of the current work is to investigate the efficacy of the swing-twist representation on a 3D human pose learning compared to other 3D skeletal joint rotation representations instead of benchmarking pose-estimation accuracy with various deep neural network architectures on a given 3D pose dataset.*

4.4 Methodology

In the current work, a 3D human pose is represented by a set of joint rotations, one for each joint with three degrees of freedom (3-DoF). We propose a framework which takes a single (or a series) 2D monocular RGB image of a human and outputs a corresponding 3D human skeleton commonly referred as 3D human pose. The swing-twist formulation is used to present 3D joint rotation of a joint in the skeleton. A forward pass through our end-to-end deep learning framework is capable of generating 3D swing-twist pose vector for each 2D image. We perform the forward kinematics by applying 3D joint rotations along the kinematic chain to get the resultant 3D human pose. Our approach is scale free and invariant to depth. The size of skeleton and body parts are obtained independently (e.g. through an anthropometric model).

The implementation of deep neural network architecture for 3D human pose can be decomposed into two sub-networks; the first one can recover human pose as 2D key points, and the second estimates 3D joint angle values for every joint of the recovered 2D pose. This multi-stage cascaded structure is capable of being trained on 2D and 3D labels in an end-to-end fashion using shared feature representations between the two sub-networks. We use 2D pose datasets having rich *in-the-wild* information to estimate 2D key points and for generalizing the model to perform better during *in-the-wild* scenarios.

Whereas 3D pose datasets, made up of *in-the-lab* motion captures, are used to fine-tune the model to estimate 3D joint angle values. Existing methods suggest either a single stage approach to directly regress 3D joint positions from raw pixel values or a multi-stage

approach including derivation of 3D joint positions from 2D key points [18, 133]. Direct regression of 3D joint locations is a challenging and non trivial problem. Additionally, it fails to capture an underlying sense of human anatomical structure and inter-joint relationship [137]. We therefore focus on using the multi-stage approach but instead of lifting 2D pose to *3D joint positions* in the second stage, we estimate *3D joint angle values*.

We present two variant of angle regression module in order to easily integrate with a multi-stage deep network. The network is trained in a supervised manner to regressing 3D joint angles expressed in terms of swing-twist representation (s_1, s_2, τ).

The variants defer on how the estimated swing (s_1, s_2) is represented. The first variant estimates swing as a 2D heatmap in swing space where as the second variant estimates swing and twist together as a single 1D vector for each frame. The first architecture is referred as *HeatmapReg*, while the second architecture is termed as *VectorReg* model henceforth in this paper for brevity.

4.4.1 HeatmapReg

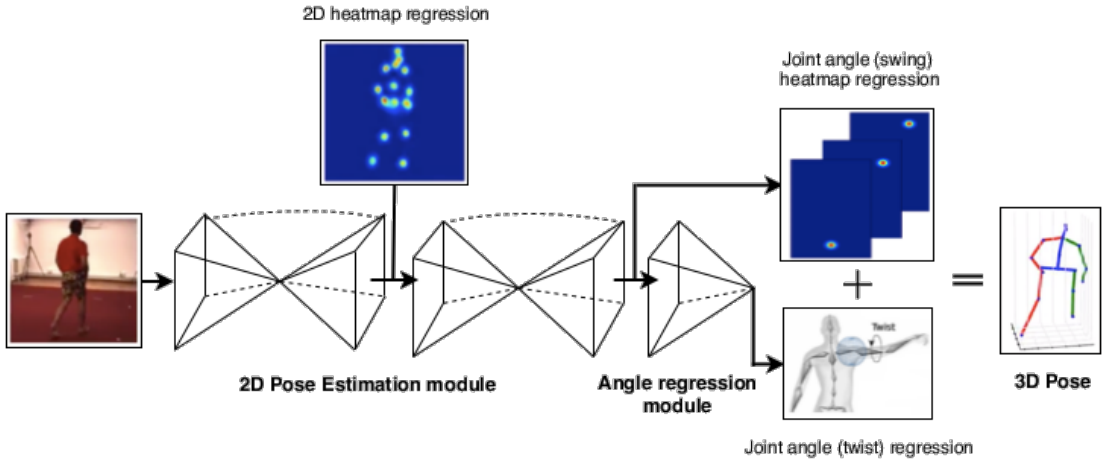


Figure 4.2.: HeatmapReg: Illustration of stacked hourglass network modules regress 16 swing heatmaps and twist joint orientations to construct the complete 3D human pose.

The stacked hourglass architecture, proposed by Newell et al. [138], stands out as a widely adopted network known for effectively capturing both local and global cues at various scales. This architecture has proven particularly successful in precise human pose estimation. In our research, we specifically chose the stacked hourglass architecture [121] for the estimation of 2D key points, integrating it as the first sub-network. Subsequently, we introduced our swing-twist angle regression module as the second sub-network appended at the end.

The configuration of our stacked hourglass network is meticulously designed to enable the independent processing of features within each module. This architectural choice ensures that the output of one module seamlessly serves as the input for the subsequent

one. In pursuit of end-to-end learning, we drew inspiration from the work of Zhou et al. [121] and investigated the potential application of their weakly-supervised method. This method allows for the concurrent learning of 2D key points and 3D joint angles.

We affirm the findings of [121] by demonstrating that augmenting an angle regression module with intermediate feature representations and the 2D pose output from a stacked hourglass architecture contributes to the regression of 3D joint angle rotation values. To supervise the learning process for 2D key points, we employed conventional intermediate supervision using ground truth synthesized as a set of heatmaps. Each pixel within these heatmaps signifies the probability of being associated with a 2D joint.

The final output of our network is a concatenation of the outputs from the stacked hourglass sub-network and the angle regression module. The swing heatmaps, generated from the second stack of the hourglass network, are provided as input to the angle regression module to estimate the 1D angle regression vector for twist, as illustrated in Figure 4.2.

In the first stage of both architectures, the objective loss function is defined as follows:

$$\mathsf{L}_{2D}(Y',\, Y) = \sum_{i=1}^{k} \left\| Y'_{2D} - Y_{2D} \right\| \tag{4.1}$$

Here, k represents the number of 2D joint heatmaps, Y'_{2D} denotes the estimated 2D joint position, and Y_{2D} corresponds to the ground truth for the same. This loss function quantifies the dissimilarity between the estimated and ground truth 2D joint positions, serving as a pivotal metric in evaluating the performance of the first stage in both architectural frameworks.

In the second stage of our model, the loss function for the 2D swing (s_1, s_2) heatmap is defined as follows:

$$\mathsf{L}_{swing}(S',\, S) = \sum_{i=1}^{k} \left\| (S'_{(s1,s2)} - S_{(s1,s2)} \right\| \tag{4.2}$$

Here, k denotes the number of joint swing heatmaps, $S'_{(s1,s2)}$ represents the estimated swing in the (s_1, s_2) space, and $S_{(s1,s2)}$ is the corresponding ground truth. Additionally, the loss for the twist τ is obtained through the following equation:

$$\mathsf{L}_{twist}(T'_{\tau},\, T_{\tau}) = \sum_{i=1}^{k} \left\| T'_{\tau} - T_{\tau} \right\| \tag{4.3}$$

Here, k signifies the number of joint twist parameters, Y'_{τ} is the predicted twist angle of a body part, and Y_{τ} is the corresponding ground truth. The overall loss function is defined as:

$$\mathsf{L}_{global} = L_{2D} + w_1 L_{swing} + w_2 L_{twist} \tag{4.4}$$

where $w_1 = 0.5$ and $w_2 = 0.3$ are derived empirically. In the case of 2D keypoint and swing heatmaps (S_1, S_2), the final stack(s) of the hourglass sub-networks undergo post-processing using the *maximum likelihood* function to generate a set of likelihood maps. It is

essential to note that detection-based methods, relying on the pixel with maximum likelihood from the map for each joint, may encounter quantization errors during the transition from heatmap resolution to the expected output resolution, as highlighted by Sun et al. [139]. To mitigate quantization errors, Sun et al. suggest operating heatmap learning at higher resolutions. However, this approach comes at the cost of increased GPU computations, model size, and storage space [139].

4.4.2 VectorReg

The VectorReg variant introduces a different approach by regressing both swing and twist angles as a single 1D vector from the 3D angle regression module. In this configuration, 2D joint positions are still estimated as heatmaps and are concatenated with other intermediate image features to input an angle regression module, which regresses both swing and twist as a pose vector from an encoder sub-network, as illustrated in Figure 4.3. The loss formulation for this configuration differs only in the second stage, angle regression module (encoder), and is given by:

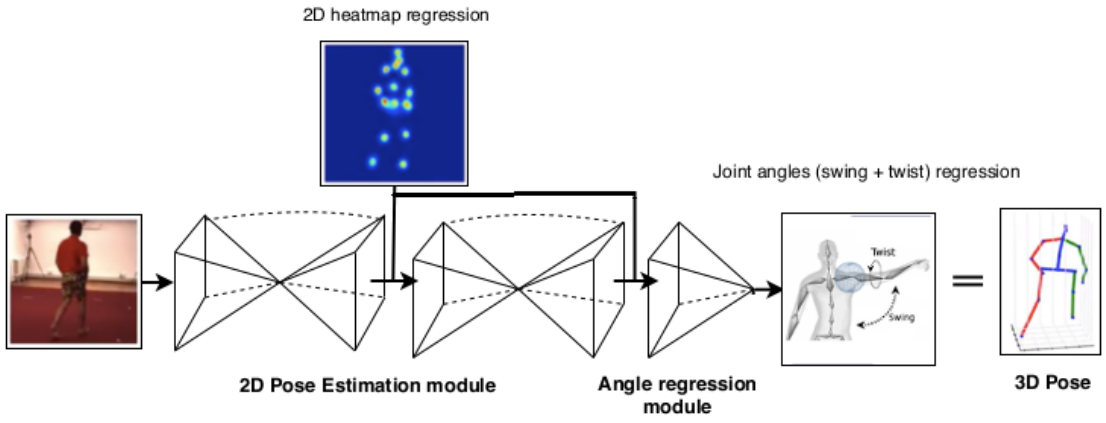


Figure 4.3.: VectorReg: Illustration of stacked hourglass network modules estimate 16 swing-twist joint orientations to construct the complete 3D human pose.

$$\mathsf{L}_{3DA}(Y',\, Y) = \sum_{i=1}^{k} \left\| Y'_k - Y_k \right\| \tag{4.5}$$

Here, k represents the number of joints, and Y_k and Y'_k are the ground truth and predicted swing-twist values $(s1, s2, \tau)$ of a joint, respectively. The objective function minimized to train an end-to-end network is:

$$\mathsf{L}_{global} = L_{2D} + w_1 L_{3DA} \tag{4.6}$$

Here, w_1 is set to 0.5 empirically. The vector-based regression in VectorReg alleviates the challenges associated with quantization problems, offering end-to-end learning with the

elimination of the maximum likelihood post-processing step. As discussed in Section 4.7.1, VectorReg outperforms the HeatmapReg architecture.

4.5 Implementation

The implementation of the proposed network was carried out in PyTorch [140], leveraging the hourglass architecture as the foundation, which was adapted from publicly available code [121]. Specifically, a shallow variant of the stacked hourglass architecture was employed, incorporating 2 stacks, each comprised of 3 residual modules [141]. The angle regression module, essential for joint angle estimation, featured 4 sequential residual and pooling modules (refer to the supplementary section for comprehensive details).

Training

The training of both the swing-twist heatmap regression network and the vector regression network was conducted using data from the Human3.6M dataset [131], encompassing 16 distinct activities observed through four cameras. Notably, subjects [S1, S5, S6, S8] were designated for the training phase, while subjects [S9, S11] were reserved for testing purposes.

The training process unfolded in two stages. Initially, 2D datasets, including MPII [142], LSP [143], and COCO [101], were employed for 120,000 iterations with a batch size of 6. This initial stage aimed at pre-training a 2D pose estimation module, aligning with the approach established in [121]. The subsequent stage involved 200,000 iterations on the Human3.6M dataset training set, specifically tailored to estimate 3D pose. The entire training regimen spanned approximately two days, executed on a dedicated GTX 1080 Ti GPU utilizing CUDA 8.0 and cuDNN 7 frameworks. The computational efficiency of the trained model was evident during testing, with a forward pass for 3D pose estimation averaging at 30ms. Noteworthy, data augmentation and hyper-parameters recommended by Zhou et al. [121] were employed consistently throughout the training. The uniformity in network architecture and training iterations underscored the integrity of our experimental framework.

In the subsequent sections, we delve into the evaluation metrics, results, and discussions arising from the implementation and training of our proposed network.

4.6 Evaluation Results

In this section, we delineate the evaluation metrics and methodologies used in our experimental evaluations. To facilitate a direct and meaningful comparison, we maintained an identical setup as detailed in [144]. Notably, our optimal results were obtained through a model that incorporates the 3DPW training dataset within our training loop. Even in the

absence of 3DPW training data, our method demonstrated superior accuracy compared to the prior state-of-the-art (SOTA).

For the assessment of our model's performance on the Human3.6M test set, we adhered to the established practice of utilizing the Human3.6M training set, ensuring consistency with prior research. Intriguingly, our most proficient model on 3DPW did not rely on Human3.6M training data, indicating a nuanced relationship between dataset-specific training and in-the-wild pose estimation performance.

Evaluation Metrics

To gauge the efficacy of our approach, we used standard evaluation metrics tailored to each respective dataset. The Mean Per Joint Position Error (MPJPE) served as a fundamental metric, computed as the mean of Euclidean distances between ground-truth and predicted joint positions. This calculation involved centering the pelvis joint on the ground truth location, aligning with common practice. Additionally, we reported the Procrustes Aligned MPJPE (PA-MPJPE), a variant of MPJPE that considers a rigid alignment of the predicted pose to the ground-truth pose.

The impact of different joint rotation representations on the performance of our model, we conducted an ablation study. Various representations, including Euler angles, quaternions, exponential maps, and swing-twist, were individually employed for training distinct models. This study aimed to validate our selection of a representation for regressing joint angles. To ensure a fair comparison, predictions from all models were converted to 3D positions using a forward kinematic approach. The estimated 3D joint angles on a fixed skeletal model were utilized to obtain 3D positions, and the mean per joint position error (MPJPE) in millimeters was then compared. Table 4.2 reports the MPJPE scores across 15 activities from the Human 3.6M test set. The results indicate that models trained using the swing-twist parameterization consistently outperform others, with the lowest error, followed by quaternion, Euler angle, and axis angle representations, respectively.

4.6.1 Quantitative Results

Our trained models were subjected to evaluation on subjects S9 and S11 from the Human3.6M dataset during testing. The mean swing-twist error for all activities across the specified test set was measured with respect to the ground truth.

Given that a significant portion of current state-of-the-art 3D human pose estimators focuses on regressing 3D skeletal joint positions, there is a lack of benchmarking standards for 3D joint angles in existing research literature. Consequently, we present evaluation scores that allow for a comparison between our different approaches.

We evaluate the model using Angle error of axis-angle representation (refer Appendix section D.5). We achieved the best mean joint angle error score of **9.43 degrees** for (s_1, s_2, τ) on Human3.6M, encompassing subjects S9 and S11, using a vector regression

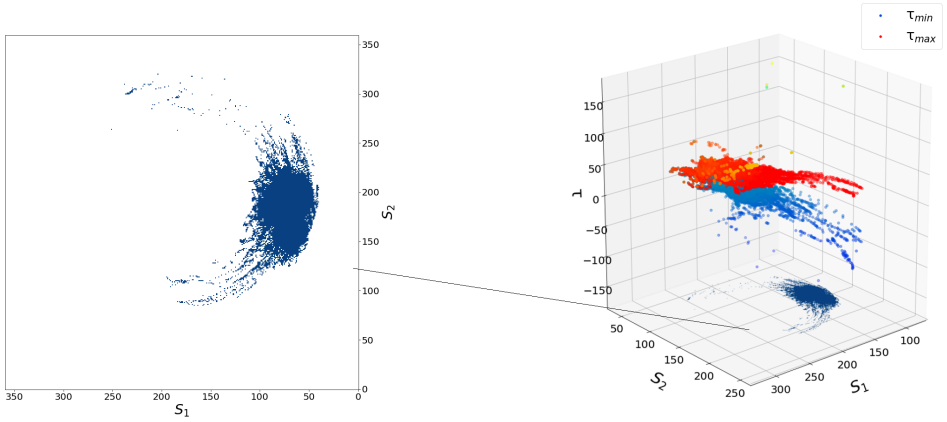


Figure 4.4.: The learned swing-twist space joint constraints [136] representing valid swing occupancy map (bottom) and twist range map [τ_{min},τ_{max}] (top) for the left shoulder joint orientation. The red points represent the upper bound (τ_{max}), and blue points represent the lower bound (τ_{min}) of the twist angle for a given valid swing (S_1, S_2) value.

	Euler angle	Quaternions	Exponential Map	Swing - Twist
Directions	0.310	0.364	0.884	**0.281**
Discussion	0.326	0.334	0.819	**0.257**
Eating	0.560	0.530	0.993	**0.445**
Greeting	0.330	0.359	0.855	**0.268**
Phoning	0.608	0.480	0.991	**0.383**
Photo	0.453	0.490	0.944	**0.408**
Posing	0.411	0.337	0.867	**0.242**
Purchases	0.416	0.341	0.804	**0.261**
Sitting	0.596	0.541	0.934	**0.486**
SittingDown	0.743	0.595	0.975	**0.514**
Smoking	0.501	0.459	0.953	**0.370**
Waiting	0.342	0.411	0.939	**0.303**
WalkDog	0.481	0.366	0.817	**0.280**
Walking	0.317	0.318	0.848	**0.206**
WalkTogether	0.537	0.293	0.810	**0.208**
ALL	0.462	0.415	0.896	**0.327**

Table 4.1.: Activity-wise mean Euler angle error values in radians for models trained on different joint angle parameterizations.

network. Notably, the vector regression network demonstrated lower error in 3D joint angle prediction compared to heatmap regression (refer to Table 4.6). Figure 4.5 visually compares the swing-twist angles $(s1, s2, \tau)$ predicted by the vector-based regression (in red) for the left shoulder joint in a "Directions" activity of subject S9 from the Human3.6 dataset, juxtaposed with the ground truth (in blue). Despite noise in the ground truth, the 3D angle estimation closely follows the trend in ground truth. However, estimation artifacts are evident in cases of partial or full occlusions of a body segment.

3D Mean Per Joint Positions Error (MPJPE): The evaluation of 3D human joint positions was conducted on test subjects S9 and S11 from the Human3.6M dataset. The 3D joint positions were obtained after performing forward kinematics on a template skeleton using regressed joint angle values. We adhered to three protocols designed to benchmark 3D joint position errors (in mm) against state-of-the-art methods [145]. We

extensively evaluated our model to compute 3D joint position errors following all three standard protocols. The Mean Per Joint Position Error (MPJPE) in millimeters between the estimated pose and the ground-truth pose was measured, and Procrustes alignment was utilized to align the predicted pose to the ground truth.

Accurate joint angle regression for hip and spine joints is exceptionally challenging, and the evaluation scores for these joints reflect this difficulty. Consequently, child joints suffer in terms of joint position accuracy when forward kinematics is performed, propagating the impact to the end joint positions and affecting the entire 3D pose. To mitigate this, we substituted ground truth for the root joint angle values, ensuring a fair evaluation of the other 15 joint positions, each equally contributing to the final 3D human pose. Therefore, Table 4.6 excludes the root joint error. Additionally, per joint-specific errors reported in the research literature for benchmarking the estimated accuracy of our 3D human pose were not found (refer to Table 4.4). Benchmarking is often restricted to mean error across activities in MPJPE values. The lower MPJPEs do not necessarily take into account twist present around the segment in a pose while evaluating 3D position regression-based methods. Comparing the accuracy of our method solely based on the reported MPJPE metric is limiting to swing accuracy only.

Table 4.2 showcases the choice of 3D joint rotation representation. We measured the mean 3D joint position errors (MPJPE in mm) of each network trained separately on Euler Angles, Quaternions, Axis-angle, and Swing-Twist. The evaluation was conducted over Protocol #1 on the Human3.6M dataset across all activities.

The Table 4.6 provides a comparison of the mean Swing-Twist angle error for HeatmapReg and VectorReg architectures with and without constraints in degrees. The error is measured over all activities performed by test set Subjects S9 and S11 from the Human3.6M dataset.

3D Joint Rotation Error: The impact of different joint rotation representations on the performance of our model, we conducted an ablation study. Various representations, including Euler angles, quaternions, exponential maps, and swing-twist, were individually employed for training distinct models. This study aimed to validate our selection of a representation for regressing joint angles. To ensure a fair comparison, predictions from all models were converted to 3D positions using a forward kinematic approach. The estimated 3D joint angles on a fixed skeletal model were utilized to obtain 3D positions, and the mean per joint position error (MPJPE) in millimeters was then compared. Table 4.2 reports the MPJPE scores across 15 activities from the Human 3.6M test set. The results indicate that models trained using the swing-twist parameterization consistently outperform others, with the lowest error, followed by quaternion and euler angles representations respectively.

4.6.2 Qualitative Results

In this section, we present qualitative results to illustrate the performance of our proposed 3D human pose estimation model. Figure 4.6 showcases the estimated 3D poses (depicted

Method	Euler Angle	Quaternions	SwingTwist
Direct	89.78	90.82	**56.25**
Discuss	96.06	95.39	**60.54**
Eat	123.9	125.6	**59.49**
Greet	100.6	106.3	**64.45**
Phone	124.3	121.9	**66.77**
Pose	94.05	86.25	**57.92**
Purch	106.4	108.3	**62.29**
Sit	144.2	143.4	**84.75**
SitDown	180.2	170.0	**107.09**
Smoke	132.7	120.2	**71.19**
Photo	128.9	132.7	**77.75**
Wait	103.6	107.1	**62.66**
Walk	108.5	112.6	**70.52**
WalkDog	126.3	122.2	**55.92**
WalkTgth	102.2	97.47	**60.99**
ALL	117.0	115.4	**67.63**

Table 4.2.: Choice of 3D joint rotation representation: Mean 3D joint positions errors (MPJPE in mm) of each network trained separately on Euler Angles, Quaternions and Swing-Twist. Evaluation over Protocol #1 on Human 3.6M Dataset [131] across all activities.

Method	Hip	Knee	Ankle	Spine	Neck	Head	Shoulder	Elbow	Wrist	Mean
HeatmapReg	12.31	13.45	9.15	8.14	16.02	16.38	10.22	14.73	12.76	12.42
HeatmapReg-wc	11.30	13.32	9.93	9.19	17.43	16.67	10.24	15.37	13.30	12.4
VectorReg	10.57	13.08	9.21	7.09	13.74	13.56	8.83	13.41	11.38	10.82
VectorReg-wc	**8.01**	**9.90**	**6.27**	**6.24**	**13.41**	**13.08**	**7.12**	**11.65**	**9.67**	**9.43**

Table 4.3.: The comparison of mean Swing-Twist angle error for HeatmapReg and VectorReg architectures with constraints(HeatmapReg-wc,VectorReg-wc) and without constraints in degrees. We measure the error over all activities preformed by test set Subjects [S9,S11] from Human 3.6M Dataset [131]

Protocols	Root	Hip	Knee	Ankle	Spine	Neck	Head	Shoulder	Elbow	Wrist	Mean
Protocol #1	35.18	39.34	65.86	103.63	32.92	37.87	54.17	39.40	75.84	136.96	67.63
Protocol #2	34.20	38.68	65.38	102.02	32.97	38.79	53.10	40.29	76.81	135.49	67.27
Protocol #3	35.66	39.60	62.36	100.09	36.07	40.28	53.33	41.34	77.88	136.27	67.52

Table 4.4.: Joint-wise evaluation of 3D joint positions errors (MPJPE in mm) - Protocol #1, Protocol #2 and Protocol #3

in red) for RGB frames in the top row, extracted from the Human3.6M dataset. The corresponding ground truth 3D poses are shown in blue. The first two cases on the left demonstrate successful 3D pose estimations, even in the presence of occlusion (second from the left, where the right upper leg is partially occluded). However, the last two cases on the right depict instances of failure in 3D pose estimation.

The accuracy of our 3D pose estimation heavily relies on the quality of the 2D joint position heatmaps obtained in the first stage (stack 1 and 2) of the hourglass network. In the failure cases, the 2D joint heatmaps (not shown) exhibit more uncertainty compared to cases where accurate 3D pose estimation is achieved. The modular nature of our architecture allows for the adaptation and selection of a more suitable 2D pose estimation

	Euler angle	Quaternions	Exponential Map	Swing - Twist
Directions	0.310	0.364	0.884	**0.281**
Discussion	0.326	0.334	0.819	**0.257**
Eating	0.560	0.530	0.993	**0.445**
Greeting	0.330	0.359	0.855	**0.268**
Phoning	0.608	0.480	0.991	**0.383**
Photo	0.453	0.490	0.944	**0.408**
Posing	0.411	0.337	0.867	**0.242**
Purchases	0.416	0.341	0.804	**0.261**
Sitting	0.596	0.541	0.934	**0.486**
SittingDown	0.743	0.595	0.975	**0.514**
Smoking	0.501	0.459	0.953	**0.370**
Waiting	0.342	0.411	0.939	**0.303**
WalkDog	0.481	0.366	0.817	**0.280**
Walking	0.317	0.318	0.848	**0.206**
WalkTogether	0.537	0.293	0.810	**0.208**
ALL	0.462	0.415	0.896	**0.327**

Table 4.5.: Activity-wise mean Euler angle error values in radians for models trained on different joint angle parameterizations.

network, which could potentially reduce the frequency of failure cases or mitigate the severity of errors.

Figure 4.5 specifically focuses on the vector regression estimates (in red) for the swing-twist angles (S_1, S_2, τ) of the left shoulder joint during a "Directions" activity of Subject S9 from the Human3.6M Dataset [131]. The corresponding ground truth values are shown in blue. This provides a detailed view of the model's accuracy in estimating specific joint angles.

4.7 Additional Experiments

In this section, we perform additional experiments to measure the impact of 3D joint angle regression on the overall regression accuracy. Additionally, we implement a 3D pose estimation pipeline to lift 2D keypoints to 3D pose. The baseline benchmarks the performance of representation space for 3D rotation representations of 6D and swing-twist. The baseline pipeline also measures the impact of forward kinematic layers used during training of the baseline regression task.

4.7.1 Effect of Using 3D Joint Angle Constraints

We also investigated the impact of incorporating 3D joint angle limits as *hard constraints* during the learning process on the final 3D pose estimation accuracy. Figure 4.7 illustrates the learned unambiguous anatomical joint constraints in (S_1, S_2) space through a 2D swing occupancy map acquired from the Human3.6M dataset [131]. For each valid swing joint rotation, upper (τ_{max}) and lower (τ_{min}) bounds are determined to understand the twist range of motion [136]. The model trained without constraints estimates the full range of motion in the Swing-Twist space, i.e., $[-\pi, \pi]$. The HeatmapReg-C and VectorReg-C

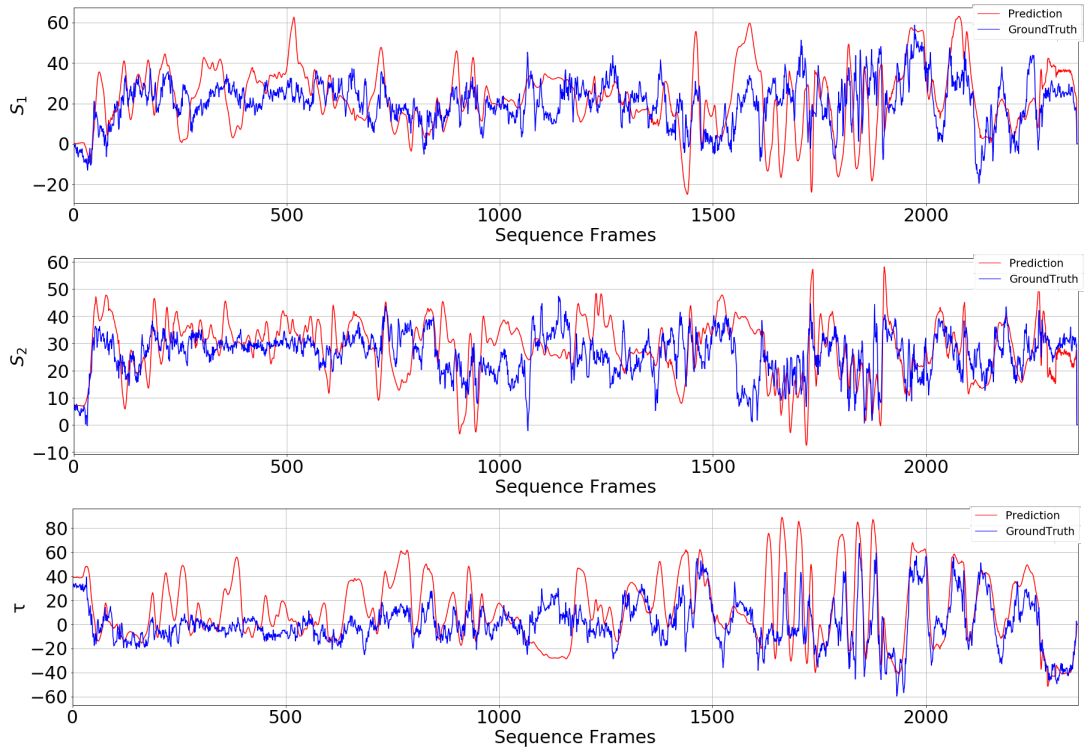


Figure 4.5.: Vector regression estimates (depicted in red) for swing-twist angles (S_1, S_2, τ) of the left shoulder joint during a 2-minute "Directions" activity of Subject S9 from the Human3.6M Dataset [131]. The ground truth values are represented in blue.

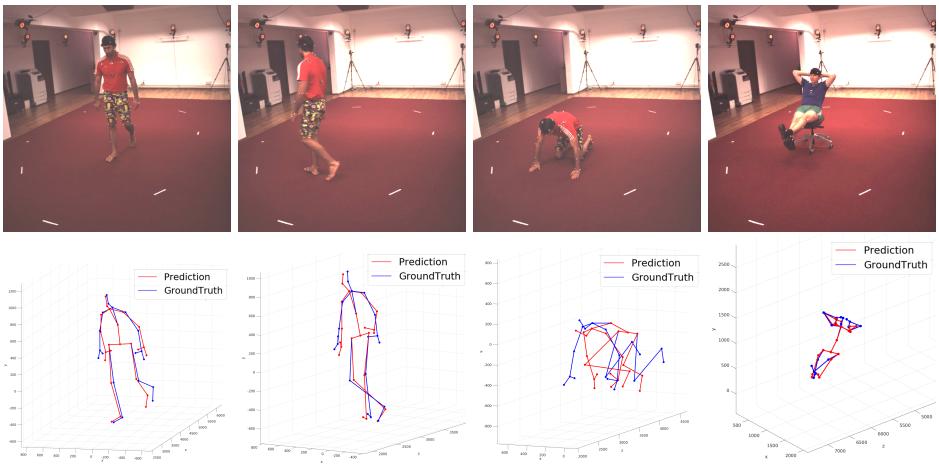


Figure 4.6.: The top row displays RGB frames of a motion sequence, and the bottom row shows the corresponding 3D human poses. Ground truth 3D poses are shown in blue, while predicted poses are depicted in red. The first two columns on the left demonstrate successful pose predictions, while the last two columns on the right exhibit predictions over challenging scenarios with occlusions.

models trained with valid range of motion restrict the output to adhere to the learned constraints over the dataset [136].

The benefits of using constraints to estimate 3D pose are summarized in Table 4.6. Table 4.6 presents the error comparisons to investigate the effect of imposing joint limits

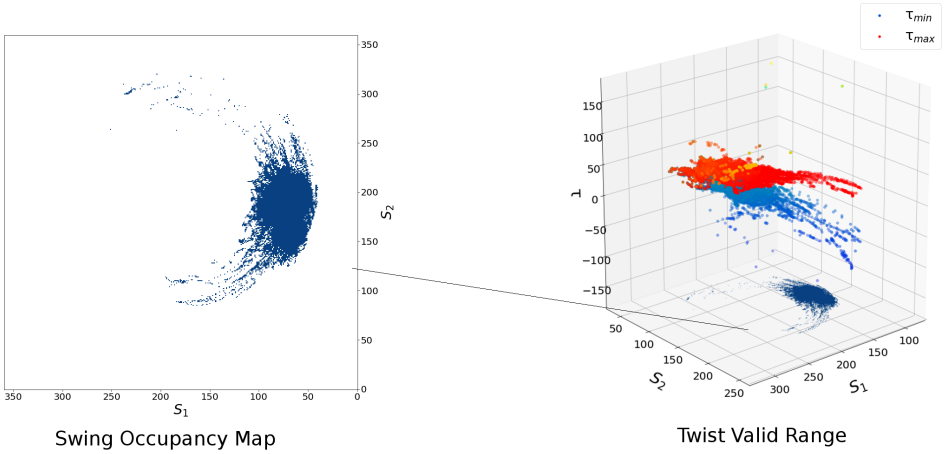


Figure 4.7.: The learned swing-twist space joint constraints [136] representing valid swing occupancy map (bottom) and twist range map [τ_{min},τ_{max}] (top) for the left shoulder joint orientation. The red points represent the upper bound (τ_{max}), and blue points represent the lower bound (τ_{min}) of the twist angle for a given valid swing (S_1, S_2) value.

in swing-twist space. Significantly improved accuracy was observed by incorporating constraints in swing-twist space.

	Heatmap Reg	HeatmapReg-C	Vector Reg	VectorReg-C
Directions	0.322	0.322	0.281	**0.265**
Discussion	0.292	0.294	0.257	**0.242**
Eating	0.473	0.483	0.445	**0.419**
Greeting	0.316	0.319	0.268	**0.250**
Phoning	0.404	0.421	0.383	**0.338**
Photo	0.429	0.424	0.408	**0.370**
Posing	0.293	0.306	0.242	**0.229**
Purchases	0.280	0.300	0.261	**0.240**
Sitting	0.516	0.508	0.486	**0.451**
SittingDown	0.555	0.573	0.514	**0.468**
Smoking	0.408	0.419	0.370	**0.343**
Waiting	0.344	0.356	0.303	**0.268**
Walk Dog	0.303	0.304	0.280	**0.257**
Walking	0.225	0.233	0.206	**0.181**
Walk Together	0.239	0.252	0.208	**0.192**
ALL	0.360	0.367	0.327	**0.301**

Table 4.6.: Comparison of mean Swing-Twist angle error for HeatmapReg and VectorReg architectures with constraints (HeatmapReg-C, VectorReg-C) and without constraints in radians. Errors are measured across all activities performed by test set Subjects [S9,S11] from Human 3.6M Dataset [131].

4.7.2 Baseline 3D Pose estimation

In this study, we established a baseline that takes 2D key-points as input and performs supervised regression to predict 3D positions. The motivation behind this baseline is to

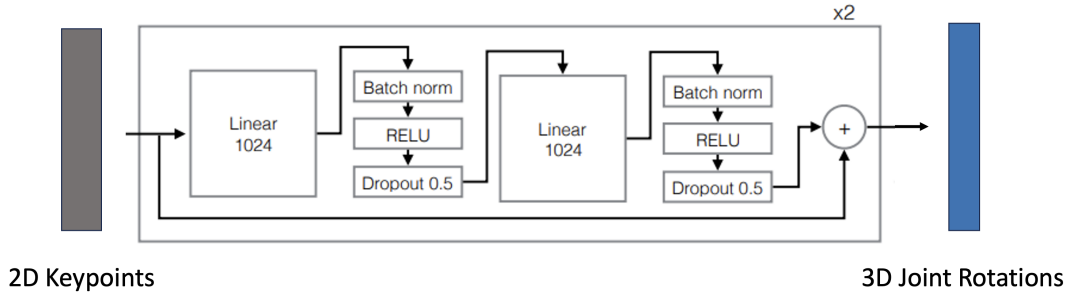


Figure 4.8.: The baseline network structure similar to [146]

achieve optimal performance and set an upper bound by using ground truth 2D key-points. Subsequently, we compare this baseline with [146], which predicts 3D positions from monocular images. We differ from [146] by lifting 2D key-points to 3D joint angles instead of 3D joint positions as target as shown in Fig 4.8 .

The experiment performed with the use of 3D joint angles surpasses the result of [146] with the same architecture and hyper-parameters. The authors report the error of 45.5 mm using ground truth 2D positions on Human3.6M dataset with rigid alignment in the post-processing. We achieve an error of 41.33 mm using 6D representations. The experiment is also extended for MPI_INF_3DHP and 3DPW datasets. The results of the upper-bound for these three datasets is shown in the table 4.7.

Model	Dataset	3D Pose Evaluation		
		PAMPJPE	**MPJPE**	**Angle error**
Martinez et. al. [146]	Human3.6M	45.5	-	-
Baseline (6D + FK)	Human3.6M	**41.33**	-	-
Baseline (6D + FK)	MPI_INF_3DHP	79.61	-	-
Baseline (6D + FK)	3DPWP	71.31	121.96	12.99

Table 4.7.: Evaluation of baseline lifting of 2D GT annotations to 3D pose. All architectures are trained for 100 epochs. MPJPE and PAMPJPE are in mm. Angle error is expressed in degree.

Analysis of Forward kinematic layer

In this section, Forward kinematic layer is implemented in the train phase to impose 3D joint position loss. The performed experiment provides comparative results for 6D and swing-twist joint representations on Human3.6m dataset.

We experiment with additional Forward Kinematic loss (FK_{loss}) in the training. The (FK_{loss}) is computes euclidean distance of each joint position obtained by applying joint rotations on normalized skeleton using forward kinematics. The (FK_{loss}) regularizes the cost function and restricts the model to learn joint locations in more confined space hence

Activity	Activity-wise PAMPJPE in mm			
	Swing-Twist	Swing-Twist + FK	6D	6D + FK
Directions	43.18	41.16	35.7	**32.39**
Discussion	42.92	42.98	39.53	**34.10**
Eating	48.42	45.09	43.76	**40.3**
Greeting	47.57	46.86	46.006	**40.08**
Phoning	48.71	49.8	44.41	**41.57**
Photo	66.78	64.75	63.63	**54.91**
Posing	46.49	43.94	41.52	**35.56**
Purchase	42.01	41.40	40.36	**35.42**
Sitting	53.64	53.33	51.34	**48.64**
Sitting Down	72.03	64.24	67.16	**64.45**
Smoking	51.07	51.7	45.83	**42.51**
Waiting	51.85	50.05	45.12	**40.45**
Walk Dog	53.37	53.93	48.78	**44.79**
Walking	42.42	36.89	22.52	**29.92**
Walk Together	48.53	45.52	40.02	**34.98**
Average	50.59	48.84	45.04	**41.33**

Table 4.8.: The activity-wise comparision of 3D mean per joint positions errors with procrustes alignment (PA-MPJPE in mm) for all the baseline methods trained on Human3.6M [131] (S1, S5, S6 and S8 and tested for S9 and S11) Training is done for all the models using same hyperparameters and 100 epochs.

giving better results. The similar idea is used in [78]. The training loss is set to as defined is in equation 4.7, where, L_{3DA} is the loss on 3D joint angles and FK_{loss} is the forward kinematic loss. λ is a hyper-parameter and is empirically set to 0.001.

$$L_{total} = L_{3DA} + \lambda * FK_{loss}. \tag{4.7}$$

We also model another experiment using a 6D representation with Forward Kinematics layer gives the best results. A quantitative analysis of all the experiments is in Table 4.8. We have reported the results after using Procrustes alignment in the post-processing.

From the above experiments performed on Human3.6M dataset as shown in Table 4.8, it can be concluded that *6D representation provides superior results with respect to swing-twist representation*. However, swing-twist decomposition provides an intuitive framework for analyzing rotational motion. The decomposition enables smoother interpolation of rotational motion and allows manipulating rotational orientations with mathematical simplicity. Direct regression from 2D positions to 3D positions gives 45.5 mm as reported in the work of [146]. However, using 3D joint angles with 6D representations we achieve 41.33 mm which is better than direct regression of 3D positions.

4.8 Conclusion

In this work, we introduced a deep learning module for 3D joint angle regression, expressed in terms of the swing-twist representation, from a single monocular RGB image. The learned

pose is more generic than estimating 3D joint positions and enables the prediction of a twist around a body segment, crucial for better 3D human body shape estimation. While estimating the twist of a body segment is more challenging than estimating the swing, our method achieves comparable results to models that regress 3D pose using 3D joint positions.

Quantitative evaluations in terms of mean joint rotation error and mean per joint position error (MPJPE) on the Human3.6M dataset [131] demonstrate that our approach offers comparable accuracy to state-of-the-art approaches in terms of 3D joint positions. Notably, the estimation error using swing-twist angles is lower than other 3D joint angle representations (except 6D), and the generalization to unseen poses is better. Additionally, the 3D position error using estimated joint rotations is comparable to other approaches on the Human3.6M dataset [131].

Our investigation further reveals that the learned twist has limited accuracy compared to the accuracy of the swing and is a subject for future exploration. This limitation arises from the intrinsic difficulty of estimating the twist of a body segment from 2D images. Additionally, we observed jitter in poses over frames with low joint angle changes due to the absence of any temporal model.

This marks the culmination of the preliminary segment of this thesis. Now that we have a solid grasp of generic 3D pose estimation modeling. In the next section, we will delve into modeling self-occlusion from an exocentric camera viewpoint. This exploration is focused on extracting pose and motion parameters in the context of competitive diving sports.

Acknowledgments

I want to convey my heartfelt gratitude to Dr. Hammad Butt and Dr. Sandesh Hiremath for their invaluable feedback and guidance throughout the development of the joint constraints work. Their insightful comments and constructive suggestions significantly enhanced the quality and precision of the content.

I also extend my thanks to students Siddharth Ingle and Ankit Kumar for their dedicated contributions to the experiments code base. Their efforts laid a robust foundation for the work presented in this preliminary section of the thesis.

Part II

Exocentric view: Self Occlusion



Illustration depicting a training session of a diving athlete, rigorously monitored by trainers for constructive feedback

Overview

Overview

Overview

In this section, we address the challenge of managing self occlusion within the exocentric perspective of the camera. We narrow our focus towards the extraction of pose parameters in the context of diving sports training videos. The research aims to extract pose and pertinent motion parameters from monocular RGB videos. The presentation of information in this part of thesis is outlined as follows:

- **Chapter 5** commences with an introduction that elucidates the challenges inherent in the realm of diving sports analytics. Subsequent sections provide an in-depth review of pertinent research and datasets within the domain of diving sports. This chapter comprehensively covers *2D and 3D diving pose* estimation models with evaluation on test diving datasets.
- **Chapter 6** is dedicated to the pivotal task of *diving action localization* within a continuous video stream, an essential prerequisite for benchmarking in the field of diving sports. The chapter meticulously outlines the methodology employed and furnishes an exhaustive evaluation aimed at accurately detecting diving actions.
- **Chapter 7** integrates the findings from the preceding two chapters, presenting a holistic framework for the extraction of diving motion parameters. The pose parameters encompass *center of mass trajectory tracking, homography computation, and body angle rotation* estimation. The chapter culminates with a rigorous evaluation of the estimated pose parameters

and introduces research tools developed for the meticulous analysis of diving sequences.

Related publications

[1] Pramod Murthy, Bertram Taetz, and Didier Stricker. “Low-latency detection of diving sports movements in training videos”. In: *20th Spring School "Technologies in Competitive Sports", Leipzig, Germany*. IAT. 2022.

[2] Pramod Murthy, Bertram Taetz, Arpit Lekhra, and Didier Stricker. “DIVENET: Dive Action Localization and Physical Parameter Extraction for High Performance Training”. In: *IEEE Access*. 2023.

[3] Youssef Farag. “Human Pose Estimation on Embedded Devices”. Media Engineering and Technology Faculty. Bachelor’s thesis. German University in Cairo, Sep. 2019.

[4] Asmita Mittal. “Probabilistic Modeling of Human Motion Dynamics”. Master’s thesis. Informatik, RPTU Kaiserslautern-Landau, Apr. 2021.

[5] Ashish Shukla. “Efficient 3D Human Pose Estimation on Mobile Devices”. Fakultät EIM, Institut für Elektrotechnik und Informationstechnik. Master’s thesis. Universität Paderborn, Sep. 2022.

Diving Sports Pose Regression

5

> *"When you see a good move, look for a better one."*
>
> — **Emanuel Lasker**
> German chess player, mathematician

Sports analytics using video is a popular and an emerging research area in computer vision [144, 147, 148, 149]. The analysis of sports videos is a challenging task due to complex body postures and manoeuvres involved in sports such as diving, gymnastics and balance-beam. Analyzing sports videos often need experts to visually observe the complex movements and provide feedback during the training. The body movements are rapid for the human eye to make a measurable observation for training coaches in order to train the athletes effectively. The body maneuvers involved in gymnastics, balance beams, and diving are too complex to be quantified in real-time.

In addition, monocular based human pose estimation often suffer from self-occlusion and motion blur due to high-speed nature of actions that are performed [150, 151, 152]. Furthermore, the accuracy demands of the model are very high to provide measurable feedback to athletes in real time. These problems often cause many markerless motion captures to provide inferior results [153, 154]. Although the use of wearable sensors such as accelerometers or optical markers can provide more accurate estimates, it often disturbs the training and provides poor results in outdoor setup. Thus, athletes tend to dislike the wearable sensors during training.

5.1 Introduction

The goal of the current chapter is to create robust deep learning model for diving athlete's pose estimation. The thesis focuses on 2D/3D human pose estimation in particular to do performance analysis of diving sports athletes. The athletes with individual trainers, who guide their performance improvement without leading to any injuries during training. Thus, we make the following key contributions:

- We release the DSV diving dataset with action localization demarcation and its corresponding 2D pose keypoints for research community. The dataset can be downloaded from `https://divenet.kl.dfki.de/` .

- We present a highly robust 2D and 3D pose estimation models trained towards estimating diving athlete's pose parameters in the wild using the diving dataset.
- Real-time estimation of 2D and 3D human pose estimation using compressed deep neural network model on the mobile platform.

The goal of diving pose parameter extraction from monocular RGB images can be summarized as shown in the Figure 5.1.

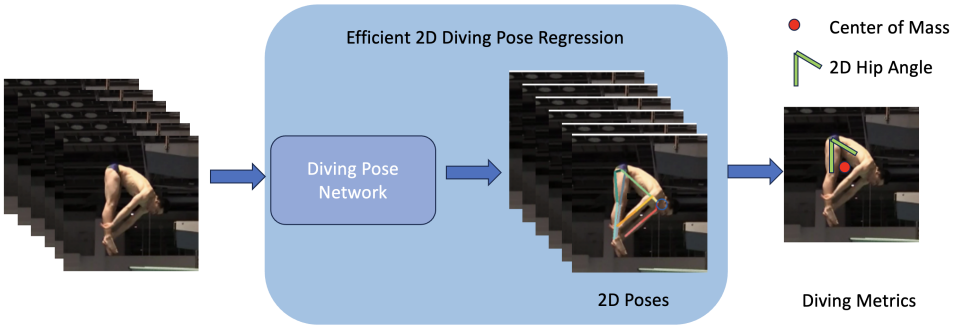


Figure 5.1.: Illustration of the problem statement: Extracting Diving Pose Metrics using Action Localization and Diving Pose extraction from RGB images capturing Diving sports motion.

5.2 Related Works

There has been a tremendous success in estimating human skeleton pose in images using deep learning methods. The works based on stacked hour glass networks [155, 156, 157, 158, 156] achieved remarkable performance on different datasets. OpenPose [61] employs Part Affinity Fields (PAF) to support bottom-up estimation. The authors of DeepHR Net [156] exploits multi-scale high-resolution networks to improve feature representation. The UniPose model [159] based on Waterfall Atrous Spatial Pooling architecture, that achieves state-of-art-results on several pose estimation metrics. The recent VitPose model [160] explore the potential of plain and non-hierarchical vision transformers [161] and provides a simple yet effective vision transformer baseline for pose estimation tasks. Another interesting work based on transformer uses a pose regression network to map images to keypoint co-ordinates, without resorting to intermediate representations [162].

However,the human motion analysis in sports videos warrants a different approach due to the complex nature of the body poses involved. Based on self-supervised learning, the suggested method in [163] employs pseudo labels as a self-supervised training strategy, together with a pseudo label filtering method, in the disciplines of triple and long jump. The study, described in [164], lead to the dataset *Kanoya*, which is composed of numerous series of videos of gymnasts doing acrobatic motions and proposes several refining approaches

to train OpenPose [61] for extreme human poses in sports. The method offered by [165] introduces Video Pose Distillation (VPD), a weakly supervised technique for learning features for novel video domains, such as individual sports that challenge pose estimation and shows the benefits of VPD on four varied sports video datasets: diving, floor exercises, tennis and figure skating, with fine grained action labels. The recent works of [166] combines the use of computer vision algorithms and fully convolutional neural networks. The proposed marker-less 2D swimmer pose estimation approach estimate the pose of a swimmer during exercise while guaranteeing adequate measurement accuracy. While the work of [167] presents a procedure-aware technique for action quality assessment, trained via a new temporal segmentation attention module, it also constructs a new fine-grained dataset, termed FineDiving, created on various diving events with detailed annotations on action procedures. The work described in [168], provides an alternative unsupervised representation learning technique based on videos captured by a single RGB camera with a focus on diving sports. First, the model uses a spatial transformer network to identify the person across all frames and then encodes the subject into time-variant and time-invariant components. They sample with temporally close, distant and intermediate frames, given a reference frame. While these methods focuses on poses, our goal is not only to estimate pose but also extract relevant diving motion physical parameters from performance videos.

5.3 Diving Sports Datasets

In the next section, we elaborate on the diving sports datasets used. Due to limited availability of the publicly available sports datasets, we curated a new diving sport training DSV datasets. The datasets differ from public datasets which often have static frames for different sports often captured from sports broadcasting performance videos. Due to high speed body movements, the videos often pose a challenge of poor quality frames in video sequence resulted due to motion blur. Thus, we selective choose better quality frames from sports videos presented in two different datasets: IAT and Sportscap[154]. The details of each dataset are explained as follows:

DSV Dataset: We use a recorded DSV (Deutscher Schwimm-Verband) diving dataset consisting of 450 diving sequences covering 15 hours of video data, with each motion clip lasting approximately 2 minutes. The video differs in varying camera viewpoints and springboard heights. The dataset consists of diving actions at four different levels of diving height: 3m, 5m, 7.5m, 10 meters. The diving action within a single video sequence ranges from 2 to 5 seconds, making the localization problem much harder. The dataset annotations consist of two temporal boundary marker frames denoting the beginning and end of the diving action. The annotations also consist of 2D keypoints representing 16 joints. *Note, all the recorded videos are with a static camera with a frame rate of 60Hz and a resolution of 3840 $\times$ 2160 px (see Fig 5.2). Although the videos are short clips of 1-2 minutes but the localizing actual dive is in the range of 1-3 seconds as shown in the Figure 5.3. To the best of our knowledge, this is the first publicly available dataset with a static camera setup*

for diving sports, which enables us to compute homography to the diving plane via our newly proposed homography estimation method (presented in Chapter 7).



Figure 5.2.: Exemplar images from the DSV diving dataset depicting a person jumping over the springboard.

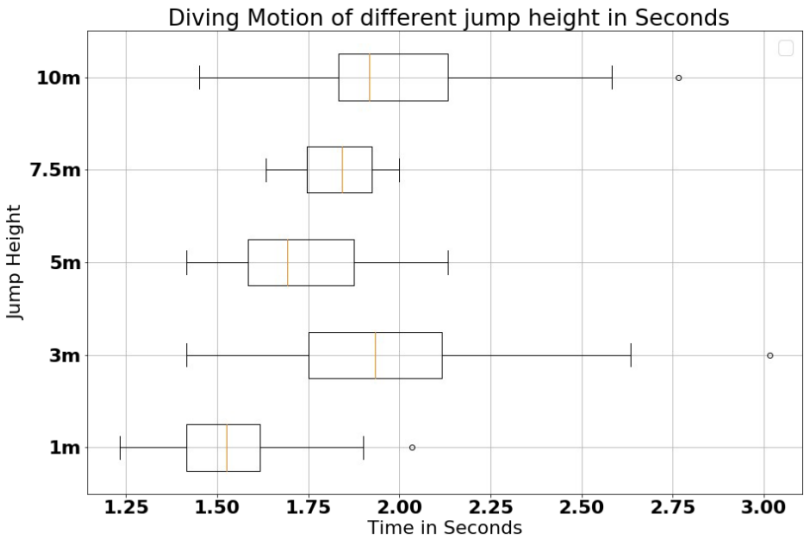


Figure 5.3.: The plot shows the range of the diving time for different diving heights in the DSV datasets.

IAT Dataset: (Institut für Angewandte Trainingswissenschaft (IAT)) dataset consists of 100 Olympic diving performance television broadcasting videos performed over 3 meter high springboard (see Fig 5.4). The video differs in varying camera viewpoints. The dataset annotations consist of two temporal markers to notify the beginning and end of the diving action. The dataset annotations consist of two temporal boundary marker frames denoting the beginning and end of the diving action. The dataset also consist of manually annotated 2D keypoints representing 16 joints. The videos involve a moving camera and is recorded with a frame rate of 25Hz and a resolution of 720 x 576 pixels.

Sportscap: The dataset includes per-frame action labels, manually annotated poses, and action evaluations from professional referees of various challenging sports video clips [154]. The dataset separates sports activity into two categories: competitive sports and daily



Figure 5.4.: Representative images from the IAT dataset showcasing frames from Olympic diving performance broadcasting videos

exercises. Competitive sports comprise balance beam, competitive diving, uneven bars, vault-women, hurdling, pole vault, and high jump, while daily exercises include boxing, keep-fit, and badminton. The dataset consists of 640 videos (110K frames) with 450,000 annotated skeletons of 25 joints and accompanying bounding boxes. We trained our model using only diving sports 2D pose data, totaling 23635 frames. Although, the frames in the dataset are annotated for different video sequence, the annotations are only available for intermittent frames. Hence, we use the dataset for training purposes only and do not use it for evaluations.

5.4 DiveNet Pose Regression Network

This section describes the pose regression module of the DiveNetPose architecture.

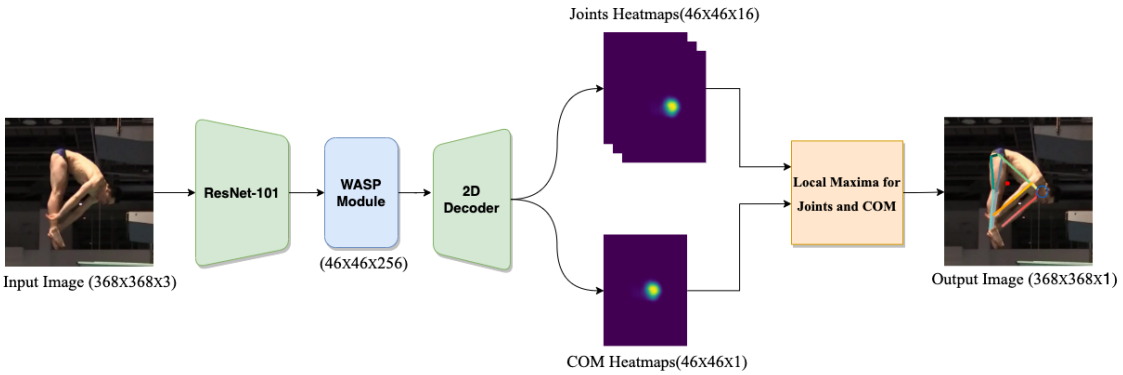


Figure 5.5.: The architecture of DiveNet pose regression model. The input image(HxW) is fed through the ResNet backbone and Waterfall Atrous Spatial Pyramid(WASP) module to obtain 256 feature channels at reduced resolution by a factor of 8. The decoder module generates 17 heatmaps, one per 16 joints and one for COM, followed by a local max operation which produces 16 keypoints and 1 COM (as shown in output image with a red circle)

The Figure 5.5 shows the DiveNet Pose regression network architecture which uses a ResNet-101 backbone feature extractor network. A Waterfall Atrous Spatial Pyramid (WASP) module inspired by [159] replaces the final layers. The decoder receives 256 low level feature maps from the first block of the ResNet backbone and 256 feature maps from WASP. The feature maps are aggregated and processed through convolutional layers, dropout layers ($p = 0.5$), and a final bilinear interpolation to scale to the original input size after a max pooling operation to match the dimensions of the inputs. We change the final decoder layers to generate $K + 1$ heatmaps where K is the number joints along with single heatmaps for the Center Of Mass (COM). The probability distributions were obtained by applying a softmax function. We train DiveNet pose regression network to regress 2D diving pose along with COM. As a criterion, the mean squared error (squared L2 norm) was used between ground the true heatmaps (H_{x_i}) and predicted heatmaps (H_{y_i}),

$$Loss_{mse} = \frac{1}{n}\sum_{i=1}^{n}(H_{x_i} - H_{y_i})^2, \tag{5.1}$$

where, $H_{x_i} = [h_1, h_2, .., h_{16}, cx_i]$ and $H_{y_i} = [h_1, h_2, .., h_{16}, cy_i]$. Here, $h_1, h_2, .., h_{16}$ are the heatmaps of joints and cx_i and cy_i are heatmaps of the prediction and ground truth COM, respectively.

Training

We train the model on DSV, IAT and Sportscap [154] dataset. We train the DiveNet-pose regression model for 50 epochs. Adam optimizer was used with 0.0001 as learning rate. The model accuracy stabilized at around 17 epochs with batch size of 8.

5.5 Evaluation Results

	DSV Dataset					IAT
Model	**3m**	**5m**	**7.5m**	**10m**	**Mean**	**3m**
SHG-8 [155]	56.44	44.48	31.88	31.69	41.12	58.81
Openpose [61]	35.81	17.66	14.04	10.18	19.42	28.49
DeepHR-Net [156]	34.69	5.165	2.25	1.44	10.88	13.87
UniPose [159]	61.5	49.87	40.56	48.125	50.01	61.30
Unipose-FT	77.98	**74.91**	70.28	68.05	72.80	**70.43**
DiveNet	**78.46**	74.37	**72.04**	**70.28**	**73.78**	67.36

Table 5.1.: The PCK@0.25 accuracy metric showing the accuracy of pose regression models on DSV and IAT test dataset.

We evaluate our 2D pose regressor in comparison to state-of-the-art 2D human pose regression models on diving-sports datasets. We use Percentage of Correct Keypoints

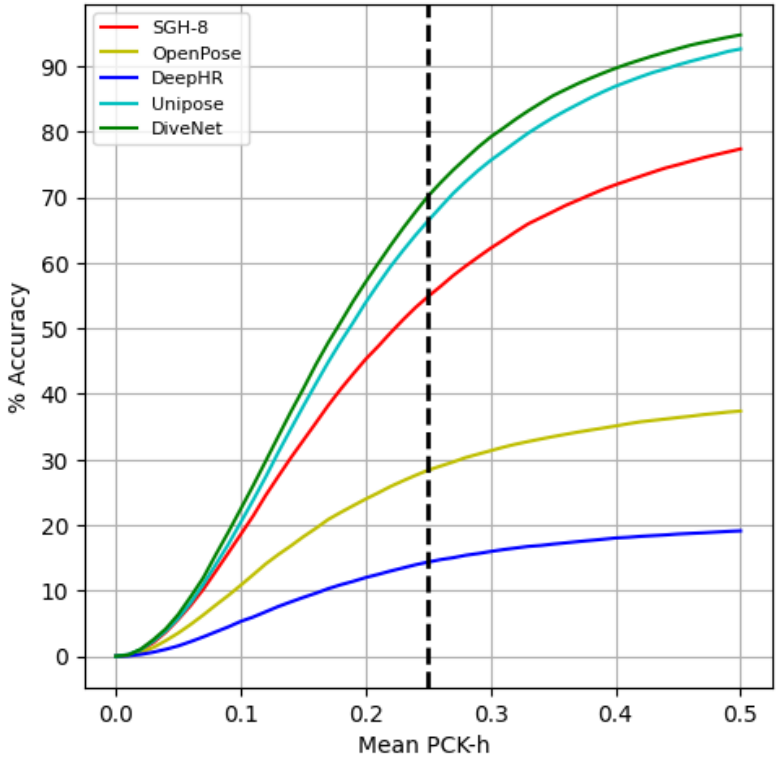


Figure 5.6.: The PCK-h plot compares the accuracy of different models at different PCK-h threshold.

normalized using head-length (PCKh). The predicted key point is assumed correct when it lies within a certain normalized threshold distance of the head length. For example, PCKh@0.25 metric is the percentage of correct keypoints at threshold of 0.25% times of the head length. We observed empirically that the threshold of 0.25 represents the joint area in pixels in the test set images. Hence, we use 0.25 as our threshold in the PCK-h metric. The Figure 5.6 shows that the change in accuracy for all models at different threshold levels. The pose regression accuracy scores are measured in PCK and can be found in Table 5.1. We found that Unipose performs best in all the state-of-the-art models without any training on a diving dataset. Hence, we further finetuned the unipose model (named Unipose-FT) on the diving dataset and obtained an accuracy of 72.80% on DSV and 70.43% on IAT dataset. DiveNet achieved a slightly higher PCK@0.25 accuracy of 73.78% on DSV dataset but a slightly lower accuracy of 67.36% on the IAT dataset.

Note that beside the competitive performance of DiveNet regarding pose estimation, the approach provides increased accuracy compared to the other approaches in estimating physical parameters, as discussed in section 7.6

Figure 5.7 depicts scenarios where DiveNet outperforms other models in terms of generating more realistic and precise joint locations. We can observe in most cases that the models were unable to correctly identify the upper body, resulting in a poor head position. In first row, both SGH-8 and OpenPose predicted the left arm incorrectly. The left elbow was mistaken for the left knee by DeepHRNet. OpenPose and DeepHRNet failed

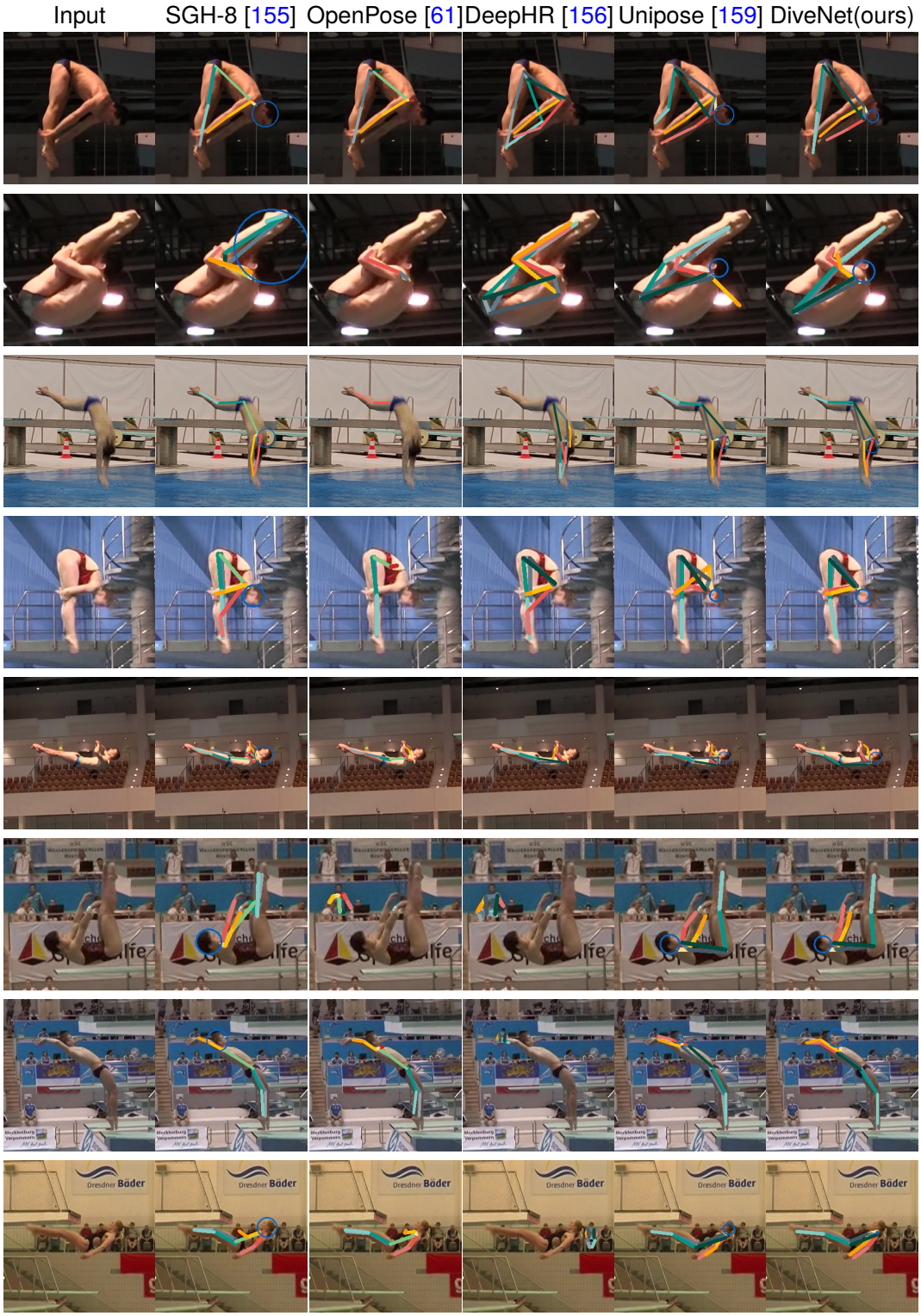


Figure 5.7.: The figures demonstrate the 2D posture estimate results of various models on the DSV Diving set. Divenet shows superior pose regression results on challenging diving images.



Figure 5.8.: The figures demonstrate the predictions of various models on exceedingly challenging scenarios in the DSV and IAT Diving sets. The last column shows the current limitations of DiveNet's predictions on challenging scenarios.

completely for the head. Due to its inability to detect both ankles, Unipose gave lesser lengths for the legs. While DiveNet was able to achieve better outcomes for the hands, legs, and head.

The images in the second row have background glare from lights, making the scenario more difficult.The SGH-8 incorrectly predicted hip and shoulder as left foot, resulting in incorrect head prediction. Only the left hand could be reliably predicted by OpenPose, which otherwise failed miserably. Right hand and right leg were combined with left leg via DeepHRNet. Unipose confused right hand with background. While it is clear that DiveNet offers a better answer in this case.

The third row shows an example with a blurred image of a diver about to complete a dive. Due to the blurry image in the hip area, none of the models were able to accurately estimate the left hip joint. SHG-8 was unable to detect the left leg due to occlusion. OpenPose failed miserably and predicted the right leg as the left hand. DeepHRNet projected legs as hands because it believed the individual to be standing upright. Unipose outperforms other models in the hands, legs, and shoulders, while DiveNet outperforms Unipose in the whole skeleton, notably in the hands, hips, shoulders, and head.

Similar to the second row, the pose in the fourth row is challenging due to a high left body occlusion. This scenario involves a position in which the hands go around the legs,

causing SHG-8 and DeepHRNet to interpret the left hand as the right leg. OpenPose was unable to identify the upper body, and Unipose failed to predict the left elbow correctly. We can see that, despite the difficult circumstances, DiveNet's prediction was fairly accurate.

The fifth row shows that SHG-8 failed for the right leg and OpenPose failed for the left leg. Additionally, SHG-8 inference resulted in poor predictions for hands. DeepHRNet failed for the right hand, while Unipose failed for both hands and the right ankle. When compared to all other models, DiveNet learned better representation to provide more accurate hands, legs, and shoulder position.

The sixth, seventh, and eighth rows all include a situation with additional individuals seated in the background. We can observe in OpenPose and DeepHRNet in row six, and DeepHRNet in rows seven and eight, that they failed in this case, and anticipated a diver from the background. SHG-8 failed in rows six and eight for hip joints and right hand respectively. In rows seven and eight, OpenPose failed for the right hand. Unipose outperformed other models, although DiveNet's results were more precise and accurate, particularly for the hips, shoulders, and legs.

Figure 5.8 depicts the extreme conditions under which pose regression models failed for various body joints. This figure also depicts the limitation scenarios where DiveNet predictions were less accurate. The first row shows the complex pose which are difficult to estimate. The SHG-8 and DeepHRNet messed up the hands and legs completely. As a result, SHG-8 misjudged the position of the shoulder, mistaking the hip for the position of the head. Only the left knee and ankle were predicted by OpenPose, and they were confused with the right ones. Unipose mistook the left hand for the right one when making its prediction. DiveNet offers a more effective solution, although it fails to accurately predict legs. Due to the background, none of the models were able to estimate the hands correctly in the second or third row. DiveNet's predictions for the second row may be seen with some offsets for the hands and right side of the body. In the third row, Unipose and DiveNet incorrectly identified the left leg as the right and struggled to identify the left hand. The last row illustrates the situation where the upper body area around the face is indistinct and has background noise. The skin tone seems to hide the hands and face, making them hardly apparent. We can see that DeepHRNet was ineffective in predicting any joint positions. All of the models, including DiveNet, failed to predict hands.

5.6 Efficient Architectures for Diving Sports Pose Estimation

The primary objective of our research is to develop a robust pose parameter extraction system tailored for trainers to analyze dives. Moreover, we envision the deployment of these models on hand-held devices in real-world scenarios, providing instantaneous feedback to athletes. This necessitates the exploration of efficient deep learning architectures that can seamlessly operate in dynamic environments.

To address this objective, the current section delves into the examination of two efficient architectures for diving sports pose estimation: BlazePose [169] and MobileHumanPose [170]. Our investigation encompasses the use of BlazePose for 2D pose estimation and MobileHumanPose for 3D pose estimation. These models are selected based on their efficiency and suitability for real-time deployment on mobile devices. The training and evaluation are conducted on diverse datasets, including MuCo-3DhP [20], MPII, MSCOCO [171], DSV, and IAT datasets. Note: The aim of efficient architectures experiments is to investigate the feasibility of mobile deployment. Thus, *we restrict the learning of diving motions to only 3 meters in order to simplify the input pose learning complexity and present images with diving person at a constant person scale.*

5.6.1 BlazePose

BlazePose, a knowledge-distillation-based model, adopts a unique approach in its architecture. Initially, a heatmap branch is trained, and the features learned from this branch are subsequently fine-tuned to directly regress 2D keypoints. The model employs a lightweight convolutional neural network architecture, enabling real-time inference on mobile devices.

BlazePose framework utilizes a two-step detector-tracker machine learning pipeline. The detector locates the region of interest (ROI) within the image, and the tracker predicts 2D pose keypoints within this region. The BlazePose model comprises two main components: the heatmap network and the regression network. The heatmap network is initially trained, and its embeddings are used to supervise the subsequent training of the regression network, which outputs 2D keypoints and their visibility. The architecture is depicted in Figure 5.9

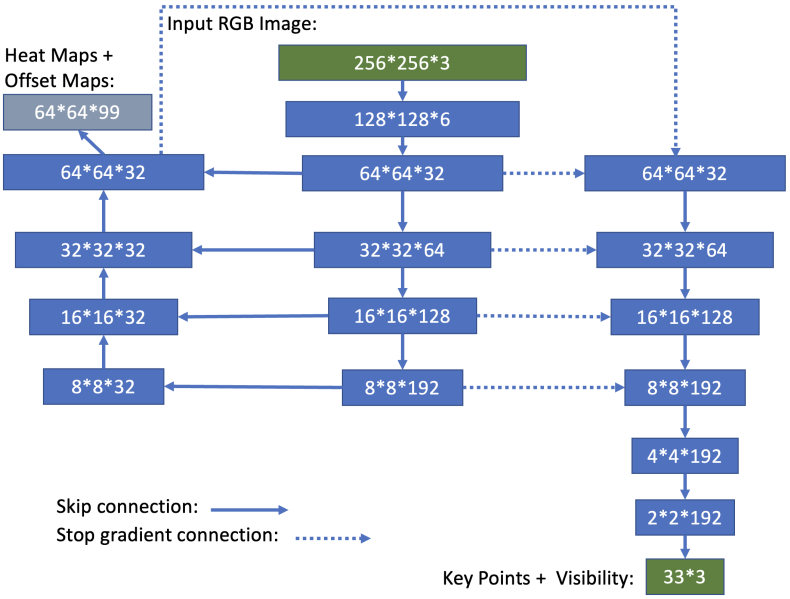


Figure 5.9.: Our representation of BlazePose model architecture [169]

5.6.2 MobileHumanPose

The MobileHumanPose model [170] addresses the often overlooked challenge of efficiency in 3D human pose estimation methods, which tend to prioritize accuracy at the expense of high computational costs. The authors present their contribution, MobileHumanPose, as an efficient model for real-time 3D human pose estimation. This model incorporates modifications from MobileNetV2 [172], skip concatenation inspired by U-Net [173], and a parametric activation function. Notably, MobileHumanPose outperforms similar alternatives with restricted computational power (less than 10 GFLOPS). Furthermore, the work emphasizes practical application by developing a real-world mobile application based on 3D human pose estimation. The proposed model successfully addresses the limitations of 3D pose estimation on low computational power devices, achieving comparable performance to state-of-the-art models while significantly reducing model size (seven times smaller than a ResNet-50 based model). Additionally, the model demonstrates a noteworthy reduction in inference time, making it well-suited for real-time 3D human pose estimation in mobile applications, particularly on devices like the Galaxy S20 CPU with a reduced inference time of 12.2ms. For further insights into the architecture of the MobileHumanPose model, interested readers are encouraged to refer appendix section A.2.

Implementation

To ensure real-time inference on the Diving Sports dataset, we implement both efficient architecture. Training involves using publicly available datasets such as MPII and MSCOCO, along with a custom-recorded DSV diving dataset and IAT Dataset. The model training for blazepose was performed using the TensorFlow framework [174] and MobileHumanPose was implemented using the PyTorch framework [175].

The model is trained on a diverse set of datasets, including DSV Diving, IAT, MPII [176], MuCo-3DHP [20](only for MobileHumanPose), and MSCOCO [101]. The data split remains consistent at 80% and 20% for training and testing frames respectively. A batch size of 64 frames is selected during for both training and testing phases. The model undergoes training for 200 epochs to ensure comprehensive learning.

The training process of BlazePose model is bifurcated into two phases. Initially, the heatmap tower is trained, followed by the subsequent training of the regression tower. During the training of the heatmap tower, the Binary Cross Entropy loss function is employed. For the regression tower, the Mean Squared Error loss function is utilized. Adam optimizer is used during both phases [177].

Similarly, the MobileHumanPose model is trained using Mean Squared Error loss to estimate 3D Pose. To facilitate real-world applications, we convert the trained model to the Open Neural Network Exchange (ONNX) [178] format and subsequently transform it into a TensorFlow Lite model. The final evaluation is conducted on Android or iOS mobile platforms.

5.6.3 Model Evaluation

We conducted training and evaluation on three models: BlazePose fine-tuned (FT), MobileHumanPose fine-tuned (FT), and MobileHumanPose full. The fine-tuned models are trained on specific datasets, whereas the full model training involves a combination of datasets, including MPII, MSCOCO, MuCo-3DHP, IAT, and DSV.

Model	Head	Sho.	Elbow	Wrist	Hip	Knee	Ankle	Mean
BlazePose FT [169]	95.49	95.99	93.64	92.24	96.69	96.37	95.83	95.29
MobileHumanPose FT [170]	88.19	86.5	78.6	76.56	87.23	88.19	87.57	84.77
MobileHumanPose-All[170]	86.76	84.69	75.59	73.63	87.17	88.73	88.07	83.09

Table 5.2.: PCKH metric applied to the DSV diving dataset.

Model	Head	Sho.	Elbow	Wrist	Hip	Knee	Ankle	Mean
BlazePose [169] FT	91.96	93.40	94.29	90.94	93.29	96.15	92.17	93.33
MobileHumanPose-FT [170]	99.32	99.55	99.03	97.56	98.97	99.87	98.26	98.13
MobileHumanPose-All [170]	96.88	97.61	94.34	88.69	97.46	99.16	92.80	95.27

Table 5.3.: PCK-h metric applied to the IAT diving dataset.

The Tables 5.2 and 5.3 present the PCKH metric results applied to the DSV and IAT diving test datasets, respectively. Notably, the full model exhibits a slight decrease in accuracy compared to the fine-tuned models. This reduction is attributed towards mitigating overfitting, fostering enhanced generalization across diverse background by combining different datasets.

The evaluation also reveals that the accuracy of the IAT dataset surpasses that of the DSV dataset. We attribute the lower accuracy in the DSV dataset to more variations in human motion on the diving board, often occluding the person and leading to inaccurate pose estimations. In particular, the DSV dataset often depicts the diving spring board at a five different height (1m, 3m, 5m, 7.5m and 10m), which contrasts with the constant position observed in the IAT dataset $(3m)$.

The 3D evaluation of the trained MobileHumanPose models on DSV dataset can not be performed due to lack of 3D pose ground truth data. Nevertheless, we evaluate the trained model on the MuCO-3DHP dataset. The 3D evaluation is necessary to verify if the model overfits to the diving sports data only. The Table 5.4 presents the quantitative results, indicating that MobileHumanPose exhibits slightly lower accuracy compared to the state-of-the-art model by Moon et al. [179] on the MuPots-3D dataset [20]. The lower accuracy can be attributed to the smaller size of the MobileHumanPose model relative to the Moon et al. model. Moon et al.'s model has a size of 133.90 MB and 34.34 million parameters, whereas MobileHumanPose is 14 MB with 3.5 million parameters. The choice of MobileHumanPose is driven by its compact size and fewer parameters, making it suitable for real-time 3D human pose estimation on mobile devices with a reasonable

trade-off in accuracy. Nevertheless, this needs further investigation by benchmarking MobileHumanPose model without any diving data as part of future work.

Model	3DPCK
Moon et al. [179]	81.8
MobileHumanPose [170]	70.17

Table 5.4.: 3D evaluation of MobileHumanPose [170] and Moon et al. model [179] on MuPots-3D dataset.

5.6.4 Ablation Studies

In this section, we conduct systematic ablation studies to explore the impact of model compression on both the accuracy and inference time of the MobileHumanPose model. Ablation experiments involve selectively reducing the number of weights in the model, allowing us to analyze the trade-offs between model size reduction and accuracy. This comprehensive investigation contributes valuable insights into the optimal threshold for weight parameters, offering a nuanced understanding of how different compression levels affect the neural network.

Comparing Performance Gains with Fine-tuning: To assess the performance of the MobileHumanPose model, we compare its pre-trained version, provided by the authors [170], with our fine-tuned model. The Figure 5.10 illustrates the outputs of both models. Notably, the pre-trained model fails to accurately estimate human pose for the DSV dataset image, while our fine-tuned model demonstrates precise estimations of 2D and 3D human pose. This clearly shows lack of training samples for diving poses in the pre-trained version of the model.

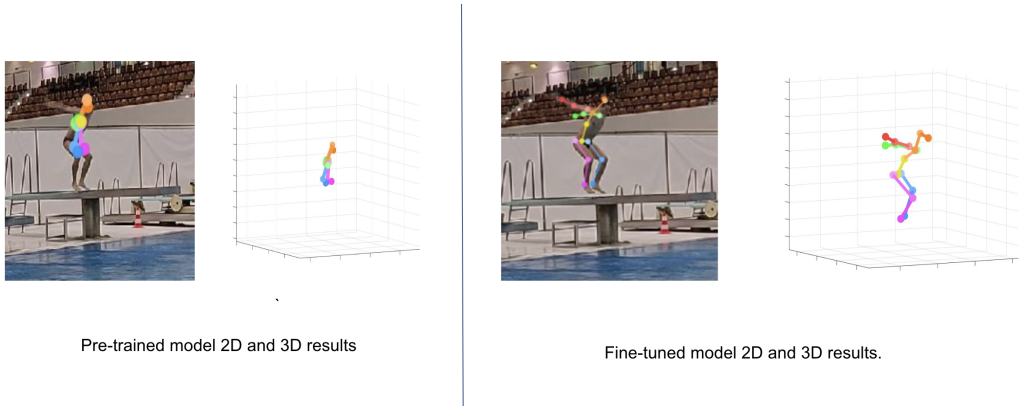
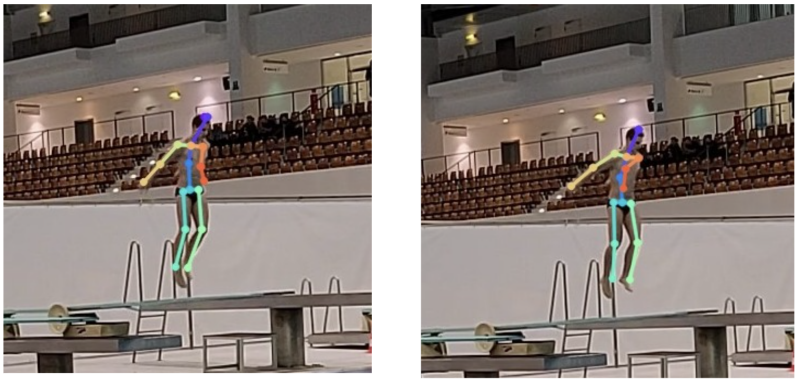


Figure 5.10.: MobileHumanPose model results for the pre-trained and fine-tuned models.

Model Compression: We perform model compression on the MobileHumanPose model by systematically reducing the number of parameters and evaluating accuracy at various compression levels. We use the PyTorch's post training static compression technique to compress model and improve inference speed on mobile devices [180]. As depicted in

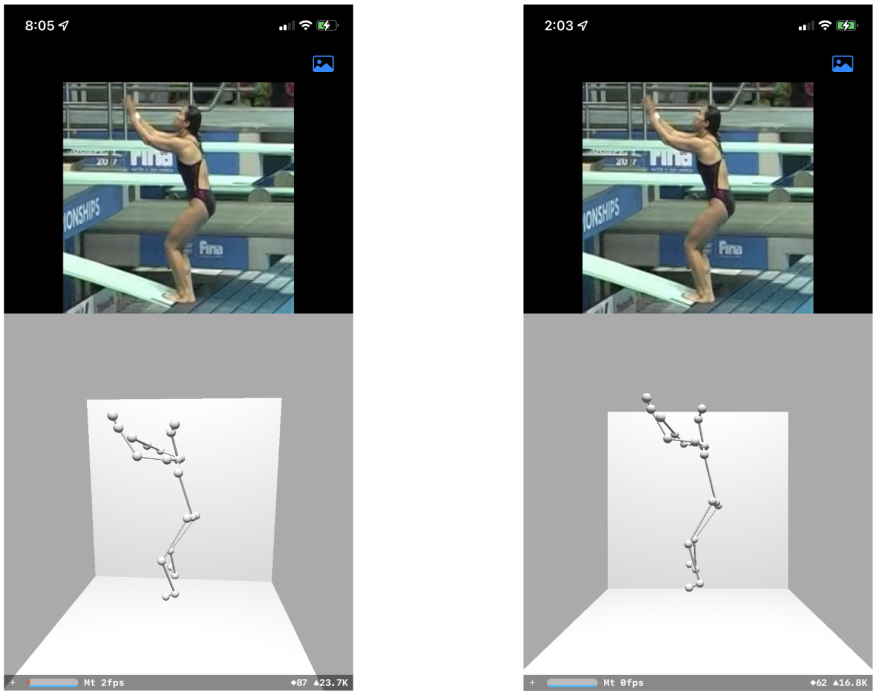
Table 5.5, accuracy decreases with increasing compression rates. Notably, a marginal accuracy drop is observed until a 30% reduction of parameters.

Surprisingly, the model at a compression level of 20% slightly outperforms the original model's accuracy. We attribute this improvement to the reduction of parameters acting as regularization, mitigating overfitting to the dataset. Therefore, the model at a compression level of 20% is chosen for further experiments.



Before compression After compression

Figure 5.11.: Effect of quantization and compression on the 2D human pose output.



Before compression After compression

Figure 5.12.: Effect of quantization and compression on the 3D human pose estimation output.

The quantitative evaluation results are shown in Table 5.5, along with qualitative evaluation show the output pose images as shown in Figures 5.11 and 5.12. The results

	Compression level				
	[100%]	**[90%]**	**[80%]**	**[70%]**	**[60%]**
Head	93.84	93.76	93.73	92.92	63.50
Shoulder	91.07	91.11	91.09	89.70	66.71
Elbow	87.66	87.34	87.85	86.04	52.14
Wrist	83.23	82.95	83.34	82.03	46.54
Hip	93.01	92.89	93.12	91.65	62.24
Knee	91.28	91.01	91.51	90.55	62.29
Ankle	92.69	92.72	92.72	90.80	63.07
Mean PCKh	90.31	90.17	90.39	89.07	60.44
Model Parameters	3524736	3178474	2832211	2485949	2139686

Table 5.5.: The joints accuracy at PCK-h@0.5 for different compression models.

demonstrate that pruning and quantization yield a smaller model and improved inference time with negligible loss in accuracy. The goal is to have a model that can estimate 2D and 3D human pose in real-time, prioritizing a smaller and faster model over absolute accuracy. As seen in Figure 5.11, there is very little difference between the output of the full model and the model pruned at 20%.

Inference Time: We experiment on the MobileHumanPose model to investigate the reduction in inference time resulting from a decrease in the number of parameters, pruning, and quantization. The full model, a pruned model at 20%, and quantization were considered. The full model took 142 milliseconds for 3D pose estimation, while the pruned model at 20% took 130 milliseconds, resulting in a reduction of 12 milliseconds per image. Additionally, the pruned model reduced in size from 13 megabytes to 11 megabytes. Although these optimizations had a limited impact on the MobileHumanPose model due to its small size, they present an opportunity for application on larger models, potentially facilitating integration into mobile devices. The optimization techniques were not applied to the BlazePose model, given its smaller size, where the reduction in inference time would be marginal.

5.7 Conclusion

In this study, we presented a dive pose regression model that surpasses the performance of all 2D pose regression models when evaluated on a diving dataset. Demonstrating robustness across various diving heights, the trained pose regressor consistently provided accurate poses even in challenging scenarios. Additionally, we address the need for efficient models suitable for mobile platforms in the context of sports involving complex poses. Our approach involves retraining the existing BlazePose model, resulting in competitive results with a 95% PCK-h on 2D pose accuracy on 3 meter height dives.

Diving Action Localization

6

> *"Don't practice until you get it right. Practice until you can't get it wrong."*
>
> — **Unknown**

The quote's essence resonates and reinforces the idea that professionals persist in their efforts until they reach a point where they perform devoid of errors, and precision becomes second nature. The continuous refinement ensures that coaches can rely on precise and accurate insights into athletes' movements. This, in turn, aligns with the idea of practicing until one can't get it wrong—striving for a level of accuracy that enhances coaching effectiveness.

In the dynamic realm of sports, where moments of triumph and athleticism unfold in the blink of an eye, the ability to capture and analyze the precise location of key actions has become paramount. The sports action analysis often involves identifying, tracking, and understanding the spatial dynamics of athletes during live events. This provides coaches with valuable insights, and advancing sports science research [41].

Sports Action Localization leverages sophisticated models, and advanced sensors, to discern and interpret the movements of athletes in real-time [181]. By meticulously dissecting the temporal motion of pivotal moments within a sporting event, the field opens new avenues for in-depth performance analysis, strategy optimization, and the creation of immersive sports content [182]. The demand for sports action localization has been fueled by the ever-growing appetite for data-driven insights in the sports industry. As fans crave a more engaging and interactive viewing experience, broadcasters and content creators seek innovative ways to deliver real-time information about the on-field action [183].

Moreover, coaches and sports scientists are increasingly relying on accurate sports analytic data to refine training programs, make informed tactical decisions, and gain a competitive edge. Also, the accurate action localization serves as a force multiplier for coaches, enabling them to monitor multiple athletes simultaneously. This efficiency allows for more effective use of coaching resources, especially in team sports, where individualized attention can be challenging. Further, it enables the coaching in real-time with instantaneous feedback during training sessions. Athletes can receive immediate insights into their performance, enabling them to make on-the-fly adjustments and corrections. This iterative process accelerates the learning curve and promotes faster skill development [184].

In the next section, we describe the overview of the action localization methodology used for diving sports training.

6.1 Overview

Our work primarily focuses on competitive springboard diving training. The figure The process begins with a video feed as the input. This could be a recorded video or a live stream capturing a dynamic scene with various activities or actions taking place. Each frame represents a still image from the dive training video sequence. The action localization task involves associating a precise temporal localization of an action by demarcating the start frame and the end frame representing diving action. To achieve this goal, we present a new low-latency solution based on Temporal Convolutional Networks (TCN) for localizing diving actions in untrimmed videos. Before delving into our methodology, we review existing approaches that address action localization in the subsequent section.

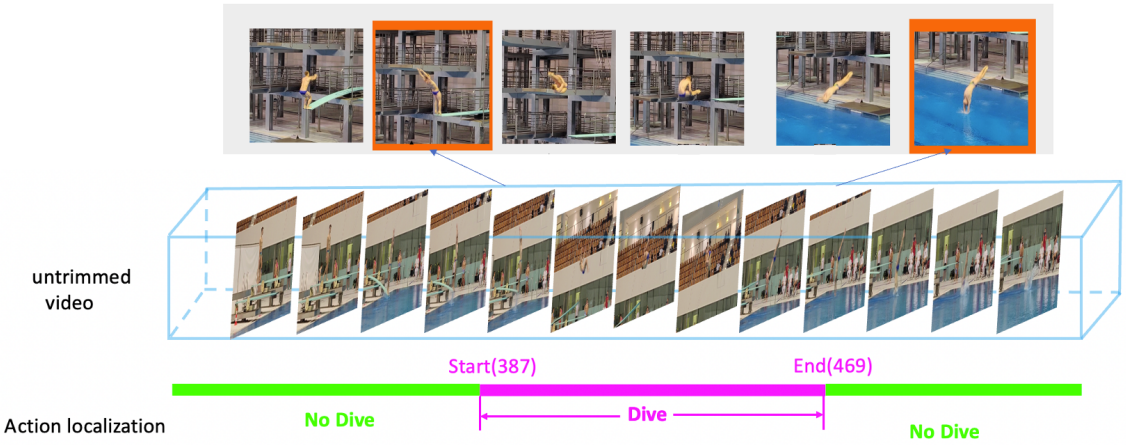


Figure 6.1.: Illustration of diving action localization in an untrimmed video.

6.2 Related Works

There is a surge in sports video analysis in recent years due to the new advances in deep learning methods and the availability of high computing resources. A common theme for learning complex temporal relationships between the actions in long untrimmed videos is to encode videos in an end-to-end manner using pre-trained 2D or 3D CNNs [141, 185, 186]. Sequence-to-sequence models use these learned high-level features to localize the action.

Recurrent Neural Networks Models:

The Recurrent Neural Network (RNN) are one of the most popular methods for modeling temporal information in sequences [187, 188, 189]. A Multi Long Short-Term Memory (LSTM) model further expands the temporal relation by using multiple input and output connections of an LSTM [190]. The work of [191] used a novel two-stream feedback network instead of a standard LSTM layer. The two-stream consists of the upper stream, which focuses on the interpretation of the frames, and the lower stream that models the temporal relations. The Kanojia *et al.*[151] presented a novel attention guided LSTM architecture with an encoder and decoder network for localizing actions in diving motion. The attention network utilizes the global context from the encoder and learns to focus on the diver during the dive without any such supervision. However, apart from being hard to correctly train, recurrent neural networks only capture implicit relationships between certain high motion actions [192].

Graph Neural Networks

The learning of contextual relationships of sports actions in videos can be modeled using graph-based methods. A *context graphs* represents the similarities among videos and the relationship between segments of a video [193]. The graph is learned using a context walk with a fixed number of steps and creating a conditional distribution over all super voxels. The Segment Action proposals are predicted using Conditional Random Fields. Thus, context graphs reduce the search space and avoid a need for a sliding window over a complete action. Farha *et al.*[193] learns context relationships and represents them by a graph for each video, where super-voxels form the nodes and directed edges capture the spatial relations between them. Graph based methods work by modeling similarities and relationships between segments of a video to be able to form a graph and cluster similar information making it easier to classify them. The Graph Convolutional Network (GCN) is also used to capture the spatial and temporal pattern in the data [194, 195]. A spatio-temporal graph convolution is applied on pose estimation from videos to generate a higher-level feature maps using a graph. These feature maps are then classified to corresponding action categories using a classifier with softmax outputs. The GCN can also be used to explicitly model appearance and motion similarities between video moments with a weakly supervised method [195]. A stacked version of GCN, Stacked Temporal GCN (ST-GCN) represents elements related to the actions such as actors, objects, etc. as nodes to better characterize the complex actions in videos [196]. The nodes are connected along the spatial and temporal dimensions instead of having body joints as nodes of the graph.

Action Proposal Networks

The action proposal networks typically use search strategies such as selective search sampling to produce sequence proposals [197]. The bounding boxes also known as *Tubelets* instead of super-pixels use super-voxels. Thereby eliminating the issue of linking boxes from one frame to another. Taking inspiration from R-CNN [198] and Faster R-CNN [199], Temporal Action Localization network (TAL-Net) exploits temporal context from the videos for both proposal generation and classification [200, 201]. Action proposal based networks are effective but require an exhaustive search of the complete video, making them computationally expensive and infeasible, especially in very long videos [200].

Temporal Convolution Networks

While RNN variants and graph-based models are hard to train [192], they also tend to capture only implicit relationships between specific motion actions and have a limited span of attention. On the other hand, Temporal Convolutional Network (TCN) can process long-range patterns due to the kernels sharing weight for all time steps. The work by Lea *et al.*[202] was the first to use TCN networks for action detection and segmentation. They presented two TCN variants. First, Encoder-Decoder TCN hierarchically modelled the actions using temporal convolutions, pooling and up-sampling. Second, Dilated TCN used a deep stack of dilated convolutions with skip connections. They showed that both TCN models outperformed their Bidirectional LSTM network [203] and were much faster to train. More recent variants of TCN for action detection are PDAN [204], MS-TCN [205] and Multi-tower TCN [206]. MS-TCN extend the work of Lea *et al.*[202] by stacking multiple dilated TCNs to form an ensemble a multi-stage networks. Each stage produces an initial prediction that is refined by the next one and by computing the loss after each stage. Multi-tower TCN also uses multiple TCN networks, but instead of forming an ensemble, the networks are stacked in parallel. The outputs from all the TCN towers (having different receptive fields) are fused before passing through the softmax layer. The intuition behind a multi-tower structure is to use multiple receptive fields to processes information at various temporal scales to deal with different event frequencies. One drawback of using convolutions is that they assign the same importance to each local feature in the kernel, preventing kernels from selecting the region of interest effectively. In an attempt to remedy this, Dai *et al.*[204] proposed the Pyramid Dilated Attention Network (PDAN) that employs dilated attention layers to allocate attention weights to local frames in the kernel. Inspired by the success and better performance of TCNs and its ability to learn short-term and long-term temporal relations over other sequence modeling methods [207], we develop a temporal convolution network for diving action localization in this work as discussed in the following section.

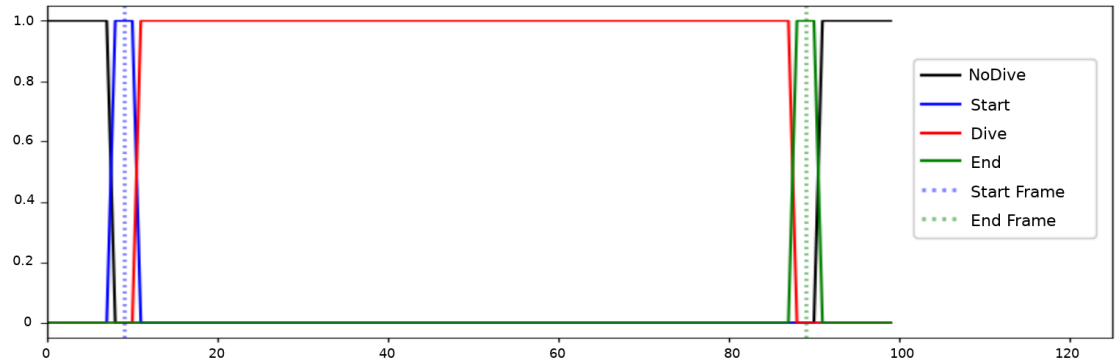


Figure 6.2.: Action Localization using class labels (NoDive, Start, Dive, End) for video frames. The dotted vertical blue line and green line represent the start frame and end frame of the diving action respectively.



(a) No Dive **(b)** Start **(c)** Dive **(d)** End

Figure 6.3.: Images from the recorded DSV diving dataset. Figure (a), (b), (c) and (d) shows examplar frames of a diving video representing the classes NoDive, Start, Dive, End, respectively.

6.3 Methodology

The diving sport involves complex body movements performed by an athlete in a short time (2-5 seconds) . We apply the task of temporal action localization for diving motion. First, the dive is demarcated by detecting the start frame and end frame of the motion as shown in Figure 6.1. The *start* frame is a representative frame where the force exerted by the athlete to the springboard is zero. The *end* frame of the motion is when the athlete touches the water. Next, we select three frame step function to represent the boundary classes for each diving motion. Additionally, we define two more classes, namely "*NoDive*" and "*Dive*". *NoDive* class comprise all the frames not representing the diving motion. These frames usually correspond to the diver standing on the board or entering the water. The *Dive* class represents all frames where the diver is in the air during the motion. The class representation for the diving motion classification is as shown in Figure 6.2. Examples of diver pose representing different classes in diving motion are shown in Figure 6.3.

The Figure 6.4 illustrates the overall architecture of the diving action localization framework. We first locate the person with a square bounding box. We crop the person repre-

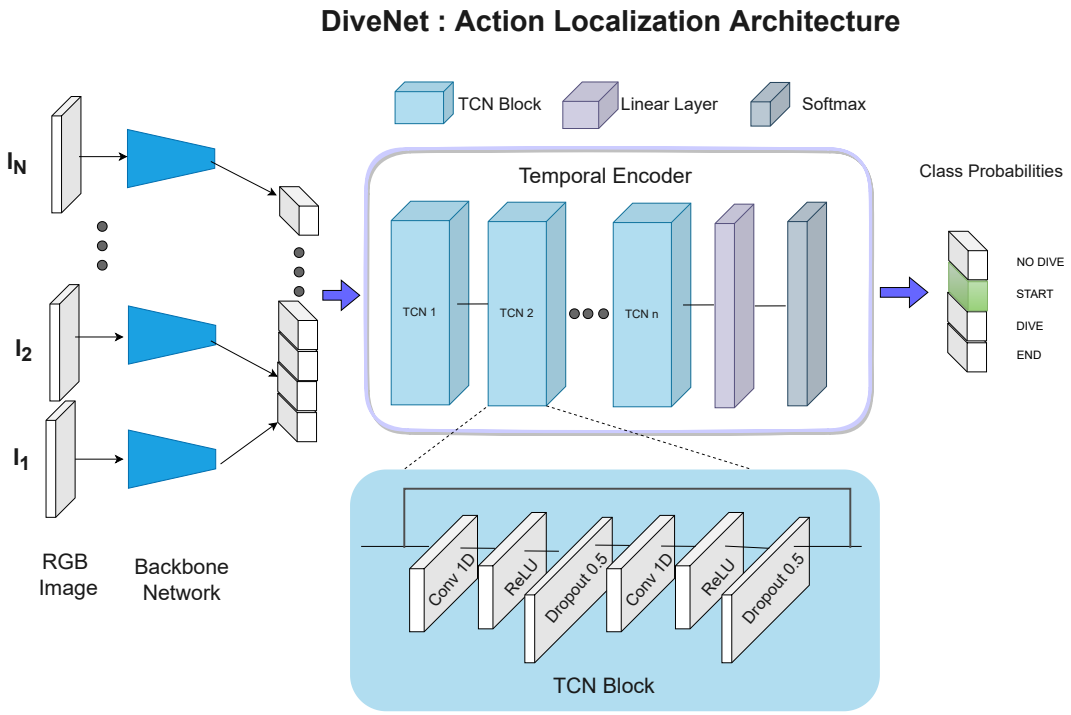


Figure 6.4.: Diving action localization - TCN block architecture

sented in the image and provide it as an input (I) to the Feature Extractor Network. The extracted features for each image are concatenated and fed to Temporal Convolutional Network (TCN) to get the final class scores. The input image is cropped with the detected bounding boxes containing the diver. The cropped images are passed through feature extractor backbone network, which extracts high-level features from the images. Given a video sequence $V = \sum_{t=1}^{N}\{I_t\}$ of length N, the feature extractor networks extracts features F_t for each cropped frame I_t. The N features are split into temporal windows of length T and used as input for the temporal encoder. The temporal encoder consists of multiple TCN blocks. The higher the number of TCN blocks leads to higher receptive field i.e. the network has larger context and can see farther in time. The output from the last TCN block is T features for each item of the sequence, which are concatenated and passed through multiple 1D convolutional layers to down sample the feature size to IR^{512}. Finally, the softmax function is applied over the outputs of linear layers. The linear layer outputs represent the confidence score for each class, and the softmax layer squashes and normalizes the score of each class to be between 0 and 1.

We use two different types of backbone feature extraction networks for action localization:

- ResNet: The ResNet-18 is used as a backbone feature extractor to segment diving frames in the specified classes. The feature vector of 512 represents a single frame as input to the classifier.

- Human Mesh Recovery: We use HMR model from [144] as a backbone feature extractor to achieve the motion segmentation. We use feature vector size of 2048 to represent the pose of the person as input.

The temporal encoder as shown in Figure 6.4 is inspired by [149] and comprises of 3 TCN blocks, each having a residual connection to the previous block. Each TCN block contains two sets of 1D convolutional layer, group normalization layers, ReLUs for non-linearity and dropout layers with a probability of 0.5 for regularization. The concatenated feature vectors from the backbone network are input to the temporal encoder. A sequence of length T $\{\phi_1, \phi_2, ..., \phi_T\}$, where ϕ_t is the feature vector of the t^{th} frame, is fed into the encoder. The output of the proposed network is a single vector $\hat{o}_t$ for each temporal window as shown in equation (6.2). The target vector y is one hot encoding, which has the class label one for positive classes and zero for negative classes. The weights vector w has value w_c for the respective c class. We set the weights vector w_c empirically to $[0.15, 0.35, 0.15, 0.35]$ (see section 6.6.4 for more details). We use a weighted cross entropy loss function as presented in equation, for training, where C is the number of output classes:

$$\mathcal{L} = -\sum_{i=1}^{C} w_i \cdot y_i \cdot log(\hat{o}_t)), \tag{6.1}$$

$$\hat{o}_t = softmax(f_{TCN}(\phi_1, \phi_2, ...\phi_T)). \tag{6.2}$$

6.4 Implementation Details

In this section, we explain the experimental setup and training procedure used to train the DiveNet model.

Experimental setup: The images used for training and evaluation are sampled uniformly from untrimmed diving videos at a rate of 60 frames per second. For training, we only use the frames involving diving motion. The images are cropped to size 224x224 around the bounding box containing the diver. The cropped images are converted into features on per frame basis using one of the feature extractors described in section 6.3.The model is trained with temporal sequences of length T=17. The frames of a diving sequence are split into temporal windows of length 17. These set of features are used as inputs to the temporal encoder network.

Training: We train two different models on DSV and IAT dataset. The 1D convolution layers have kernels size of 3 and padding size of 2. The dilation size is set to 1. We train all the networks for 50 epochs with a batch size of 16. Adam was used as an optimizer with 0.0001 learning rate. The accuracy stabilized at around 17 epochs with batch size of 8. The dropout rate is set to 0.5. We use three temporal blocks. The learning rate and weight

decay are set to 0.00001 and 0.001, respectively. The weights used for cross-entropy loss are [.15, .35, .15, .35]. The model is implemented in Python 3.6 using PyTorch on NVIDIA GeForce GTX 1080 GPU. We train DSV diving dataset with the following split: 300 sequences for training, 50 sequences for validation and 100 sequences for testing. The frames in each sequence belong to one of the four classes (No Dive, Start, Mid, End).

6.5 Evaluation Results

This section discusses the result of various experiments we performed to determine the optimal parameters for the action localization model for diving. An example of ground truth and predicted output probabilities is shown in Figure 6.5. For the given video sequence, the start and end markers are predicted within a single frame latency with respect to target temporal markers. The predicted pink and purple colored area shows the latency of start and end boundary markers respectively over the bold lines representing the ground truth.

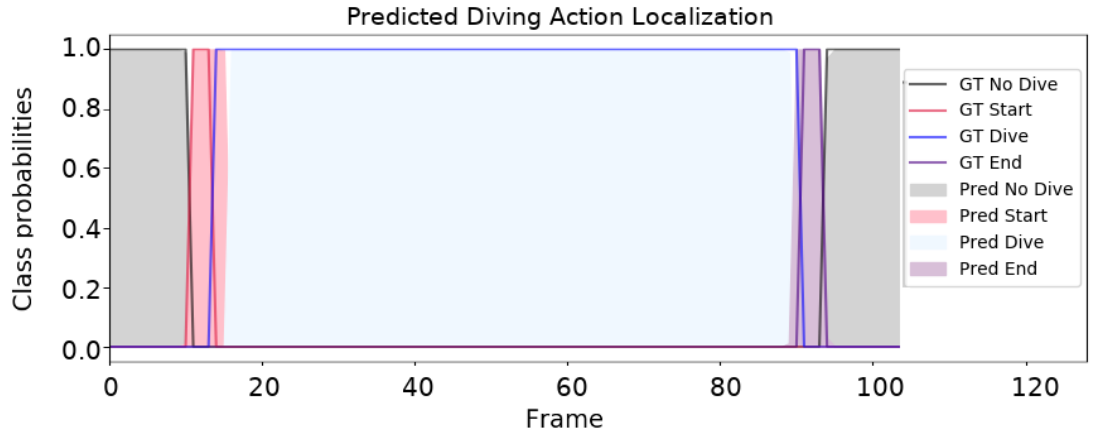


Figure 6.5.: The ground truth and predicted probability signals for the 4 class model (No Dive, Start, Dive, End). Bold lines represent the ground truth probability signal, and colored areas represent the predicted probabilities.

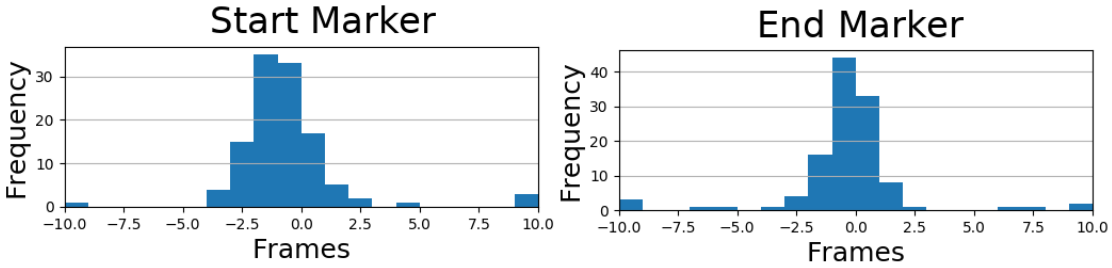


Figure 6.6.: Histogram of the latency measure of the estimated start and end boundary markers. The y-axis represents the number of sequences and x-axis represents the lag in frames. The negative values on x-axis represents prediction prior to the reference frame and positive values represents prediction after the reference frame.

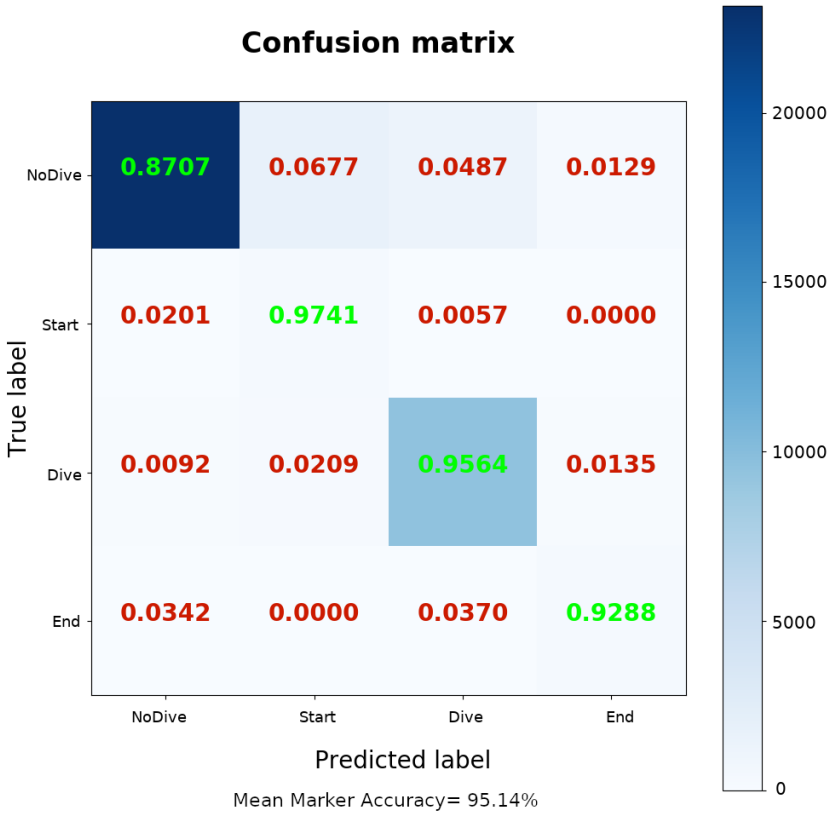


Figure 6.7.: The confusion matrix shows the model accuracy over test set. The model uses a HMR backbone network with 17 frames as input.

We evaluate our diving action localization model with two metrics: (i) classification accuracy of the frame and (ii) latency in the position of boundary markers.

6.5.1 Classification Accuracy

The mean classification accuracy is the ratio of the number of correctly classified frames to the total number of input frames representing boundary classes. Due to the presence of highly imbalanced classes (start and end class frames having very low representation), mean accuracy can be misleading and incorrect metric. To this end, we further analyze the performance of a model with a confusion matrix as shown in Figure 6.7. The confusion matrix clearly shows a low number of misclassification for all the classes, which is essential for accurate localization of the diving motion.

6.5.2 Localization Latency

We also evaluate the latency of the predicted boundary markers of the trained model. Latency gives additional information about how far are the start and end markers predicted from the ground truth in time. For example, given a sequence where the start marker is

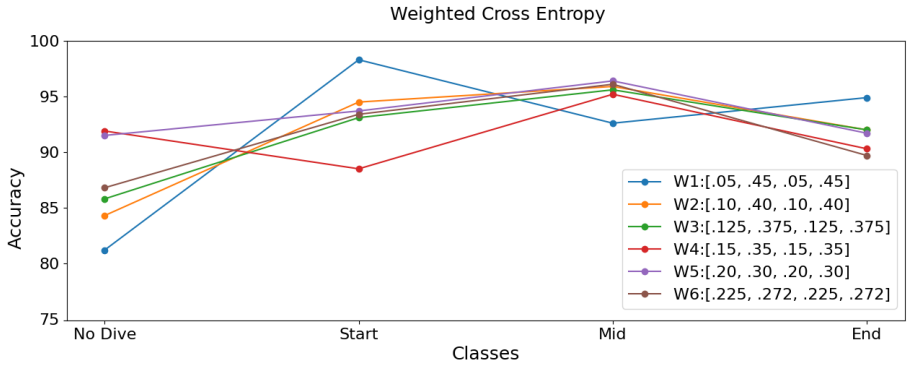


Figure 6.8.: Model accuracy over various individual classes using different class weighting configurations.

the i^{th} frame of the video, and the network predicts the j^{th} frame to be the start marker, the sequence is termed as accurately detection if the latency $|(j-i)|$ lies within a threshold limit of 1. We measure the percentage of test sequences that are within the low latency threshold limit. The higher the percentage over the test set, the better the predictions (see Figure 6.6).

6.6 Ablation Studies

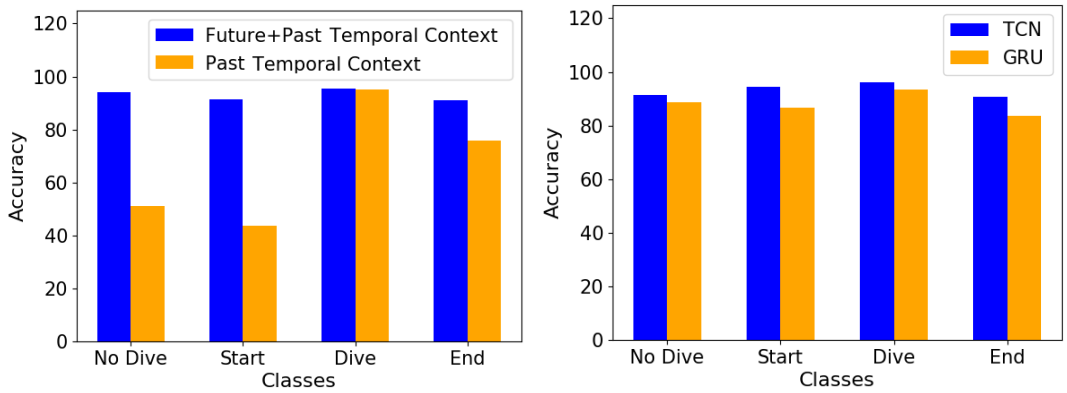
This section involves conducting ablation studies to explore the impact of balanced sampling in mitigating the adverse effects of imbalanced classes. Additionally, we experiment to determine the necessary temporal context required to achieve optimal results. The study also encompasses the influence of employing different backbone networks, the utilization of Gated Recurrent Units (GRU), and the exploration of various loss functions.

6.6.1 Importance of Balanced Sampling

The diving dataset consists of temporal diving motion boundary markers denoted as a step function consisting of three frames (the boundary frame and adjacent $+/-1$ frames). With such a representation, the data is highly imbalanced with 95$\%$ of frames belonging to class 'Dive' and 'NoDive'. The problem of imbalanced classes is, they lead to representation bias in classification results, leading to erroneous conclusions [152]. To this end, we use balanced sampling to sample an equal number of representatives for each class randomly at each epoch with weighted cross-entropy loss to improve the model performance.

6.6.2 Backbone Feature Extraction Network

We investigate two different backbone networks, as discussed in Section 6.3 for action localization task. We used ResNet-18 trained on Object classification task and HMR



(a) Impact on Accuracy of previous frames as input context vs bi-directional context for HMR-17 network.

(b) Accuracy results comparison of TCN as temporal encoder vs GRU [208] as temporal encoder with input size of 17 frames.

Figure 6.9.: Ablation study to analyze the required context in (a) and effectiveness of TCN with respect to GRU networks (b).

Network trained on a 3D Human Pose in the wild (refer Table 6.1). The backbone network ResNet-18 [141] network and HMR network are trained with the same temporal input frame configurations $\{5, 9, 13, 17, 21\}$. The 3D human pose based feature extractor HMR provides superior results, reaching above the overall accuracy of 90% . The pose based extractor performs high despite slightly lower accuracy on the end marker.

The ResNet-18 based feature extractor performs poorly and does not reach above 90% accuracy for all four classes in all scenarios. Also, even when it achieves 91.4% accuracy for the "Start Marker", it fails for the "NoDive" class due to misclassification, resulting in high latency for "Start Marker". Furthermore, our experiments with deeper networks of ResNet architectures reported no improvements or inferior results.

Backbone Network	Classes				Mean Accuracy
	No Dive	**Start**	**Dive**	**End**	
Resnet18_5	77.6	84	93.3	91.5	81.8
Resnet18_9	75.2	76.1	96.8	87.2	80.9
Resnet18_{13}	77.5	91.4	95.1	90.9	82.4
Resnet18_{17}	84.5	85.6	95.3	89.2	87.4
Resnet18_{21}	89.7	87.6	92.6	92.3	90.5
HMR_5	76.9	93.7	92.9	90.9	81.3
HMR_9	92.1	92	95.1	89.2	92.9
HMR_{13}	**95.2**	**93.4**	**95.2**	**89.7**	**95.1**
HMR_{17}	**91.5**	**94.5**	**96**	**90.6**	**92.7**
HMR_{21}	89.1	95.7	93.5	90.9	90.4

Table 6.1.: Quantitative results: Ablation study results of different type of feature extractor with different input length. For instance, HMR_5 is Human Mesh Recovery model with five frames as input. The metric is per frame classification accuracy, the higher the better. The top two best results are highlighted in bold.

6.6.3 Number of Input Frames

Table 6.1 presents experimental results with different input frames. The size of the temporal window fed to the temporal network as input affects the model's performance. We trained different models where N=$5, 9, 13, 17, 21$. The models with the input window sizes of $N = 5$ and $N = 9$ perform poorly. Thus, the model does not learn the necessary temporal embedding if the input window is too short. Hence, the model does not have adequate information to make accurate localizations.

On the other hand, too long sequences could also produce poor results, as the model is prone to overfit. With the larger temporal window of $N = 21$ frames, the model does not improve results. We found the input window size of $N = 13$ and $N = 17$ frames produced the best results. Therefore, we restrict further experiments and evaluation to the models with 13 and 17 frames input windows.

6.6.4 Weighted Cross Entropy Loss

Balanced sampling does mitigate the issue of highly unbalanced data, but it does not provide the best results. Therefore, we further experiment with weighted cross-entropy loss. Even though $[.20, .30, .20, .30]$ provides a stable accuracy for across the classes, its latency (Table 6.2) in predicting the marker position is poor, especially for the start marker (Figure 6.8). Thus, we found the weights of $[.15, .35, .15, .35]$ provide, empirically, the best results of over 90% in all cases. The other class weight configurations deteriorate the latency of the boundary markers, which implies that they do not assist the learning of start and end boundary classes and are not suitable for the prediction model.

Class Weights	Start Marker		End Marker	
	Acc	Lat$(+/-1)$	Acc	Lat$(+/-1)$
[.225, .272, .225, .272]	93.4	81	89.7	86.2
[.200, .300, .200, .300]	93.7	71.5	91.7	88.7
[.150, .350, .150, .350]	**94.5**	**90.5**	**90.6**	**87.9**
[.125, .375, .125, .375]	93.1	80.1	92	82.7
[.100, .400, .100, .400]	94.5	79.3	90.2	87.9
[.050, .450, .050, .450]	98.3	81.8	94.9	66.3

Table 6.2.: Accuracy(Acc) and Latency (Lat) scores for models with different weighted configurations used in weighted cross entropy loss during the training. The latency values represent the percentage of frames that have latency between $[-1, 1]$ in the respective columns.

6.6.5 Temporal Context

We experiment to analyze the importance of the bidirectional temporal context presented to the model. We trained two different models: (i) model with only past frames including

the boundary frame F_t as input $[F_{t-n}, F_t]$ (ii) model with equal amount of past and future frames relative to the boundary marker frame $[F_{t-n}, F_{t+n}]$. We found the model trained with only the past frames ($[F_{t-n}, F_t]$) as input to predict the class of the current frame F_t under-performs with the accuracy of 70 %. Whereas by showing the information in both directions, i.e. frames immediately before and subsequent frames to boundary markers, the model prediction shows improvement over 91 %, as shown in Figure 6.9a. Thus, it is clear that it is essential to have context from both directions to detect boundary classes accurately.

6.6.6 TCN vs GRU

We experimented with two different sequence modeling architectures to evaluate diving action localization. We trained GRU architecture inspired by [208] and trained with the same experimental setup as our TCN architecture. The temporal encoders process features for bi-directional frames to classify a sample. The plots in Figure 6.9b clearly shows the superiority of TCN model over GRU [208] model. The TCN motion encoder outperforms the GRU model on all classes.

6.7 Conclusion

We present a diving motion analysis framework, DiveNet to extract performance parameters. A generic action localization (or recognition) of daily activities often performs well for most actions. However, the diving motion, which involves complex poses and manoeuvres, necessitates a specialized network to accurately localize and segment long untrimmed videos. We propose a diving action localization network, a novel temporal convolution network-based block that accurately localizes action with low latency using spatio-temporal features over video frames. Our proposed method accurately predicts the start and end of the diving motion with more than 90% classification accuracy and latency accuracy of 90% within a threshold of $[-1, 1]$ marker positions. We also investigate the optimal context required to achieve high localization accuracy with low latency.

In the next chapter, we combine the pose estimation methods with the action localization network to present a comprehensive diving motion parameter extraction framework.

Diving Motion Physical Parameters Extraction

7

> *"The beauty of a move lies not in its appearance but in the thought behind it."*
>
> — **Aaron Nimzowitsch**
> Chess player

In the realm of sports, where each move holds a unique significance. The field of sports video analysis [144, 147, 148, 149] seeks to unveil the intricacies behind each athletic move to gain an upper edge in a highly competitive sports. It aims to provide objective measures to compare the athlete's performance, e.g. during training. For a given sports performance, video analysis can help to improve the athlete's technique with real-time feedback [209, 210, 211, 212, 213].The existing feedback provided after manual video analysis is time-consuming, laborious and error prone. By precisely identifying and analyzing specific movements, techniques, and gestures, coaches can offer targeted feedback to help athletes refine their skills with greater accuracy.

Furthermore, athletes can be benchmarked against key indicators, facilitating a data-driven approach to training. This data-driven feedback loop helps set realistic goals, track progress over time, and tailor training regimens to address specific weaknesses or areas for improvement. Thus, it assists sport trainers in technique analysis, thereby increasing the overall efficiency of the coaching process [214].

The paramount advantage of sports parameter analysis is the identification of biomechanical patterns that may contribute to injury risk. By understanding an athlete's movement patterns, trainers can implement targeted injury prevention programs, prescribe corrective exercises, and modify training loads to mitigate the risk of common sports-related injuries.

Although the use of wearable sensors such as accelerometers or optical markers can provide more accurate estimates, it often disturbs the training and provides poor results in outdoor setup. Thus, athletes tend to dislike the wearable sensors during training.

7.1 Overview

In this chapter, we present a comprehensive diving motion analysis framework, as illustrated in Figure 7.1.The integrated framework leveraging both modules of action localization and

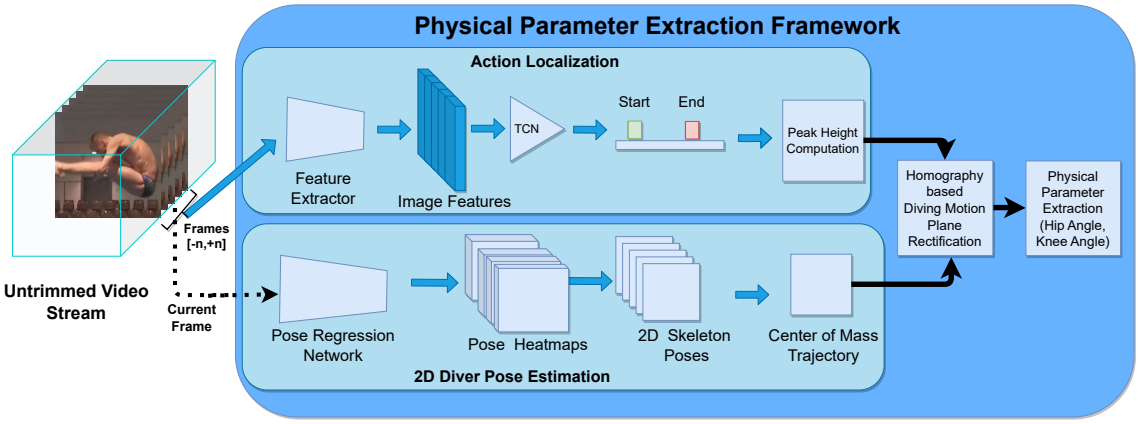


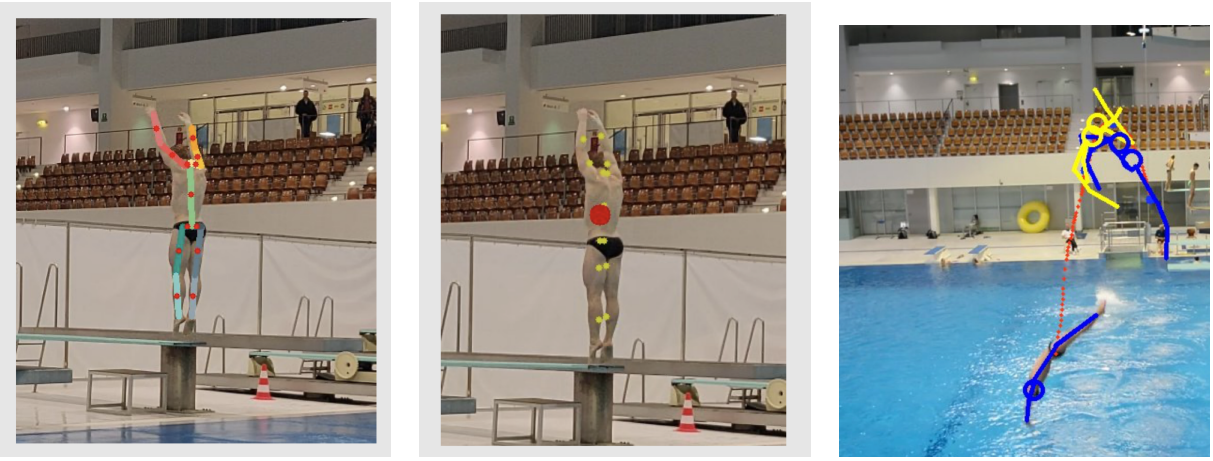
Figure 7.1.: Illustration of physical parameter extraction framework from diving action video.

diving pose regression. Further, the processing of continuous training video stream using action localization module eliminates the necessity for manual intervention in localizing motion or editing sports videos to capture a particular athlete's performance. In addition, the diving pose regression module exclusively assigns pose parameters to relevant diving frames, focusing solely on those pertinent to performance analysis. In this chapter, we make the following key contributions:

- We present a method to compute dive peak height, an important parameter to benchmark dives during training paradigm.
- We propose a novel homography computation method between the diving motion plane and the image-view.
- we provide center of mass trajectory by building upon 2D dive pose regression module and homography computation.
- We also present methods to measure angular motion performed during the diving manoeuvre.
- We present a parameter analysis tool on desktop and mobile phones to assist trainers in order to provide real-time training feedback.

7.2 Center of Mass Trajectory

We compute the COM trajectory as illustrated in Figure 7.2. First, we compute the midpoints of bone segment representing bone centroids (see Figure 7.2a). Next, we compute a weighted sum of centroids to represent the COM (see Figure 7.2b). In order to compute the COM, we used a weighted sum of bone segment centroids as proposed in [215, 216, 217]. Finally, we compute the COM on each frame to obtain the resulting trajectory in the image space (see Figure 7.2c).



(a) Segment midpoints **(b)** COM computation **(c)** COM trajectory

Figure 7.2.: Illustration of the COM computation. First, we compute the midpoints of bone segments representing bone centroids (a). Next, we compute the COM by a weighted sum of centroids (b). The COM trajectory in the image space (c).

7.3 Peak Dive Height Estimation

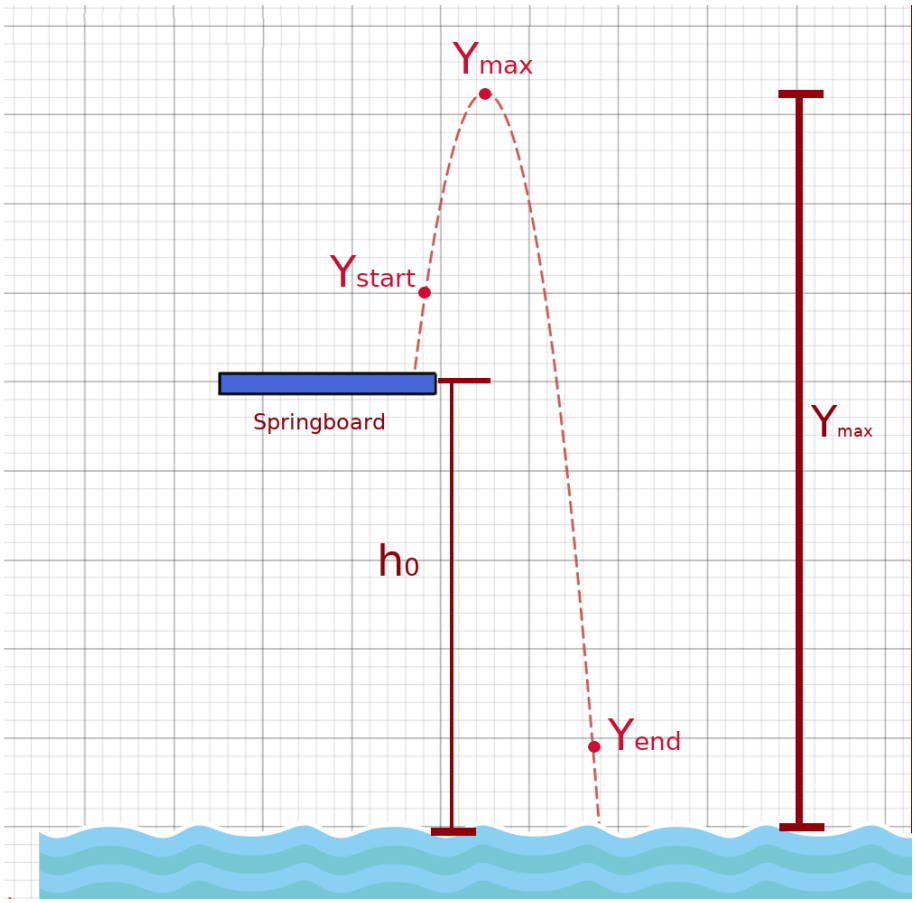


Figure 7.3.: Diving motion peak height estimation

The dive peak height estimation uses the localized action time computed using the action localization TCN Network. The Figure 7.3 shows typical COM trajectory and its peak height Y_{max}. The total time T_{dive} of the diving motion is computed using the equation (7.1). The F_{end} and F_{start} represents the frame numbers in the untrimmed video. The V_{freq} is the recorded video frequency of the untrimmed video, they relate as follows:

$$\Delta T_{dive} = \frac{(F_{end} - F_{start})}{V_{freq}} \tag{7.1}$$

The highest point of the trajectory Y_{max} is computed as:

$$Y_{max} = Y_{start} + \frac{1}{2g}\left(\frac{Y_{end} - Y_{start}}{\Delta T_{Dive}} + \frac{g\Delta T_{Dive}}{2}\right)^2, \tag{7.2}$$

based on [218].

The Y_{start} is the vertical position of the body COM represented in meters at the start of the dive relative to the water-level. Y_{end} represents the body COM position at the end of the dive above water. And g is the gravitational constant. The assumed values for Y_{end} = 1 and Y_{start} = $Y_{start} = h_0 + 1$. The height h_0 is the springboard height above the water and is assumed to be known for a given diving video, e.g. $h_0 = 3$ for a three meter tower. Typically, we assume the body COM for an average athlete as $1m$ above the springboard. Note that in case the exact measurements of height and body mass of the diver are available, then a more precise COM position can be used instead of assumed values.

The following section provides two mathematical derivation for the dive peak height estimation:

7.3.1 Mathematical Derivation 1

Let us assume the position of the center of mass (COM) of the person be $p(t) = [X(t), Y(t)]^T$ as a point in a 2D plain and $v(t) = [U(t), V(t)]^T$ the corresponding velocity.

We use the following Kinematics equations for Projectile motion:

$$Y(t) = Y_{start} + V_{start} + \frac{1}{2}at^2 \tag{7.3}$$

Further, we consider only the y-component of the position and velocity, since this component only contributes to the height.

$$V(t) = V_{start} + at \tag{7.4}$$

At first, we consider the motion of diver to reach the peak height Y_{max} from the start position Y_{start}. Thus, we consider only the vertical component of COM's velocity$V(t)$ and the only acting acceleration as gravity $-g$. Let $t_{start} = 0$ be the initial time, Y_{start} as initial height and $V(t_{start})$ as initial velocity.

Then, we know at the peak height Y_{max}, the velocity$V(t)$ would be equal to zero i.e

$V(t_{max}) = 0$ at time t_{max}
Using equation (7.4) we can compute this time via:

$$\begin{aligned} 0 &= V_{start} - gt_{max} \\ \Rightarrow V_{start} &= gt_{max} \end{aligned} \tag{7.5}$$

replacing $V_{start} = gt_{max}$ from 7.5, we get,

$$\begin{aligned} Y_{max} &= Y_{start} + gt_{max} * t_{max} - \frac{1}{2}gt_{max}^2 \\ &= Y_{start} + gt_{max}^2 - \frac{1}{2}gt_{max}^2 \\ &= Y_{start} + \frac{1}{2}gt_{max}^2 \end{aligned} \tag{7.6}$$

Now, we consider the free fall of the diver from Y_{max} to Y_{end} in equation 7.3. The initial velocity for the free fall is zero i.e $V_{t_{max}} = 0$ .The time take for the rest of the dive is obtained by subtracting time take to reach max height t_{max} from time taken for the complete dive t_{dive} i.e $t_{Dive} - t_{max}$.

Therefore, we get using equation 7.3.

$$\begin{aligned} Y_{end} &= Y_{max} + V_{t_{max}} * (t_{Dive} - t_{max}) - \frac{1}{2}g(t_{Dive} - t_{max})^2 \\ Y_{end} &= Y_{max} + 0 - \frac{1}{2}g(t_{Dive} - t_{max})^2 \\ Y_{end} &= Y_{max} - \frac{1}{2}g(t_{Dive} - t_{max})^2 \end{aligned}$$

$$\begin{aligned} \frac{1}{2}g(t_{Dive} - t_{max})^2 &= Y_{max} - Y_{end} \\ (t_{Dive} - t_{max})^2 &= \frac{2}{g}(Y_{max} - Y_{end}) \\ t_{Dive} - t_{max} &= \left(\frac{2}{g}(Y_{max} - Y_{end})\right)^{\frac{1}{2}} \end{aligned}$$

$$t_{max} = t_{Dive} - \left(\frac{2}{g}(Y_{max} - Y_{end})\right)^{\frac{1}{2}} \tag{7.7}$$

replacing t_{max} from equation 7.7 in 7.6.

$$Y_{max} = Y_{start} + \frac{1}{2}g\left(t_{Dive} - \left(\frac{2}{g}(Y_{max} - Y_{end})\right)^{\frac{1}{2}}\right)^2$$

Now, simplifying the expression under the square term, we get

$$
\begin{aligned}
Y_{\text{max}} &= Y_{\text{start}} + \frac{1}{2}g(t^2_{\text{Dive}} - 2t_{\text{Dive}}\left(\frac{2}{g}(Y_{\text{max}} - Y_{\text{end}})\right)^{\frac{1}{2}} + \left(\frac{2}{g}(Y_{\text{max}} - Y_{\text{end}})\right)) \\
Y_{\text{max}} &= Y_{\text{start}} + \frac{gt^2_{\text{Dive}}}{2} - gt_{\text{Dive}}\left(\frac{2}{g}(Y_{\text{max}} - Y_{\text{end}})\right)^{\frac{1}{2}} + Y_{\text{max}} - Y_{\text{end}} \\
0 &= Y_{\text{start}} - Y_{\text{end}} + \frac{gt^2_{\text{Dive}}}{2} - gt_{\text{Dive}}\left(\frac{2}{g}(Y_{\text{max}} - Y_{\text{end}})\right)^{\frac{1}{2}} \\
gt_{\text{Dive}}\left(\frac{2}{g}(Y_{\text{max}} - Y_{\text{end}})\right)^{\frac{1}{2}} &= Y_{\text{start}} - Y_{\text{end}} + \frac{gt^2_{\text{Dive}}}{2} \\
g\left(\frac{2}{g}(Y_{\text{max}} - Y_{\text{end}})\right)^{\frac{1}{2}} &= \frac{Y_{\text{start}} - Y_{\text{end}}}{t_{\text{Dive}}} + \frac{gt_{\text{Dive}}}{2} \\
\left(\frac{2}{g}(Y_{\text{max}} - Y_{\text{end}})\right)^{\frac{1}{2}} &= \frac{1}{g}\left(\frac{Y_{\text{start}} - Y_{\text{end}}}{t_{\text{Dive}}} + \frac{gt_{\text{Dive}}}{2}\right)
\end{aligned}
\tag{7.8}
$$

Now, squaring both sides,

$$
\begin{aligned}
\frac{2}{g}(Y_{\text{max}} - Y_{\text{end}}) &= \frac{1}{g^2}\left(\frac{Y_{\text{start}} - Y_{\text{end}}}{t_{\text{Dive}}} + \frac{gt_{\text{Dive}}}{2}\right)^2 \\
Y_{\text{max}} &= \frac{1}{2g}\left(\frac{Y_{\text{start}} - Y_{\text{end}}}{t_{\text{Dive}}} + \frac{gt_{\text{Dive}}}{2}\right)^2 + Y_{\text{end}}
\end{aligned}
\tag{7.9}
$$

replace $t_{Dive} = \Delta T_{Dive}$ such that $\Delta T_{Dive} = t_{Dive} - t_{start}$ i.e in case the start time is not zero, we get a generic form of the equation as:

$$
Y_{max} = Y_{End} + \frac{1}{2g}\left(\frac{Y_{start} - Y_{end}}{\Delta T_{Dive}} + \frac{g * \Delta T_{Dive}}{2}\right)^2 \tag{7.10}
$$

7.3.2 Mathematical Derivation 2

We use the following Kinematics equation for projectile motion, where a position of the COM of an object is described with $p(t) = [X(t), Y(t)]^T$ as a point in a 2D plane and its corresponding velocity $v(t) = [U(t), V(t)]^T$. Since the vertical (Y-axis) component only contributes to the height, we consider only the y-component of the COM position and velocity.

$$
Y(t) = Y_{start} + V_{start} + \frac{1}{2}at^2 \tag{7.11}
$$

$$
V(t) = V_{start} + at \tag{7.12}
$$

Furthermore, we are only interested in the trajectory of the COM in the air, the only acceleration acting on the COM is gravity, i.e. $a = -g$. Here, t_0 is the initial time and

$Y_{start} = Y(0), V_{start} = V(0)$ are the initial position and velocity in y-direction, respectively. The highest point (Y_{max}) can now be computed as follows. We know, the velocity at this point in time (t_{max}) is 0 i.e. $(V(t_{max}) = 0)$.
Using equation (7.12) we can compute this time via:

$$\begin{aligned} 0 &= V_{start} - gt_{max} \\ \Rightarrow t_{max} &= \frac{V_{start}}{g} \end{aligned} \tag{7.13}$$

Using equation (7.11 and 7.13) we obtain:

$$\begin{aligned} Y_{max} &= Y_{start} + V_{start}t_{max} - \frac{1}{2}gt_{max}^2 \\ &= Y_{start} + V_{start}\frac{V_{start}}{g} - \frac{1}{2}g\left(\frac{V_{start}}{g}\right)^2 \\ &= Y_{start} + \frac{(V_{start})^2}{2g} \end{aligned}$$

Now, utilizing (7.12) we can write: $V_{start} = V(t) + gt$. We can now approximate V_{start} via an average approximation as:

$$\begin{aligned} V_{start} &\approx V(\frac{\Delta T_{Dive}}{2}) + g\frac{\Delta T_{Dive}}{2} \\ &\approx \frac{\Delta Y}{\Delta T_{Dive}} + g\frac{\Delta T_{Dive}}{2} \\ &\approx \frac{Y_{end} - Y_{start}}{\Delta T_{Dive}} + g\frac{\Delta T_{Dive}}{2}. \end{aligned} \tag{7.14}$$

Inserting this into equation (7.14) gives:

$$Y_{max} = Y_{start} + \frac{1}{2g}\left(\frac{Y_{end} - Y_{start}}{\Delta T_{Dive}} + \frac{g\Delta T_{Dive}}{2}\right)^2 \tag{7.15}$$

as required.

Note that, although there are slight difference with terms and reverse sign of $(Y_{end} - Y_{start})$ in the final equations of Derivation 1 and Derivation 2[1]. The peak height obtained using both final equations were validated to provide the exact same answer for any diving motion.

7.4 Proposed Diving Plane Homography Estimation

In this section, we derive how we estimate a homography to identify points from the 2D image plane from unknown camera view to a physical 2D plane [2]. A homography provides a projective geometry of two cameras and a world plane. In simple terms, homography maps

[1] The second variant of the derivation is credited to Dr. Bertram Taetz
[2] The contribution and development of the homography estimation section are credited to Dr. Bertram Taetz

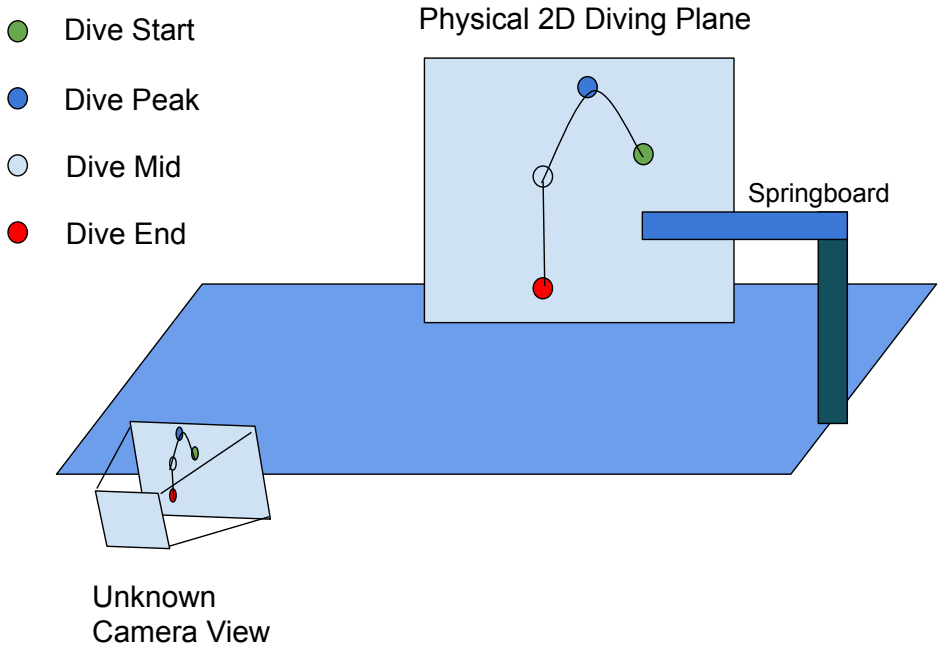


Figure 7.4.: Diving motion COM trajectory estimation using homography.

images of points which lie on a world plane from one camera view to another. However, our setup consists of a single camera and without any known camera calibration procedure or precise depth information available about diving plane.

We begin with the motion assumptions from [218] for water diving. The primary assumption is that the divers COM follows a projectile motion while the diver is airborne. Thus, the COM can be completely described with a 2D parabola in a physical 2D plane. We aim to register the physical 2D plane (looking from an orthogonal point of view onto the plane) with the camera plane (might have a non-orthogonal view angle) via the homography estimation. The Figure 7.4 shows the unknown recording image view plane, physical diving plane and selected four center of mass trajectory points for registration. The mathematical derivation to approximate physical peak heights derivation is explained in section 7.3.2. The homography

$$H = \begin{bmatrix} h_{11}, h_{12}, h_{13} \\ h_{21}, h_{22}, h_{23} \\ h_{31}, h_{32}, h_{33} \end{bmatrix}$$

maps the image coordinates (x, y) to the respective physical plane (x', y'), i.e.

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} \sim H \begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} \tag{7.16}$$

$$x' = \frac{h_{11}x + h_{12}y + h_{13}}{h_{31}x + h_{32}y + h_{33}}, \tag{7.17}$$

$$y' = \frac{h_{21}x + h_{22}y + h_{23}}{h_{31}x + h_{32}y + h_{33}}. \tag{7.18}$$

Given 4 points, a homography could be solved with a unity constraint to cover the last degree of freedom, i.e.

$$Ah = 0, \tag{7.19}$$

$$s.t. \quad \|h\|_2 = 1, \tag{7.20}$$

with $h = [h_{11}, h_{12}, h_{13}, h_{21}, h_{22}, h_{23}, h_{31}, h_{32}, h_{33}]^T$ and A can be derived by rearranging equation (7.17) and equation (7.18).
Via the COM tracking from section 7.2, we can easily identify multiple points in the image plane ($(x, y)_i$) that are not lying on one line. The challenge in the present setting is that we can only estimate the respective physical points in the y-direction, via the equation (7.11) and the respective time-stamp of the dive. However, we do not have any information about the x-direction, since no starting angle is given that could be used to estimate this component via a starting velocity.
In this work, we propose to estimate a coefficients of a quadratic function jointly with the homography that defines the x coordinates, together with a relation on certain points that have to hold via the Cayley-Klein metric, defined via the cross-ratio.
To obtain a minimal number of parameters to estimate, we describe the projectile motion in the physical plane, via a parabola centered at the highest point $a_1 = Y_{max}$, with the restriction that we only allow concave functions, i.e.

$$y(t) = -a_0 x(t)^2 + Y_{max}, \tag{7.21}$$

$$s.t. \quad a_0 > 0. \tag{7.22}$$

This can also be formulated as unconstrained equation

$$y(t) = -\exp(\tilde{a_0}) x(t)^2 + Y_{max}, \tag{7.23}$$

with $\tilde{a_0} = \log a_0$. The $x(t)-$ coordinate for the respective point $y(t)$ on the parabola can be described as

$$x(t) = \begin{cases} -\sqrt{\frac{-y(t)+Y_{max}}{\exp(\tilde{a}_0)}} & : t > t_{max} \\ \sqrt{\frac{-y(t)+Y_{max}}{\exp(\tilde{a}_0)}} & : t \leq t_{max} \end{cases}, \tag{7.24}$$

with t_{max} given via equation (7.13).

Using equation (7.24) and equation (7.23), we can describe coordinates (x', y') on the physical plane for different instances in time. However, to do this we still need to estimate $\tilde{a}_0$, which leads to, at least, one condition for the x-direction on the physical plane. To this end, we create a ratio that has to match between to two coordinate systems. Let us start by computing the time (t_f) that the COM needs to fall from the highest point (Y_{max}) at time t_{max} until the y position is back to the same height as the initial height (Y_{start}). Looking at (7.13) at $t = t_{max}$, we have $V_{start} = 0$ and $Y_{start} = Y_{max}$. Thus, the time until the initial height is reached again, can be computed by rearranging the equation

$$Y_{start} = Y_{max} - \frac{g t_f^2}{2} \tag{7.25}$$

$$\Rightarrow t_f = \sqrt{\frac{2(Y_{max} - Y_{start})}{g}}. \tag{7.26}$$

Thus the total time from take-off until the same height is reached again is $t_r = t_{max} + t_f$. The time span t_r describes the amount of time the diver needs to jump, pass through the highest point and come back to the same height. Together with $t_0 = 0$ and t_{max}, the respective points ($p(t)$) of the COM trajectory in the image plane, we can span a triangle in the 2D plane, were the divers COM lies in. We use this fact to create a relation between a line pointing along the x direction (in the plane) and the y direction (in the plane) defined as follows

$$d_x^{\text{img}} = p(t_r) - p(t_0) \tag{7.27}$$

$$p^{avg} = p(t_0) + \frac{1}{2}(d_x^{\text{img}}) \tag{7.28}$$

$$d_y^{\text{img}} = Y_{max} - (p^a vg). \tag{7.29}$$

To obtain the relation, we exploit the Cayley-Klein metric, which is based on the cross ratio that is invariant to perspective changes. The metric measures the distance between two points a and b, denoted as $|ab|$, with respect to a quadric, e.g. a circle, with intersection points q and p; it is defined as

$$d(a, b) = C \log(\frac{|bp||qa|}{|ap||qb|}), \tag{7.30}$$

for a constant C.

Since we have two distances (x direction and y direction) that we compare we define one quadric for both distances, defined to be the circle with center p^{avg} and radius $r = \|d_y^{\text{img}}\|_2$. The circle can thus be parametrized as

$$x(f) = p^{avg} + r\cos(f) \tag{7.31}$$

$$y(f) = p^{avg} + r\sin(f) \tag{7.32}$$

The points p and q for the respective directions (x and y, thus denoted as p_x, q_x and p_y, q_y respectively) can now be computed via the intersection point f_x^* and f_y^* of the following lines with the circle

$$p_x = p^{avg} + f_x^* d_x^{\text{img}} \tag{7.33}$$
$$q_x = p^{avg} - f_x^* d_x^{\text{img}} \tag{7.34}$$
$$p_y = p^{avg} + f_y^* d_y^{\text{img}} \tag{7.35}$$
$$q_y = p^{avg} - f_y^* d_y^{\text{img}}. \tag{7.36}$$

The intersections can therefore be computed as

$$f_x^* = \sqrt{\frac{r^2}{\|d_x^{\text{img}}\|_2^2}} \tag{7.37}$$
$$f_y^* = \sqrt{\frac{r^2}{\|d_y^{\text{img}}\|_2^2}}. \tag{7.38}$$

Now we can compute the relation between x and y direction via a quotient of the following distances

$$rel_{xy} = \frac{d(p(t_r), p^{avg})}{d(Y_{max}, p^{avg})}. \tag{7.39}$$

Note that the constant C cancels out, since we use the same quadric for both distances. This finally gives a relation for the x direction in the physical plane that we can use in the estimation of the homography with included parabola constraint to obtain the remaining degree of freedom for the parabola. The distance in x direction, in the physical plane can thus be computed from the velocity in y direction, multiplied with the ratio and the time from the time span t_f, as

$$x^{phy}(t_f) = \frac{-(Y_{max} - Y_{start})}{t_f} rel_{xy} t_f \tag{7.40}$$
$$= -rel_{xy}(Y_{max} - Y_{start}). \tag{7.41}$$

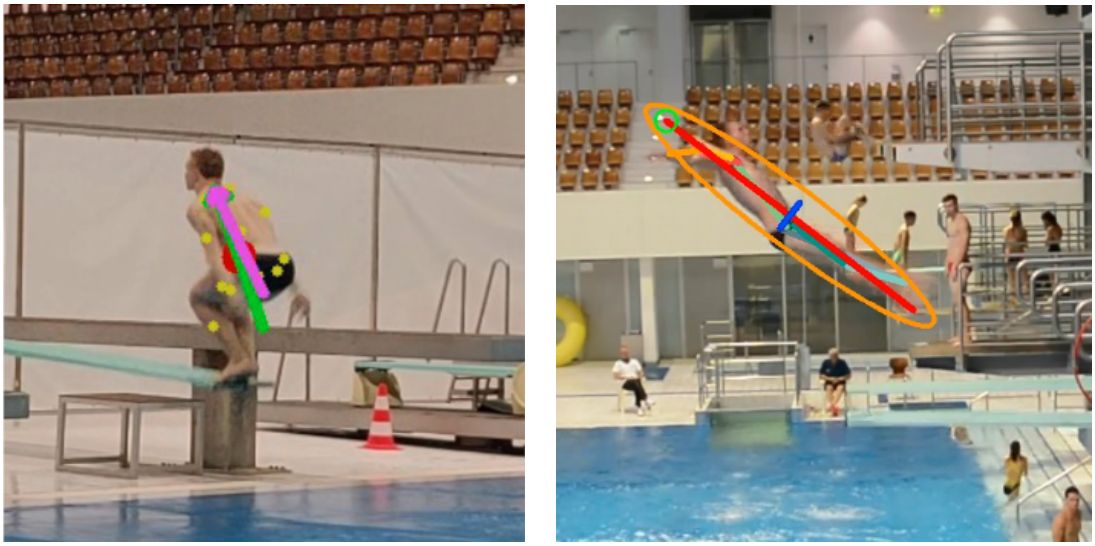
The final optimization problem to solve reads as,

$$\min_{\tilde{h}} \|A\tilde{h}\|_2 \tag{7.42}$$
$$s.t. \quad \|h\|_2 = 1 \wedge x^{phy}(t_f) = x(t_f), \tag{7.43}$$

were $\tilde{h} = [h^T, \tilde{a}_0]^T$.

7.5 Body Angular Rotation

In this section, we work on measuring rotation of the body around itself during the dive by using two methods. We represent a body axis (green vector) relative to the Y-axis as shown in Figure 7.5a. The body axis is computed using least-square fit of the estimated 2D pose keypoints. As the ending joints of arms(wrists) and legs(ankle) are very unstable, we computed body axis using only torso joints(magenta) as shown in Figure 7.5a. Another way to measure is to approximate the diver's body as an ellipse to fit the estimated 2D keypoints. We use the major axis (red) of ellipse to track the angular rotation as shown in Figure 7.5b. We observed with empirical study that the ellipse approximation method to be more stable as compared to the least-square fit method, which is very prone to unstable outputs of occluded joints.



(a) Least-square fit method. **(b)** Ellipsoid approximation method.

Figure 7.5.: Measurement of body axis in the images. Figure (a) shows the body axis (green arrow) computed using least-square method over estimated all the 2D pose joint keypoints. The body axis using only torso joints joints(magenta). Figure(b) shows ellipsoid approximation of the body represented in orange color. The body axis is measured using the major axis(red) in an ellipse

The ellipse approximation helps to estimate the angular body rotation in the trajectory as shown in Figure 7.6.The Figure 7.7a shows the body rotation measure with respective vertical y-axis running through ellipse center. The Figure 7.7b shows the cumulative body rotation from the start of the dive until the end of the dive. We further apply linear Kalman Filter [219] along with RTS smoothing [220] to have more robust estimates of angular body rotation (refer Figure 7.7b). The filtering uses a constant velocity model using state space with 2D pixel location and corresponding velocity in each dimension.

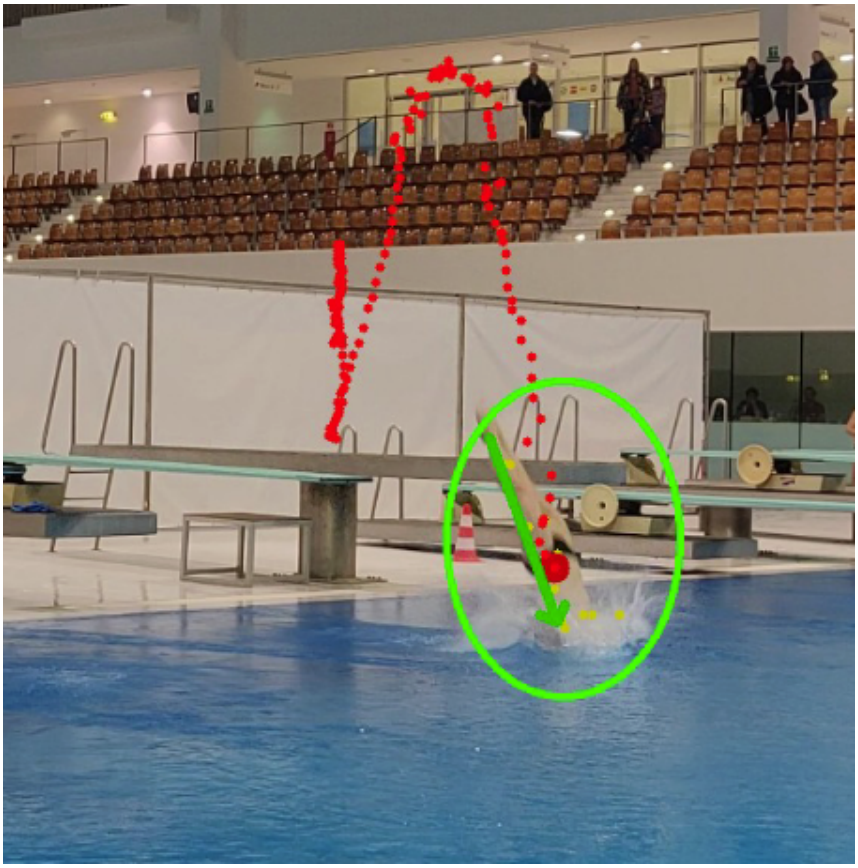
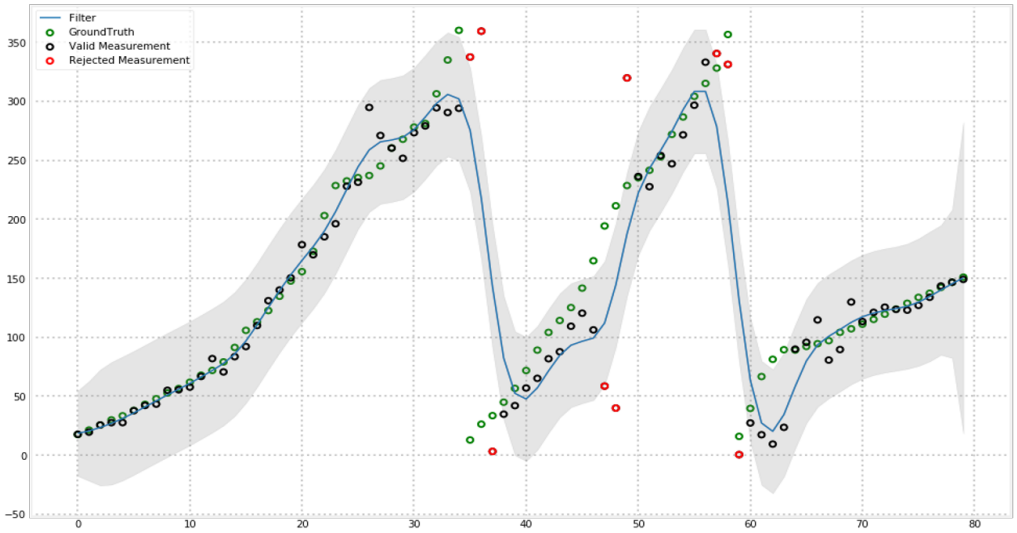


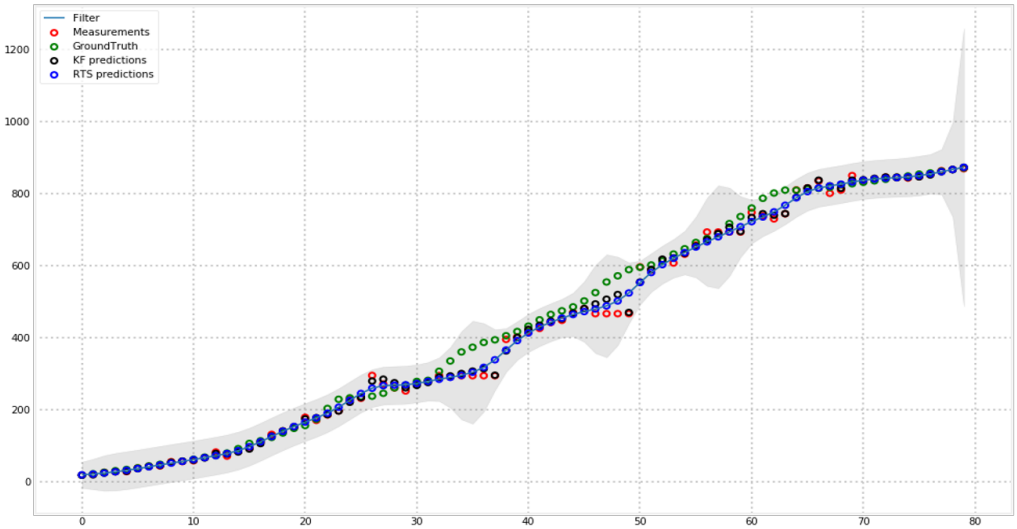
Figure 7.6.: The diving trajectory of estimated center of mass in each dive and body axis(using ellipsoid method) while ending the dive.

7.6 Evaluation of Physical Parameters

We measure different physical parameters of diving performance : COM, Hip Angle and Knee angle. Also, we only compare DiveNet with Unipose-FT inorder to have fair comparison of models, trained on same datasets setup. The Table 7.1 shows the accuracy of the COM prediction over the test set. The median COM prediction error is 5.87 pixels on DSV and 6.05 pixels on IAT dataset. This is significantly lower than Unipose-FT model which results in 34.30 and 37.00 pixel on DSV and IAT datasets respectively . Due to the lack of availability of actual transformation of images to the real diving trajectory plane, we restrict our evaluation to pixel space. The mean joint angle error are as shown in Table 7.2 for hip and knee joints are 10.39 degrees and 9.73 degrees respectively. The peak height estimation sensitivity is influenced by the action localization module. We observed a mean peak height error of 0.14 meters on DSV dataset and 0.21 meters on the IAT dataset as shown in Table 7.3. The Figure 7.8 shows the error distribution plot for COM (left) and joint angles (right plot). The COM error distribution shows 95 percentile of errors lie under 16.64 pixels for the complete diving testset. Similarly, the Figure 7.9 show the errors distribution plot of 2D angle errors of two trained models. The median angle error of Unipose-FT model is 8.2825 and 95 percentile error threshold lies at 35.435 degrees. The



(a) The body rotation angle plot of a complete dive.



(b) The filtered cummulative body rotation angle for a complete dive.

Figure 7.7.: Body Angle Filtering.

median angle error of DiveNet is marginal higher at 8.52 degrees and the 95 percentile threshold lies at 38.96 degrees.

	DSV Dataset					IAT
Dive Height	**3m**	**5m**	**7.5m**	**10m**	**Mean**	**3m**
UniposeFT	43.96	31.16	30.04	32.05	34.30	37
DiveNet	**6.26**	**5.44**	**5.23**	**5.23**	**5.54**	**6.05**

Table 7.1.: The table shows the COM physical parameter mean pixel errors over different height jumps performed in the test set of diving datasets

		DSV Dataset					IAT
	Dive Height	**3m**	**5m**	**7.5m**	**10m**	**Mean**	**3m**
Unipose-FT	Hip Angle	7.96	10.35	9.12	9.71	**9.28**	8.51
	Knee Angle	8.09	9.35	13.13	8.88	9.86	**6.03**
DiveNet	Hip Angle	7.77	12.19	11.70	9.91	10.39	**8.20**
	Knee Angle	7.90	8.63	14.56	7.82	**9.73**	6.30

Table 7.2.: The Table shows a comparison of the mean joint angle errors in degrees over different height jumps in the diving test set. The lower the angle value, the better is the model performance. We observed less than 1 degree difference and no significant variation in Hip angle and Knee angle error between the two models.

	DSV Dataset					IAT
Dive Height	**3m**	**5m**	**7.5m**	**10m**	**Mean**	**3m**
Peak height	± 0.19	± 0.13	± 0.13	± 0.13	± 0.14	± 0.21

Table 7.3.: The Table shows the sensitivity in peak height estimation of the diving trajectory measured in meters over different height jumps performed in the test set of DSV diving datasets.

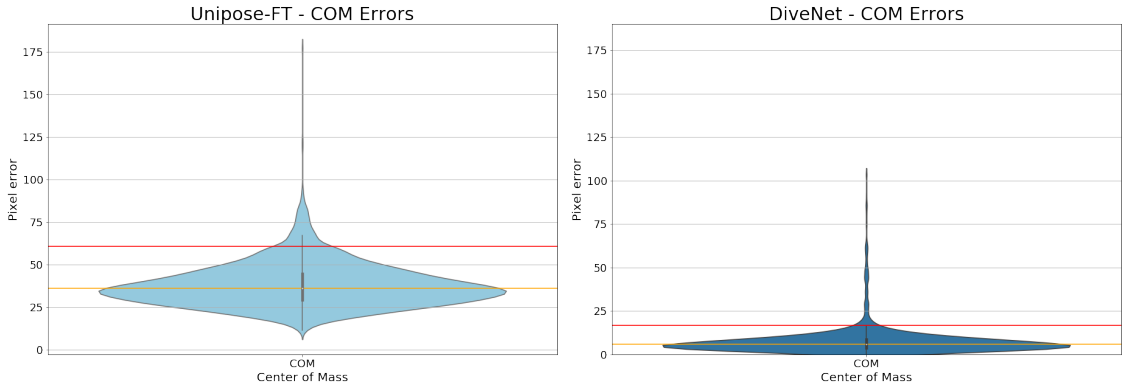


Figure 7.8.: The plots show the COM error distribution of Unipose-FT (top) and DiveNet (bottom) model. The median COM error (left-orange line) of Unipose-FT model is 36.08 pixels and 95 percentile error threshold (left-red line) lies at 60.76 pixels. The median COM error of DiveNet (right-orange line) is significantly lower, at 5.87 pixels and the 95 percentile threshold (right-red line) lies at 16.64 pixels.

The quantitative evaluation to measure the accuracy of estimated homography is not feasible due to lack of availability of 3D real-world ground truth data of body poses. We evaluate qualitatively measure the projected center of mass pixels and skeleton points on to diving physical plane as shown in the Figure 7.10. The projection shows the estimated point on center of mass trajectory perfectly coincides with estimated COM trajectory based on projectile motion. The projection also helps us recover the true angles between the joints by rectifying the distorted skeleton segments due to varying camera perspective view.

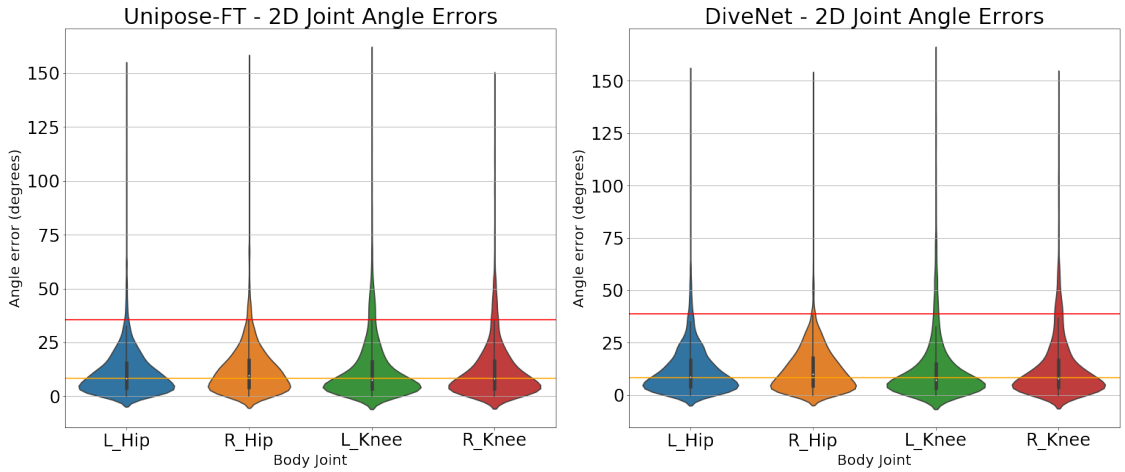


Figure 7.9.: The plots show 2D joint angle error distribution of Hip Angles and Knee Angles for Unipose-FT (left) and DiveNet (right). The median angle error (left-orange line) of Unipose-FT model is 8.28 degrees and 95 percentile error threshold (left-red line) lies at 35.435 degrees. The median angle error of DiveNet (right-orange line) is marginally higher at 8.52 degrees and the 95 percentile threshold (right-red line) lies at 38.96 degrees

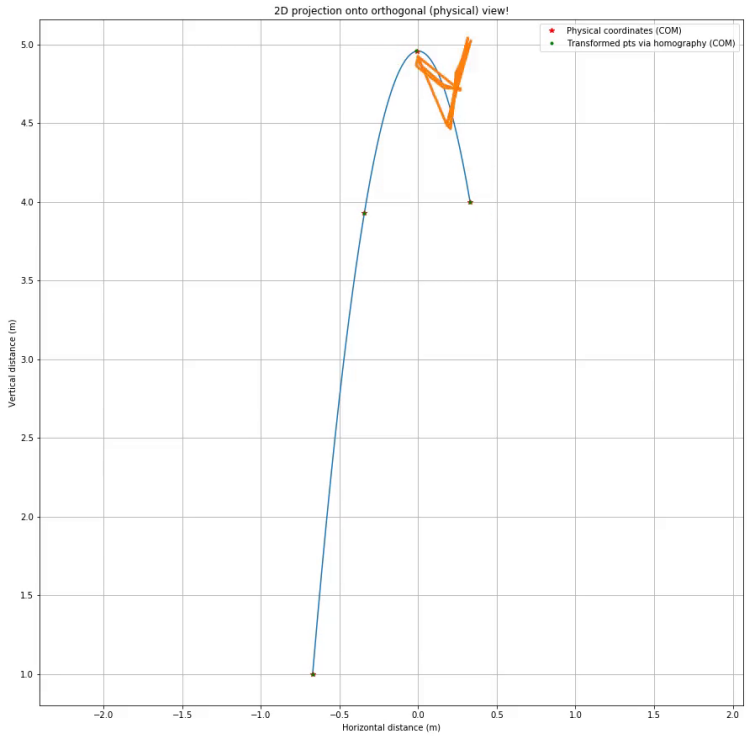


Figure 7.10.: Homography based projections on Orthogonal Physical view.

7.7 Demonstrator Tool

We also present a demonstration and diving analysis desktop tool as shown in Figure 7.11. The desktop tool provides a real-time feedback solution which integrates pose and action localization models with an user interface showing different motion parameter. The coach or trainer can assess the performance of dive during the training. The tool enables the athletes to benchmark against his/her previous training performance video to measure the precise improvement along with a qualitative assessment.

Furthermore, we convert BlazePose (Tensorflow model) and MobileHumanPose (Pytorch model) to Open Neural Network Exchange (ONNX) format [178]. The optimized ONNX models are converted into tensorflow lite models. We also provide analysis tools on mobile platform for the optimized deep learning models. The Figure 7.12 shows the screenshots

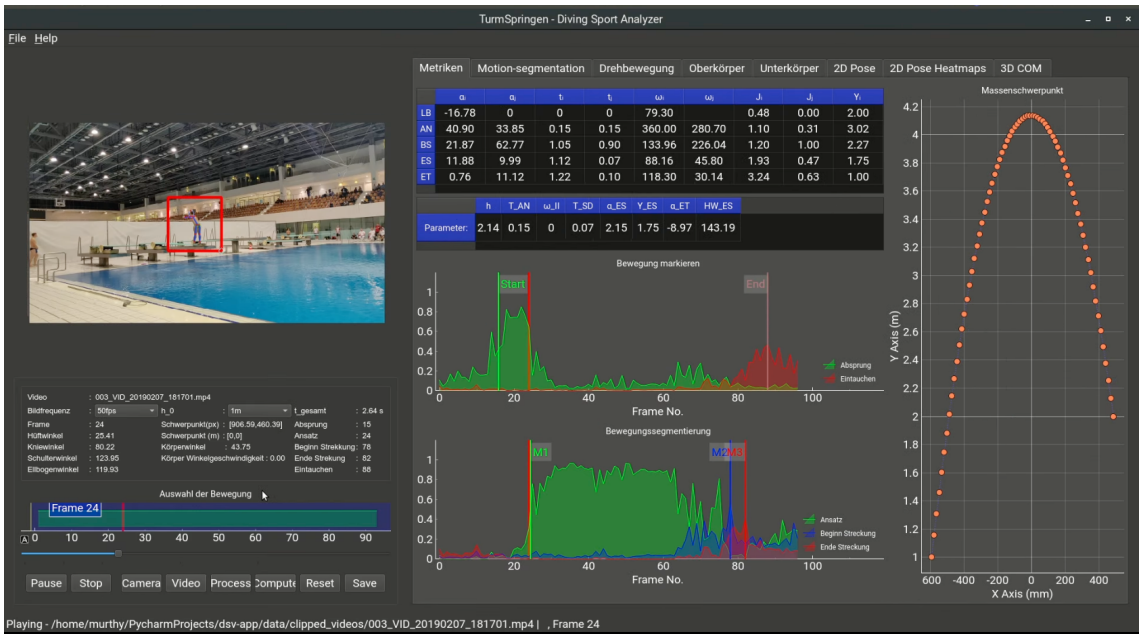


Figure 7.11.: Illustration showing demonstrator tool to extract different pose parameters in recorded training session diving videos.

of pose estimation results inferred by BlazePose model. The mobile app visualizes the 2D human pose skeletal keypoints overlayed on each frame of the selected diving video for analysis.

The Figure 7.13 shows the screenshot of the iOS app. The app uses the MobileHumanPose tflite model to estimate 3D human pose on DSV (top row) and IAT (bottom row) datasets. The app does an model inference on a single frame. The MobileHumanPose model estimates 3D human pose keypoints on the device. The application interface shows two sections. The top section shows the input image fed to the model and the bottom section shows the estimated 3D human pose for a given frame.

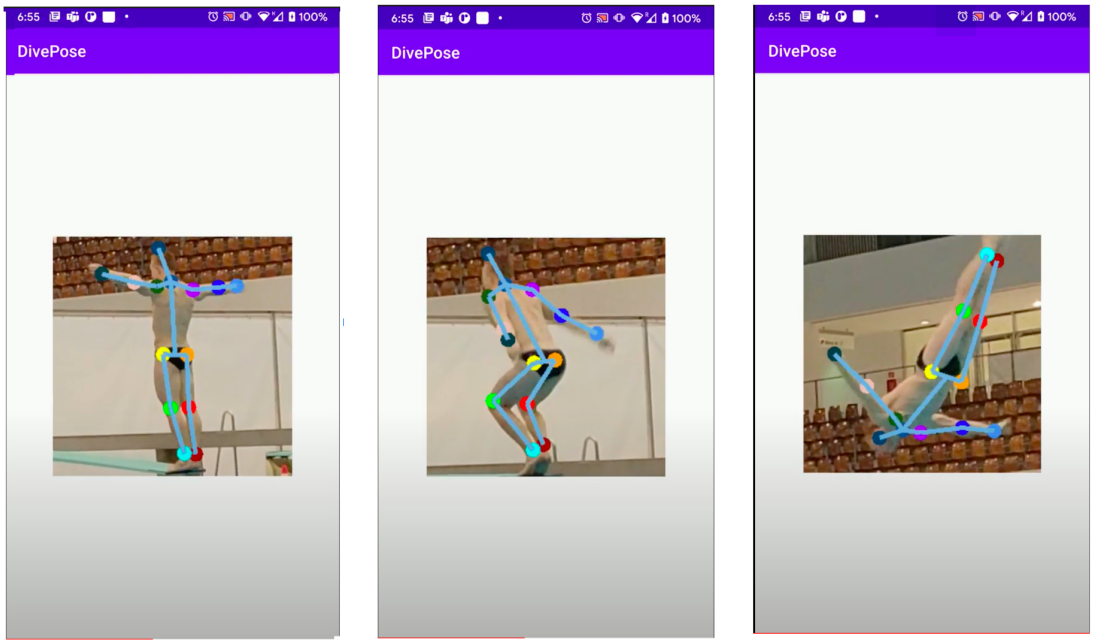


Figure 7.12.: BlazePose model integrated on android phone.

7.8 Conclusion

In this part of thesis, we presented a diving motion analysis framework, DiveNet to extract performance parameters. A generic action localization (or recognition) of daily activities often performs well for most actions. However, the diving motion, which involves complex poses and manoeuvres, necessitates a specialized network to accurately localize and segment long untrimmed videos. Thus, we propose a diving action localization network, a novel temporal convolution network-based block that accurately localizes action with low latency using spatio-temporal features over video frames. Our proposed method accurately predicts the start and end of the diving motion with more than 90% classification accuracy and latency accuracy of 90% within a threshold of $[-1, 1]$ marker positions. We also investigate the optimal context required to achieve high localization accuracy with low latency.

In addition, we train a dive pose regression model, which outperforms all the 2D pose regression models on diving dataset. The trained pose regressor is robust for all diving heights and provided accurate poses even in challenging scenarios. The trained models are used to achieve a high accuracy on COM trajectory estimation around 6 pixels and 2D joint angle estimation accuracy around 10 degrees. We believe the presented physical parameter extraction can easily be generalized to other sports action tasks. Thus, we would extend the learning to other sports, especially sports with varying speeds of motion, as part of future work.

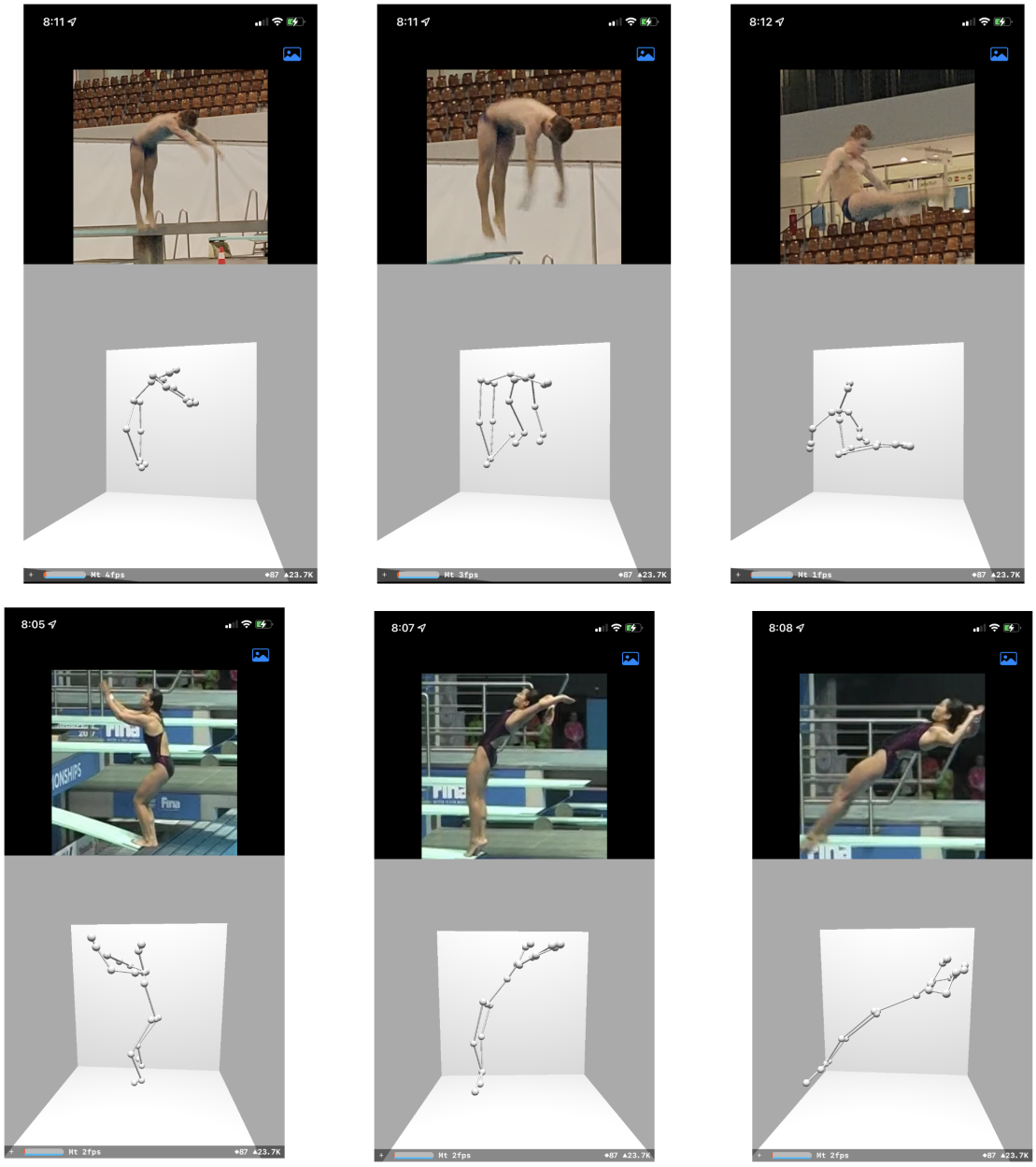


Figure 7.13.: MobileHumanPose model inference results on DSV (top row) and IAT dataset(bottom row).

This concludes the part II of the thesis work. In the next part, we focus on modeling human pose under partial visibility in an exo-centric camera view.

Acknowledgments

I would like to express my sincere appreciation to Dr. Bertram Taetz(DFKI) and Dr. Thomas Köthe(IAT) for their invaluable feedback and guidance during the development of this part

of thesis. Their insightful comments and constructive suggestions greatly contributed to the refinement of the content.

I extend my gratitude to students Asmita Mittal, Ashish Shukla for their contributions to the divenet action localization and pose regression codebase, upon which our implementation is partially based. I would also thank diligent efforts by Ankit Kumar, Arpit Lekhra and Aditya Dash in annotating diving datasets. Their efforts provided a solid foundation for the work presented.

Finally, I am thankful to German Swimming Federation (Deutscher Schwimm-Verband) - DSV, for their financial support and providing access to their amenities, which played a crucial role in the successful completion of this research.

Part III

Exocentric view: Partial Visibility



Illustration of in-cabin 3D car driver pose estimation scenario

Overview

Overview

In this part, we address the challenges of handling partial visibility of human body in the exocentric perspective of the camera, with a focus on advancing driver pose estimation to develop more robust and accurate models in this domain. The objective is to regress 3D human pose in the context of in-cabin car driving simulation environment. The research is motivated by the importance of driver pose estimation in driver-assistance systems, which provides crucial information about the driver's activity and awareness.

The information presented in this part is structured as follows:

- **Chapter 8** introduces a novel driver pose estimation dataset comprising 19 participants, each captured in 14 videos engaging in various driving activities. Provides comprehensive data including RGB, IR, and depth images, alongside SMPLX model parameters and annotations such as 2D/3D joint coordinates along with activity labels.
- **Chapter 9** aims to develop an end-to-end pipeline for estimating 3D human pose from IR and depth images. A joint volumetric heatmap autoencoder is used to compress and improve accuracy of the 3D pose. Further, the framework presents a transformer-based architecture for learning fused information from IR and depth sensors.

- **Chapter 10** Proposes a PointConvNet architecture incorporating random sampling and a Local Feature Aggregation (LFA) module. Discusses the role of the LFA module in mitigating feature loss associated with random sampling techniques.

Related Publications

[1] Pramod Murthy, Onorina Kovalenko, Ahmed Elhayek, C Couto Gava, and Didier Stricker. “3D human pose tracking inside car using single rgb spherical camera”. In: *ACM Chapters Computer Science in Cars Symposium (CSCS)*. 2017.

[2] Xue Yang. “3D Human Pose Estimation using Pointclouds”. Master’s thesis. Informatik, RPTU Kaiserslautern-Landau, Oct. 2022.

[3] Shubham Gupta. “3D Driver Pose Estimation from IR and Depth Images”. Master’s thesis. Informatik, RPTU Kaiserslautern-Landau, Nov. 2022.

In-Cabin Car Driver Pose and Activity Monitoring Dataset

8

> *I'm really looking forward to a time when generations after us look back and say how ridiculous it was that humans were driving cars.*
>
> — **Sebastian Thrun**
> Entrepreneur, educator and computer scientist

The future where autonomous car technologies redefine our relationship with vehicles as the above quote resonates with the transformative journey underway. Human Pose Estimation (HPE) provides a nuanced understanding of human body postures and movements, which is critical aspect in the evolution process. Specifically, within the domain of autonomous driving, the emphasis on driver pose estimation becomes paramount. The evolution from human-driven vehicles to autonomous marvels hinges on ensuring not just convenience but also safety. Nevertheless, the safety implications are profound, as driver pose estimation can recognize potential hazards, such as instances where the driver is at risk of falling asleep or experiencing a medical emergency, prompting timely system interventions. Beyond safety considerations, HPE enhances the overall user experience within autonomous vehicles by dynamically adjusting the interior environment, including climate control and infotainment, based on the driver's preferences and comfort. Additionally, the data collected through driver pose estimation proves valuable for post-incident analysis, insurance claims, and gaining insights into driver behavior across different scenarios. Meeting regulatory standards, certain regions mandate the incorporation of driver monitoring systems in autonomous vehicles to ensure driver readiness to assume control as needed. Overall, the integration of HPE into autonomous driving systems aligns with the broader trend of making vehicles more aware of their surroundings and the individuals within them, promising not only enhanced safety but also sophisticated and personalized in-car experiences.

8.1 Introduction

SAE International, formerly known as the Society of Automotive Engineers, which develops standards with a special focus on automotive; delineates six distinct levels of driving

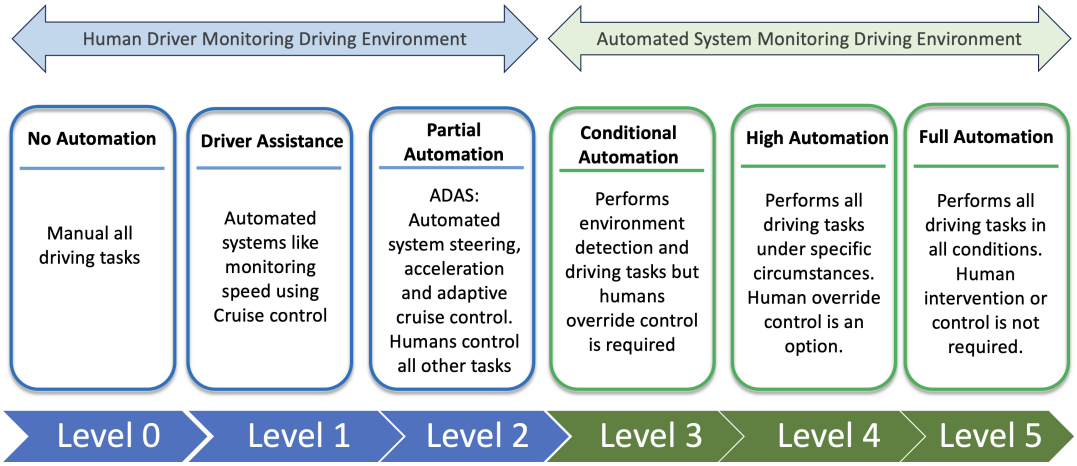


Figure 8.1.: Own representation of driving automation levels defined by ASE [221]

automation. According to the SAE Levels of Driving Automation standard [222]. Within this classification, Level 0 signifies a complete absence of driving automation, while Level 5 represents the pinnacle of full driving automation [223, 224, 225]. A comprehensive presentation of each of these levels is graphically presented in Figure 8.1.

In light of various technical, legal, and other pertinent considerations, it is anticipated that autonomous driving systems will predominantly remain within the realms of partial automation (Level 2) and conditional automation (Level 3) for an extended period in the foreseeable future. Levels 2 and 3 entail that the human driver retains primary control over the vehicle, with the autonomous driving system serving as an auxiliary aid. This assistance is primarily directed toward facilitating the execution of relatively uncomplicated driving tasks.

Notably, while autonomous driving technology undeniably affords drivers greater ease and convenience compared to traditional driving, it concurrently introduces a new set of safety concerns. In traditional driving, continual focus is imperative to prevent potential accidents. However, in Level 5 autonomous driving, drivers do not necessarily need to have constant attention towards road conditions, enabling their engagement in the non-driving activities. Yet, at lower autonomy levels, any lapse in attention from the driver can result in hazardous situations and accidents. Furthermore, public comprehension of the distinctions between autonomous driving levels remains limited.

For instance, Tesla's official website explicitly clarifies that its current autopilot features require active driver supervision and do not render the vehicle fully autonomous [226, 227]. Despite this, a notable number of accidents stem from negligent drivers overly relying on autonomous systems. Hence, the integration of driver pose estimation becomes pivotal in assessing a driver's safety within the realm of autonomous driving. This technology serves the dual purpose of averting driver distraction and reducing accident probabilities.

In recent years, substantial advancements have materialized in the domain of Human Pose Estimation (HPE), driven in part by the widespread availability of cost-effective depth

cameras. This proliferation has led to significant strides in HPE research grounded in depth maps or point clouds.

There has been active research to exploit different input modalities such as RGB images, depth maps, or distinguishing between monocular HPE and multi-view HPE. Various types of input data present inherent advantages and drawbacks. RGB images represent the most ubiquitous form of input data but may prove less effective in low-light environments. Infrared (IR) images excel in scenarios characterized by darkness or inclement weather conditions, yet they are unsuitable when targeting distant objects. Depth maps contain valuable depth information but are not universally available across all datasets. When addressing the HPE task within the context of 3D driver pose estimation, the initial classification identifies it as a single-person pose estimation problem, necessitating the inclusion of depth information in the output key point coordinates. Given the range of lighting conditions encountered in real-world driving scenarios, it is prudent to also employ depth maps or point cloud data as the preferred additional input modality.

Notwithstanding these advancements, there remains a conspicuous paucity of studies and datasets concerning 3D human pose estimation for a in-car environment. Relative to other contexts, the in-car setting presents unique challenges, chief among them being pronounced partial visibility of the body due to highly restricted data acquisition angles. Consequently, data collection and pose estimation within this environment pose formidable challenges.

Our focus in this part of thesis is to estimate the pose of the driver in-car cabin environment. The occlusion of the in-car environment is more severe than that of other environments, and the angle of data gathering is quite constrained, making data acquisition and pose estimation extremely difficult. Another significant obstacle in researching the problem of driver posture estimation is the absence of appropriate in-car environment HPE datasets.

The contributions in the current chapter can be succinctly summarized as follows:
Introduction of a novel driver pose estimation and activity monitoring dataset, encompassing 19 participants (5 female and 14 male), each of whom is captured in 14 videos. These videos comprise 12 instances with the vehicle's lights on and 2 instances with the lights off. Participants engage in a variety of common activities, both in manual driving mode and automatic driving mode. The dataset is comprehensive, providing RGB, IR, and depth images. Annotations encompass essential elements such as 2D/3D joint coordinates,SMPL-X parameters with activity identifiers.

The most common inputs for human pose estimation tasks are RGB images but we restrict our focus in this part of the thesis to only depth and IR images. In the next section, we cover recent literature focused on estimating pose using depth or infrared camera images.

8.2 Related Works

Xiong et al. proposed an anchor-based end-to-end network, i.e., Anchor-to-Joint regression network(A2J) [228]. The anchor points are local regressors of the joints to obtain global-local information and can enhance the generalization of the prediction results. A2J uses ResNet as the backbone network to improve the computational efficiency; three branches are used for anchor proposal, prediction in-plain offset and depth estimation. Moon et al proposed V2V-PoseNet [229], a deep learning network that estimates per-voxel likehood from a 3D voxelized grid. Unlike other methods that obtain 3D joint coordinates by direct regression from 2D depth images, they first convert depth image into voxelized grid and then estimate the per-voxel likehood of each joint. Wang et al using inference embedded multi-task learning [230] to estimate human pose from depth images. They first generate the heat map of body parts by fully convolutional network and then determine the optimal configuration to measure the appearance and geometric consistency. In the next section, we look at the car driving pose and activities datasets.

8.3 Driving Datasets

To perform effectively, deep learning models need a large collection of data. There are publicly available synthetic datasets for in-cabin car monitoring provide activity and pose annotations. The model learned on these synthetic datasets often perform with limited accuracy in real world driving videos. Thus, there is a lack of publicly available datasets for 3D driver pose with real RGB or RGBD videso performing various tasks. To this end, we record a 3D driver pose and activity monitor dataset as presented in the section 8.3.1. Nevertheless, let us first look at the two synthetic representative datasets providing in-cabin car pose dataset: TiCam [231] and SVIRO [232] in the following subsections.

TiCaM

TiCaM (Jigyasha et al. [231]) is a dataset consisting of images from time-of-flight camera placed inside cars. They offer depth, RGB, and infrared images with front car cabin that were captured by a Kinect Azure. The driving simulation system, the monitoring system, and the ground truth sensor system make up the whole in-car test platform, as shown in Figure 8.2. Three projectors, a wide-angle projection screen, and a realistic in-cabin model make up the driving simulation system's gear. The OpenDS emulator software creates the simulated scenes [233]. A wide-angle 2D/3D camera system is used as the monitoring system. In order to simultaneously watch the driver and passenger, the camera is mounted near to the rearview mirror. OptiTrack [43] serves as the motion capture and tracking system for the ground truth sensor system. It records several dynamic driving scenarios inside a driving simulator. Additionally, they offer a synthetic dataset of in-cabin using simulation software. It includes ground truth annotations for 2D and 3D object detection, as

well as example segmentation, for both datasets. Furthermore, it offers activity annotations for real TICaM dataset has about 6.7K real images and 3.3K synthetic images. However, the real TICaM datasets do not have pose annotations.

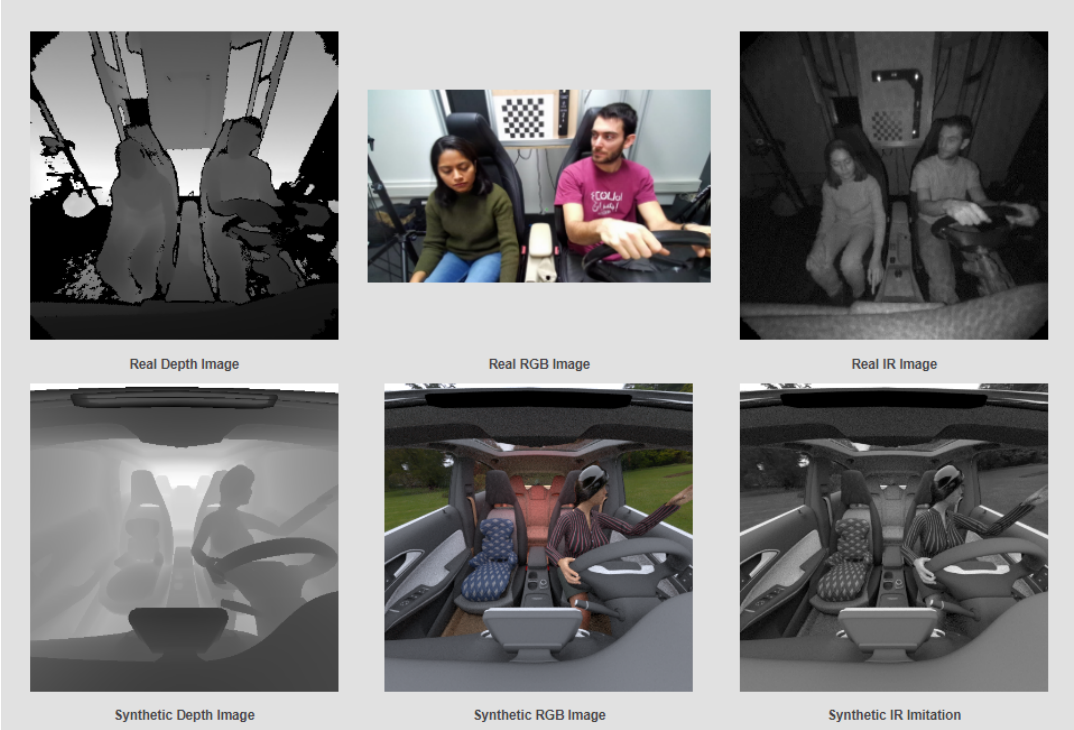


Figure 8.2.: Example images of TICaM dataset consists of the real depth images, real RGB image, real infrared image, synthetic depth image, synthetic RGB image, synthetic infrared image [231].

There are 13 participants in the TICaM dataset with a total of 19 common activities performed by driver and passengers, which includes sitting, driving and conversing.

SVIRO

Steve et al. [232] introduced the Synthetic dataset for Vehicle Interior Rear seat Occupancy detection and classification (SVIRO). This dataset comprises 25,000 scenes from ten distinct vehicles, offering diverse simulated sensor inputs and corresponding ground truth data. SVIRO serves as a valuable resource for evaluating the generalization capabilities and reliability of machine learning models, especially when trained on a limited set of variations. The objective was to establish a standardized benchmark for testing approaches. Figure 8.3 illustrates some examples from the SVIRO dataset.

Synthetic 3D landscapes were crafted and rendered using Blender 2.79, a versatile and open-source 3D computer graphics program. Within this simulated environment, child seats were denoted as "realistic child safety seats" or "child restraint systems" (CRS). A selection of commercially available chairs served as the dataset source, from which a 3D model was derived for simulation purposes. DEP Meshworks [234] is used to generate 3D

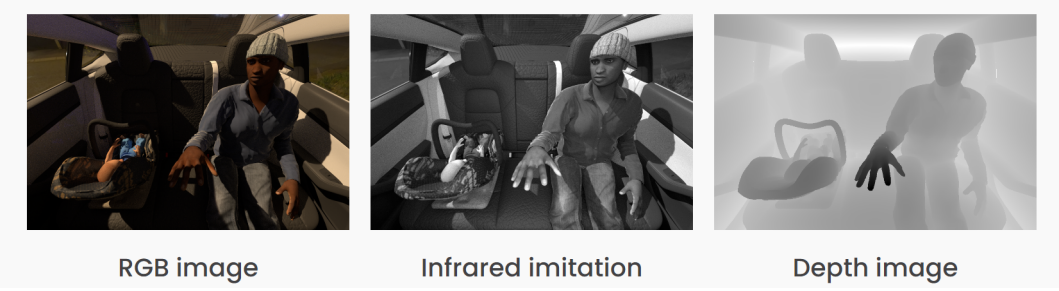


Figure 8.3.: Example images of SVIRO dataset consists of the synthetic depth image, synthetic RGB image, synthetic infrared image [232].

models, facilitating the determination of each 3D object's perception by the camera under simulated lighting conditions. To define the reflection properties and visual characteristics of the scene's objects, textures (Albedo, Normal, and Roughness images) from Textures.com were utilized with permission. High Dynamic Range Images (HDRI) from HDRI Haven contributed to the creation of natural backgrounds and lighting.

MakeHuman 1.2.0 [235] served as the open-source 3D graphic program for generating randomized human models (adults, children, and babies) along with their clothing, supplemented by additional clothes from the community assets. Vehicle 3D models were acquired from Hum3D [236], and various common items like pillows, boxes, and backpacks were downloaded from Sketchfa. Human models were created with identical, neutral facial expressions, and two random poses were assigned to each person in each scenario, blended using spherical linear interpolation (Slerp) for intermediate poses. The object type in each position for every landscape was randomly chosen, aiming to avoid repetition within the same scene.

The dataset comprises scenes involving ten distinct vehicles: a Lexus GS F, a Mercedes A-Class, a BMW X5, a BMW i3, a Hyundai Tucson, a Tesla Model 3, a Toyota Hilux, a Toyota iQ, and a Ford Escape. Varying window configurations in different cars lead to diverse lighting conditions, and some vehicles feature only two rear seats instead of three. Each scene in the dataset includes a set of images with corresponding ground truth information, encompassing a rendered RGB image of the scene without a red bulb next to the camera, a rendered RGB image with a red lamp next to the camera displayed in grayscale. The dataset annotations consists of instance segmentation map, bounding boxes for all objects in the scene, keypoints for human positions and a depth map.

8.3.1 DREICAP-3D

In this study, we introduce a novel dataset named **DR**iv**E**r **I**n-car **C**abin **A**ctivity monitoring and **P**ose **3D** (DREICAP-3D) dataset to recreate authentic driving scenarios. Our methodology involves employing the DFKI cabin simulator, mirroring the approach taken with the TICaM dataset, utilizing the Kinect Azure camera to capture RGB, infrared, and depth

images during the recording sessions. The Table 8.1 gives a brief comparison of each defined dataset and Figure 8.4 shows sample images selected from the DREICAP dataset.

The dataset encompasses 19 participants, comprising 14 men and 5 women, who actively took part in the data collection process. Markers were strategically placed on crucial joints to facilitate the tracking of human body movements using OptiTrack technology, which records reflective markers. A total of 21 joints, including the head, chin, neck, clavicles, shoulders, elbows, wrists, hands, chest, spine, torso, pelvis, hips, and knees, were tracked with markers.

Each participant contributed to the dataset by recording 12 light-on sequences and 2 light-off sequences, resulting in a total of 14 sequences. Representative images from these recordings are illustrated in Figure 8.4. The dataset is organized into six distinct action sequences (sessions), each encompassing five common driving-related activities, contributing to a total of 37 sub-activities. Additional insights into these activities are provided in Figure 8.5, categorizing the 37 activities into 7 specific scenarios.

For each session, participants were required to repeat the process twice—once in automatic driving mode and once in manual driving mode. Additionally, each participant randomly selected one session to repeat twice with the lights off and in two different driving modes. Table 8.2 outlines the specific activities for each session. All activity annotations, including activity ID, start frame, end frame, illumination, driving mode, and other relevant details, were manually recorded.

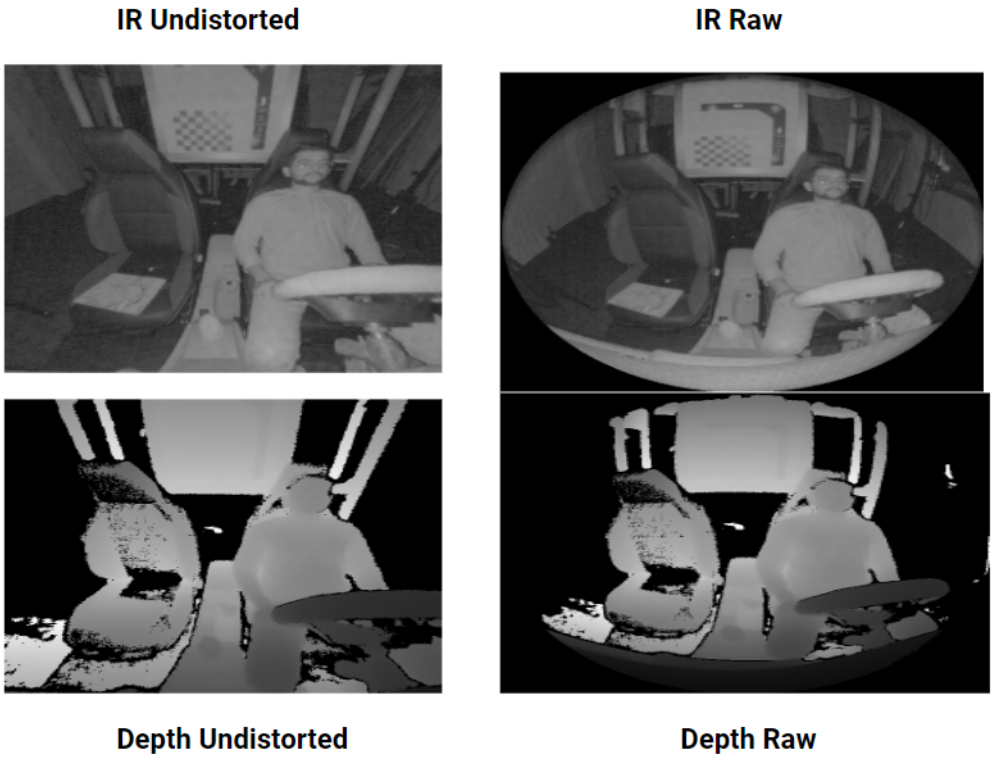


Figure 8.4.: The figure shows the sample images in DREICAP-3D dataset. There are four images consisting of IR and depth with fish eye view (raw) and perspective view (undistorted) images.



Figure 8.5.: Different activities in the DREICAP-3D dataset

	Driving Pose Datasets		
	SVIRO	**TiCaM**	**DREICAP-3D**
RGB	Yes	Yes	Yes
IR	Yes	Yes	Yes
Depth	Yes	Yes	Yes
Pose 2D	Yes	Yes	Yes
Pose 3D	No	No	Yes
SMPL-X	No	No	Yes
Activity	No	Yes	Yes
Bounding Box	Yes	Yes	Yes
Segmentation	Yes	Yes	No

Table 8.1.: The table describes the detailed comparison of each in-car cabin datasets.

session 1	session 2	session 3
touch screen	move hand to head/face	check weather
hands-free call answering	take selfie	adjust the ventilation
lean forward	adjust radio	look at advertisement
look right	look around	drink water
move hand to head/face	strong light interference	look right, turn wheel
read paper/book	wear glasses	-
-	remove glasses	-
session 4	**session 5**	**session 6**
use laptop	answer the phone	eat snack
turn up the phone volume	greet pedestrian	yawn
look left, turn wheel	check map	close eyes
text message	put on the hat in bag	nodding
turn backwards	remove hat	use lip balm

Table 8.2.: The specific activities of each session in DREICAP-3D dataset



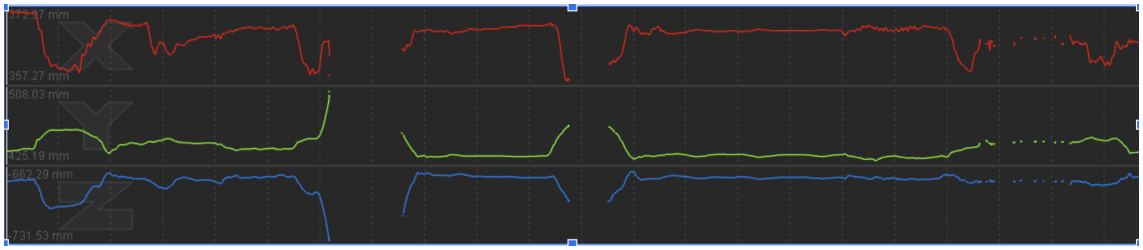
Figure 8.6.: Illustration of motive application showing a snapshot of attached markers for a sample motion sequence recording of DREICAP-3D dataset.

8.4 Automated Mocap Annotation

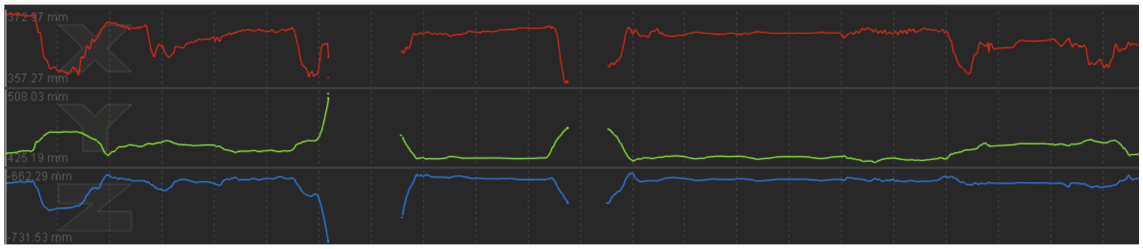
We use the OptiTrack system [43] to capture the driver's motion, and subsequently manually annotated the activities for all the recorded video sequences. However, the raw motion capture point clouds exhibited considerable noise, necessitating a cleanup process. Notably, instances such as holding a handheld object or a glass of water introduced numerous



(a) Raw data with jitter



(b) Raw data after smoothing



(c) Reconstructed missing data

Figure 8.7.: The plots show reconstruction of motion capture data tracked over motion sequence in the dataset. Plot (a) show the raw jitter data captured from markers. Plot (b) shows the smoothed raw data after jitter clean up. Plot(c) shows the final reconstructed missing data for short gaps using Motive application.

reflection points around the marker on the hand. Resolving this required identifying the accurate marker reflection point based on its running track and its corresponding position in each camera view. Addressing these challenging issues forms a crucial aspect of our presented work.

Throughout the entire video footage, short gaps and jitter were consistently present in the movement track of the marker (see Figure 8.6 and Figure 8.7). While interpolation methods like linear, constant, cubic, and pattern-based functions could address minor gaps as shown in Figure 8.7c. The larger gaps posed a challenge with no definitive solution. Additionally, the occlusion caused by the car seat and the constraints of camera positions makes capturing or markers position reconstruction on body parts like elbows particularly challenging.

The problem mocap point cloud labeling is also complex due to presence of ghost points and missing points to be considered together. For instance, the Figure 8.8 shows the gray bar at the top of the figure representing the motion trajectory of the captured points, and the black parts represents occlusion. Assigning colors to the different bars represents assigning labels to the points. The middle of the figure shows that labels are assigned to

each motion track. The label represented in red is the ghost point, and the red oblique line means that the ghost point is incorrectly perceived as a marker. Tracks with the same label can finally be merged together, as shown in the lower part of the Figure 8.8
This process also requires considering three constraints: 1) each point can be assigned

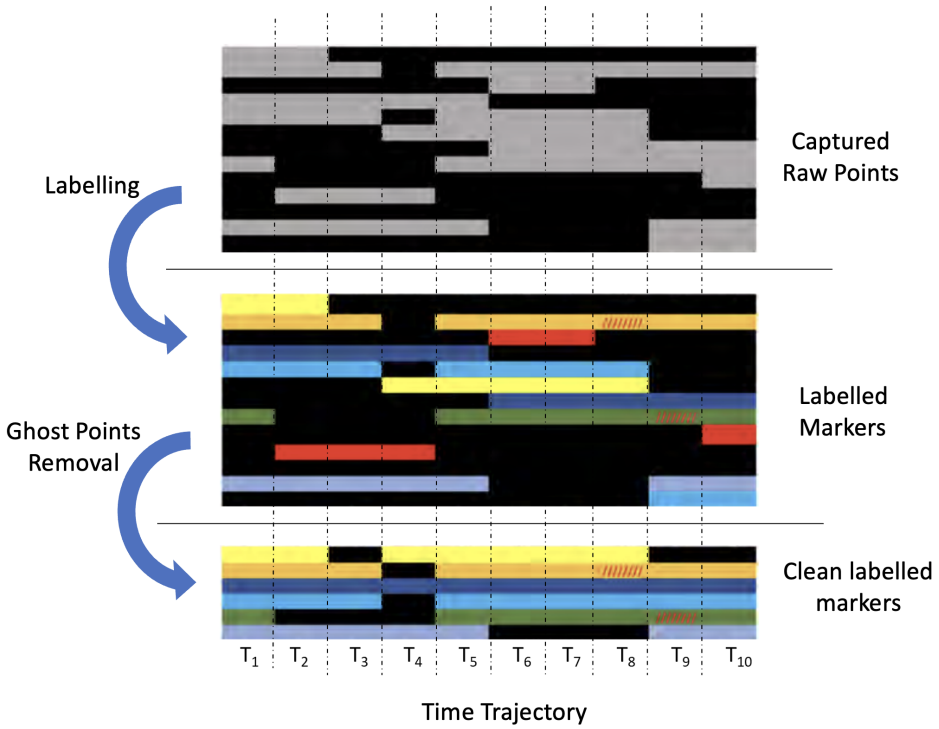


Figure 8.8.: Illustration of the process of assigning labels(color) to mocap point clouds(grey) trajectory [237] by removing ghost points(red)

only one label, and similarly, each label can be assigned to only one point; 2) each point can belong to only one track; and 3) ghost point label can be assigned to multiple points or can appear in multiple tracks at the same time.

We use Solving Optical Marker-based mocap Automatically (SOMA) [237] model to automatically label the original point cloud obtained by marker-based optical motion capture (mocap). Initially, we generate a synthetic dataset for training the SOMA model by labeling the target mocap point cloud. The trained model is then used to finally generate the SMPL-X human body model parameters that fits the point cloud. Interested reader can read appendix section A.5 to know more about the architecture details and the network training.

Synthetic Data Generation

The synthetic training data generation for SOMA model requires a human pose dataset and a defined marker layout template. We use the defined marker template selected during the Optitrack recording session. The markers are placed appropriately on SMPL human models surface using the selected template. Further, we use pose data from AMASS

dataset [105] to train a SOMA model. The AMASS [105] dataset unifies 15 different mocap dataset with common SMPL shape and pose parameters. In addition, to enhance robustness, we need to add some noise.

Given the marker layout template, we can place markers on the human body model. By setting an appropriate distance from the marker to the body surface, we can more realistically simulate the placement of the marker on the clothing.

To enhance robustness, it is also necessary to add some controllable noise to the placement of the markers. For example, the marker can be placed on a randomly selected vertex near the original picked vertex. Or add a random rotation value to the root orientation. And it is also possible to add real noise directly from the AMASS dataset. In addition to adding noise from the markers, the diversity of the synthesized data can also be increased by adding ghost points. The ghost points are randomly selected from a 3D Gaussian distribution, and the mean and standard deviation of the ghost points are required to be equal to the median and standard deviation of the marker locations, respectively. We can also consider the case of occlusion by defining the marker to be equal to 0 when it is occluded.

8.5 Implementation

The cleaned motion capture (mocap) point clouds obtained can be representing the human body or worn clothes. Thus, it is makes difficult to accurately represent the position of joints, often leading to missing joints. The SOMA approach is used to generate human body models that fit the mocap point cloud, and thereby calculate the 3D joint coordinates. The entire pipeline is as shown in Figure 8.9.

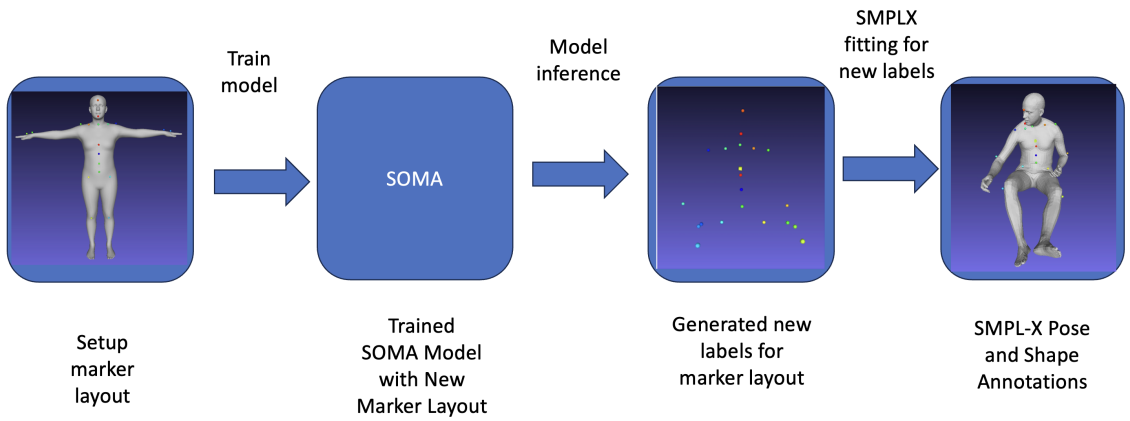
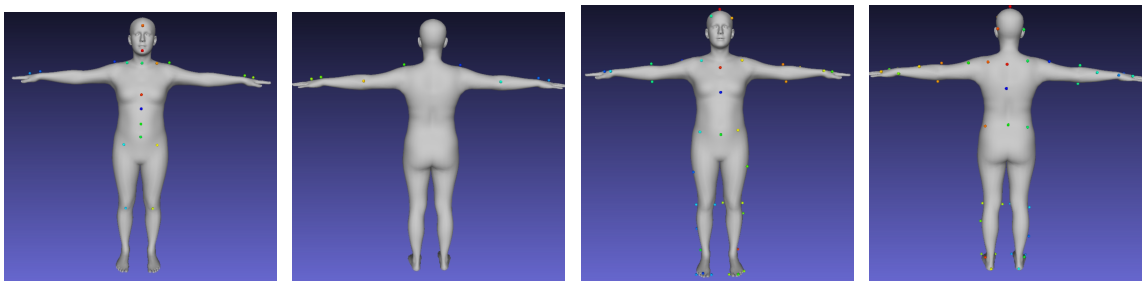


Figure 8.9.: The process of generating smplx annotations for raw mocap point cloud using the SOMA [237] method

Our dataset's marker layout template and the original marker layout template are shown in Figure 8.10. By comparison, we can see that the number of markers in our dataset is

tiny, and the information of body shape cannot be well reflected, which will cause great difficulty in generating the shape parameters of the human body model.



(a) (b) (c) (d)

Figure 8.10.: The figure (a),(b) are front view and back view of markers layout in Optitrack recording dataset respectively. The figure (c) and (d) corresponds to the front view and back view synthetic training

We apply similar training procedure as outlined in [237]. The trained model to label the mocap point clouds. Given the marker layout template and the AMASS dataset, we can generate a synthetic training dataset. All training parameters use default values.

The SOMA model is evaluated using evaluation metrics to measure mean and standard deviation of accuracy and F1 score per frame. F1 score is computed by precision and recall, shown as Equation 8.1. Recall is the ratio of correct predicted labels to actual labels, and precision is the ratio of actual correct labels to predicted labels.

$$F1score = 2 \times \frac{precision \times recall}{precision \sum recall} \tag{8.1}$$

The evaluation results are shown in Table 8.3.

	head	neck	clavicles	elbows	wrists	hands	chest	spine	torso	pelvis	hips	knees
precision	1.00	0.996	0.998	0.998	0.993	0.995	0.109	0.034	0.204	0.97	0.996	0.996
recall	0.996	0.996	0.996	0.994	0.993	0.993	0.108	0.033	0.07	0.725	0.996	0.996
f1-score	0.998	0.996	0.999	0.996	0.993	0.994	0.108	0.033	0.104	0.83	0.996	0.996

Table 8.3.: Evaluation results of the trained model on DREICAP-3D dataset

After labeling all mocap point clouds, we generate the fitted SMPLX human body shapes. The SMPLX model describes the human body through two types of statistical parameters: pose parameters and shape parameters. Thus, the obtained pose and shape parameters are saved as part of dataset annotations. The generated SMPLX models using SOMA are as shown in Figure 8.11.

Since our dataset images and markers do not capture lower leg labels, no ground truth data is available for driver's lower leg segments. Thus, the evalution procedure do not consider any errors occuring in the lower legs. Nevertheless, we focus on the upper body for in-car environment to obtain accurate pose results.

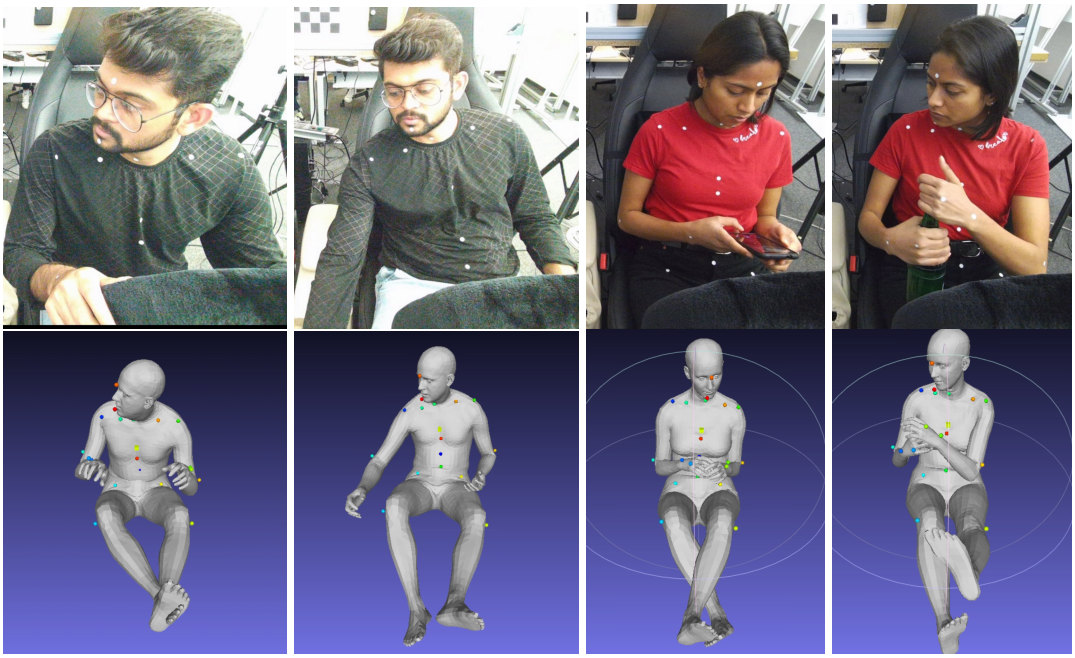


Figure 8.11.: The DREICAP-3D dataset sample images(top row) and their corresponding smplx pose annotations(bottom row).

Finally, we compute the 3D joint coordinates from the SMPLX model by transforming axis-angle joint rotations to rotation matrices. Further, we translate all the joint coordinates after forward kinematics with the root translation to represent in the camera coordinate system.

8.6 Conclusion

In this chapter, we introduced a DREICAP DriverPose dataset, designed to capture driver posture and activities in an in-car simulator environment. The dataset utilizes the DFKI in-cabin car driving simulator with a combination of a Kinect camera and an OptiTrack motion capture system. Comprising 19 participants and 266 videos encompassing 37 distinct actions, the dataset provides synchronized RGB images, IR images, and depth data. Furthermore, we leverage the SOMA method [237] to generate fitted SMPLX human models, obtaining crucial 3D pose and shape parameters. The dataset annotations also consists of 2D keypoints, bounding boxes, and activity labels for research purpose.

In the next chapter, we exploit the recorded DREICAP driver pose dataset to estimate 3D pose using different modalities.

9 Driver Pose Estimation using Compressed Volumetric Heatmap

> *"To deal with a 14-dimensional space, visualize a 3-D space and say 'fourteen' to yourself very loudly. Everyone does it."*
>
> — **Geoffrey Hinton**
> Computer scientist

The above quote humorously suggests the handling of high-dimensional spaces, which can be overwhelming (also known as curse of dimensionality). As the number of dimensions increases, the data becomes increasingly sparse and difficult to analyze effectively. Techniques such as autoencoders are helpful in learning high dimensional data by learning its lower-dimensional representations, enabling more efficient processing and interpretation [238, 239]. Autoencoders also learn a meaningful representation of the input data in an unsupervised manner. This learned representation often captures important underlying structures or patterns in the data, facilitating downstream tasks such as classification or regression. The autoencoder networks are effective in denoising corrupted or noisy data by learning to reconstruct the clean input from the noisy version. This ability is beneficial in scenarios where the data collection process introduces noise or errors [240]. In the present chapter, we use this ability of autoencoders to compress and denoising of high dimensional data along with a volumetric data for 3D pose representation task.

9.1 Introduction

In the realm of driver pose estimation, the utilization of multi-modal imaging, specifically combining infrared (IR) and depth sensors, presents distinct advantages. One notable benefit lies in their enhanced performance under low-light conditions, where traditional imaging methods may falter [241]. IR and depth sensors demonstrate superior functionality in such challenging environments, and their synergistic use can result in superior performance, as each sensor complements the strengths of the other.

With the advent of low-cost depth cameras, there is an increasing amount of research into the use of depth images for human pose estimation. Depth maps typically provide more accurate localization of body parts compared to RGB images alone. Depth information reduces ambiguity, as it directly measures the distance of body parts from the camera [242]. We can estimate the joint coordinates directly from the depth map, a voxelized grid or volumetric heatmaps [117]. We can use depth maps to estimate human pose to better address issues such as heavy occlusion. Prior works successfully showed accurate estimation of 3D joint positions, leading to improved accuracy of pose estimation using depth maps. The authors of RefineNet [243] refined 2D poses by projecting them into 3D using depth maps. Similarly, Crescitell et al. [244] successfully employed both RGB and IR images to predict human pose in inefficient low-lighting conditions. Xiong et al. proposed an anchor-based end-to-end network, i.e., Anchor-to-Joint regression network(A2J) [228]. The anchor points are local regressors of the joints to obtain global-local information and can enhance the generalization of the prediction results. A2J uses ResNet as the backbone network to improve the computational efficiency; three branches are used for anchor proposal, prediction in-plain offset and depth estimation.

Moon et al proposed V2V-PoseNet [229], a deep learning network that estimates per-voxel likehood from a 3D voxelized grid. Unlike other methods that obtain 3D joint coordinates by direct regression from 2D depth images, they first convert depth image into voxelized grid and then estimate the per-voxel likehood of each joint. Wang et al. [230] use multi-task learning to estimate human pose from depth images. They first generate the heat map of body parts by fully convolutional network and then determine the optimal configuration to measure the appearance compatibility and geometric consistency. DoubleFusion [245] is a real-time system for capturing human pose, which uses a double-layer surface representation, where the inner layer is the skeletal and shape parameters based on the SMPL model and the outer layer is the visible surface. Relying only on a single depth sensor, detailed geometry, non-rigid motion and inner body shape can be reconstructed through three steps: joint motion tracking, geometric fusion and volumetric shape-pose optimization.

Using depth maps for pose estimation offers several benefits: Depth-based pose estimation is often more robust to changes in scale. Since depth maps directly encode physical distances, they are less affected by changes in camera distance or zoom levels compared to 2D RGB images. This makes depth-based pose estimation suitable for applications where the camera-to-subject distance varies, such as surveillance or human-computer interaction systems.

For the heavily occluded cases in RGB images, multiple depth maps from multiple views can be used. Hansen et al [246] use fused information from multiple 2D depth images for 3D human pose estimation in the operating room. In such tasks, depth images can protect the privacy of the patient. They initially predicted 2D joint heatmap probabilities are predicted for each depth image. These heatmaps are then fused into a pointcloud. Finally, the pointcloud are voxelized to obtain 3D voxel probabilities. The sensors, such

as time-of-flight (ToF) can provide reliable depth information even in low light conditions where traditional RGB cameras may struggle. Therefore, depth-based pose estimation methods can perform better in challenging lighting conditions, making them suitable for applications in indoor environments or night time surveillance [247].

Moreover, the occluded body parts can be challenging to detect accurately in RGB images, leading to pose errors. However, with depth information, even if a body part is occluded in the image, its position can still be estimated accurately based on its depth relative to visible parts. This enhances the robustness of pose estimation algorithms in real-world scenarios.

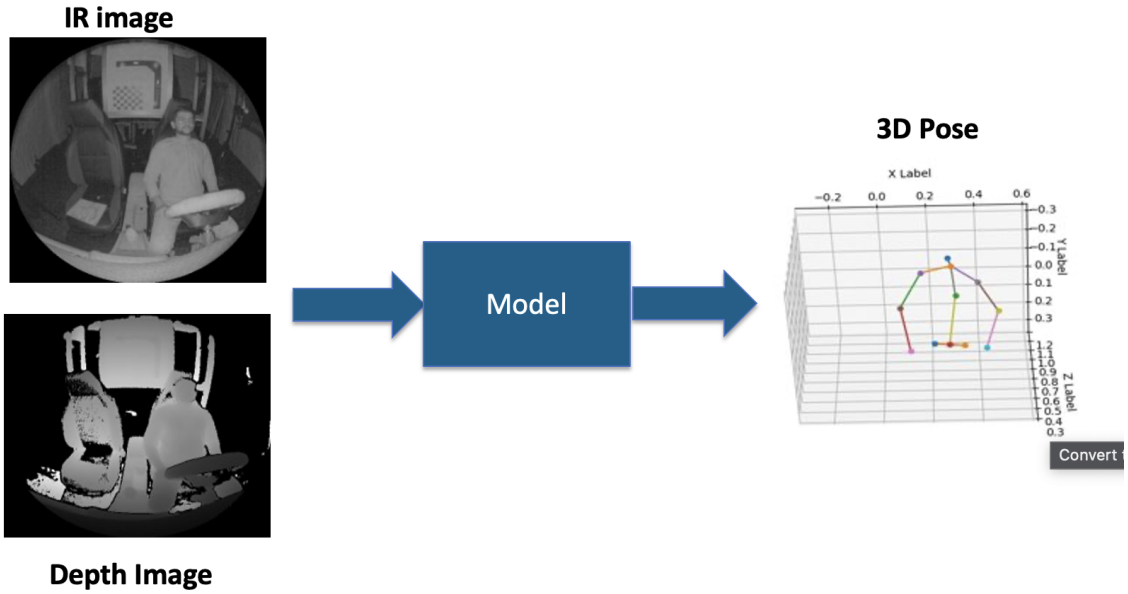


Figure 9.1.: The fusion model which takes infrared image and depth image to estimate the 3D driver pose.

The current chapter focus on the estimation 3D driver pose using volumetric representation. In addition, we explore the fusion of information derived from both modalities (IR and depth) to predict 3D poses as illustrated in Figure 9.1.

By leveraging both the compression and dimensionality reduction on volumetric heatmap, the 3D body poses can be represented in a latent space where information redundancy is reduced towards a compact representation. This is often done by minimizing the loss of data whereas keeping the important spatial nature of the representation. Particularly, the model pipeline comprises of two modules: to begin with, the pretrained Volumetric Heatmap Autoencoder is used to get a smaller/denser representation of the volumetric heatmaps. These “codes” are at that point utilized to supervise the Code Predictor, which aims at determining multiple 3D joint locations from a monocular IR and depth input. Our contributions are summarized as follows:

- An end-to-end pipeline dedicated to 3D driver pose estimation under partial visibility using Volumetric Heatmap Autoencoder.
- Introduction of a multi-modal transformer-based fusion architecture that integrates information from both IR and depth data to estimate robust 3D driver pose.

9.2 Methodology

Our proposed methodology consists of Volumetric Heatmap Autoencoder (VHA), inspired by the works of [248, 249]. The VHA compresses joint locations volumetric heatmaps into a latent code. A Code Predictor Network to ascertain these codes. During inference, the generated code is transmitted to the decoder block of the Volumetric Heatmap Autoencoder, facilitating the estimation of joint heatmaps. Subsequent post-processing steps culminate in the computation of the 3D pose.

Furthermore, we introduce a Pose Refiner Network designed to refine 3D poses by employing root relative versions of joint coordinates. This network, implemented as a feed-forward MLP, refines poses and predicts missing joints.

Our study encompasses the development of three variants. The first variant processes IR images to determine the 3D pose, the second focuses on depth images for 3D pose determination, and the third integrates information from both IR and depth sensors to predict both 3D and 2D poses. This multifaceted approach aims to provide comprehensive insights into driver pose estimation using multi-modal imaging.

The following section describes the key elements of our proposed methodology. Figure 9.2 gives an overview of our model which consist of two major components. The Volumetric Heatmap Autoencoder (VHA) and Code Predictor Network (CPN). The following subsections describe each component in detail.

9.2.1 Volumetric Heatmap Autoencoder

Volumetric heatmaps representation provides multiple advantages for 3D pose estimation regression task. Firstly, it generalizes well to different environments and camera viewpoints, as they capture the 3D structure of poses in a coordinate-independent manner. This makes them applicable across a wide range of scenarios and settings.

Secondly, their robustness to occlusions ensures reliability in scenarios where certain body parts may be obstructed or unseen, as they consider the entire volume rather than isolated keypoints. Additionally, volumetric heatmaps efficiently handle multi-person pose estimation by representing each individual's pose within its own volumetric space, facilitating simultaneous estimation for multiple individuals in a scene. Furthermore, they enable pose refinement through techniques like volumetric regression or optimization, enhancing accuracy by refining keypoint locations within the volumetric space [251].

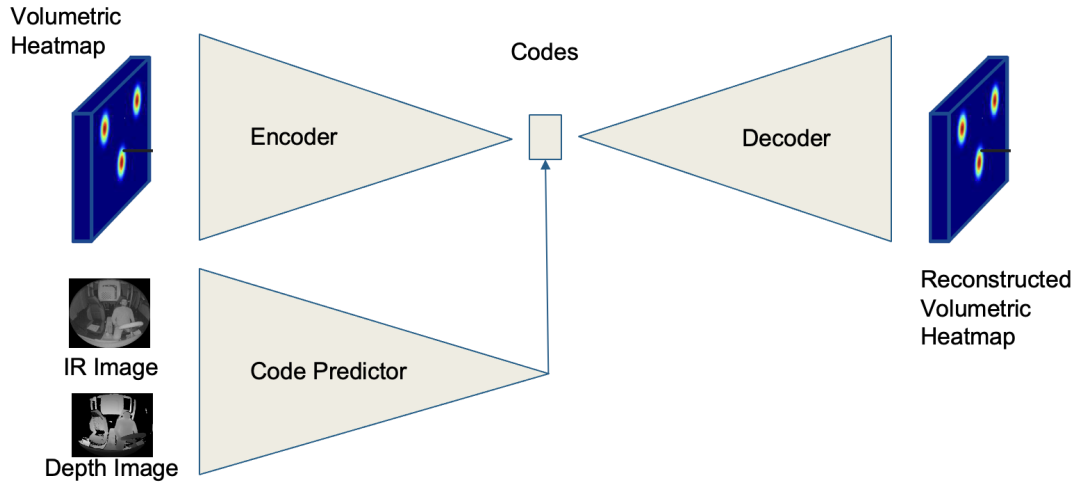


Figure 9.2.: The figures shows our own representation of the network architecture [250]. Volumetric Heatmap Autoencoder (VHA) aims to learn the compressed representation of input volumetric heatmaps that preserve the information content (top). The Code Predictor Network (CPN) aims to predict the codes obtained by the VHA(bottom)

.

Lastly, their seamless integration with deep learning architectures enables end-to-end pose estimation, leveraging powerful learning frameworks such as 3D convolutional neural networks to effectively learn features and patterns from volumetric data [252]. However, a major challenge of voxel representation is it's memory and computational demands for real-time applications. To this end, we present a compressed high-resolution volumetric heatmaps to determine joint location, using an Volumetric Heatmap Autoencoder (VHA) network to reduce the size of the volumetric representation.

Volumetric Heatmap Autoencoder (VHA) model learns the 3D pose represented as volumetric heatmaps into a dense intermediate latent code representation [248]. While considerable progress has been made in the realm of 2D heatmaps for each joint. Similarly, we model volumetric joints heatmaps for each joint. A notable limitation associated with volumetric heatmaps lies in their demanding computational and memory requirements, necessitating compromises during execution that curtail their optimal functionality.

To address these challenges without introducing quantization errors or added complexity, we utilize a Volumetric Heatmaps Autoencoder (VHA), inspired from the work of W. Luo et al [250]. Illustrated in Figure 9.3, the VHA comprises an encoder (*e*) and decoder (*d*) network. The VHA takes independently constructed volumetric heatmaps (*h*) of joints (*j*) as input, processing them through *e-c2d*. Subsequently, these heatmaps are concatenated and fed into *e-3cd*. The resulting encoded representation, $e(H)$, is then decoded by its mirrored architecture $d(e(H)) = H$. The overall structure of the model is depicted in Figure 9.3. The primary objective of the VHA is to learn a compressed representation of the input volumetric heatmaps while preserving their essential information content.

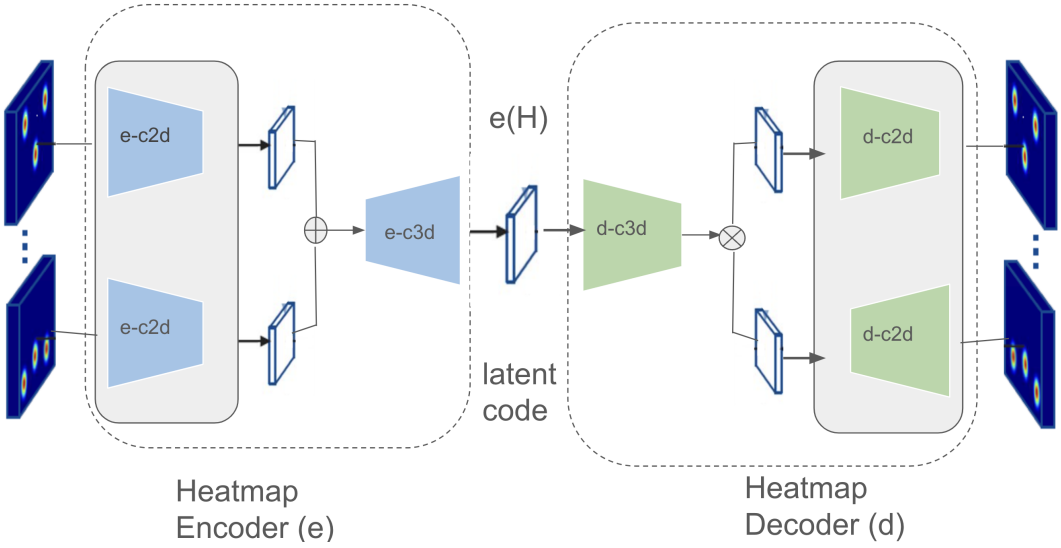


Figure 9.3.: Our representation of the architecture presented in W. Luo, et al [250]. The Volumetric Heatmap Autoencoder (VHA) aims to learn the compressed representation of input volumetric heatmaps that preserve the information content in the latent code $e(H)$.

9.2.2 Code Predictor Network

The Code Predictor Network (CPN) draws inspiration from the work of W. Tang et al. [253], which utilized a pre-trained feature extractor, Inception-v3. Throughout our study, we introduce three variations of the Code Predictor Network: (41) CPN-IR, which takes an IR image as input, (2) CPN-Depth, which takes a depth image, and (3) CPN-Fusion, which takes a concatenated IR and depth image. The following sub-section provides a concise explanation of the CPN network.

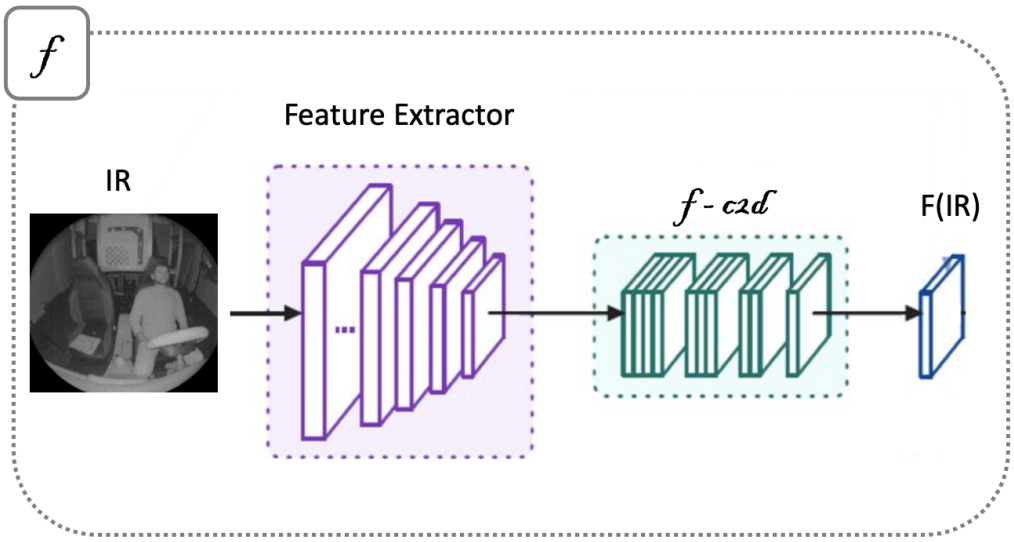


Figure 9.4.: Code Predictor Network taking an IR image as input and predicting the codes.

The Code Predictor Network (CPN), illustrated in Figure 9.4, takes IR as input and aims to predict the latent representation (codes) of Visual Human Attributes (VHA). A pre-trained feature extractor is employed to extract features, which are then passed through a series of 2D convolutional blocks to obtain the codes. The CPN is trained using the Mean Squared Error (MSE) between $f(\text{IR})$ and $e(H)$. Similarly, the second variant uses a normalized depth image as input $f(\text{IR})$ instead of an IR image maintaining the same network architecture.

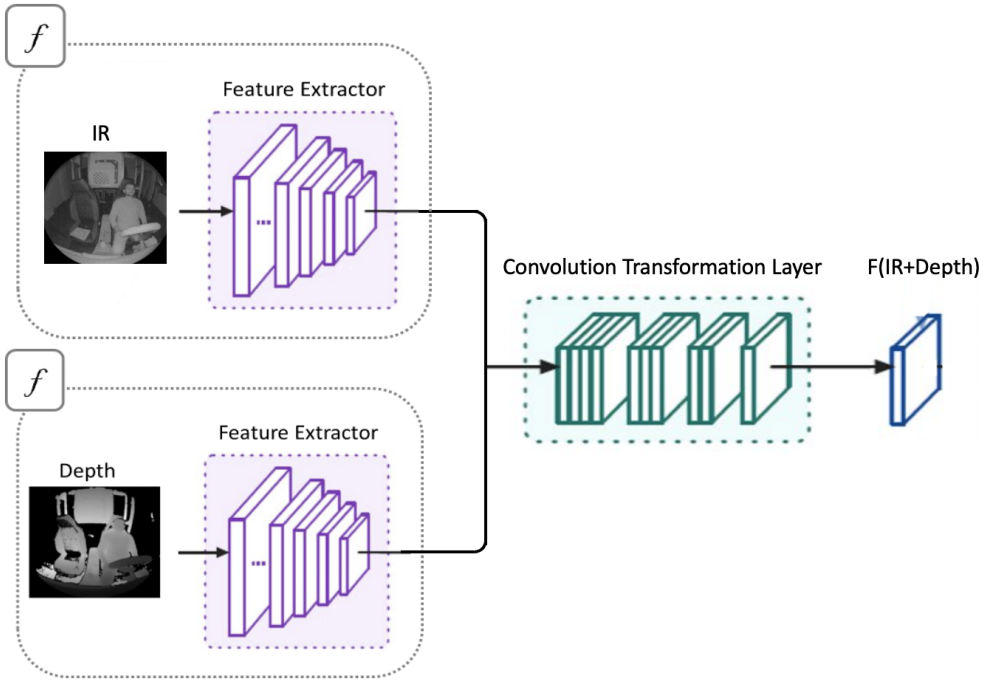


Figure 9.5.: Code Predictor Network - Fusion which takes features from IR and Depth images to predicts the codes.

CPN-Fusion: The CPN-Fusion network, depicted in Figure 9.5, takes input images *IR* and *depth* with the goal of predicting the compressed latent code representation of volumetric heatmaps. The features extracted from both IR and depth images are concatenated and passed through a convolutional vision transformer-based architectureto learn the attention between both images for predicting the codes 9.6. In this architecture, four encoders, each consisting of attention and a fully connected layer, are stacked. After each encoder layer, the 1-dimensional inputs are projected to a spatial dimension and then passed to the 2D convolution layers. The CPN is trained using Mean Squared Error (MSE) between $f(IR + Depth)$ and $e(H)$. During inference, the CPN's output is fed into decoder *d* to obtain the volumetric heatmap representation. The 3D coordinates can subsequently obtained using the pinhole camera equation.

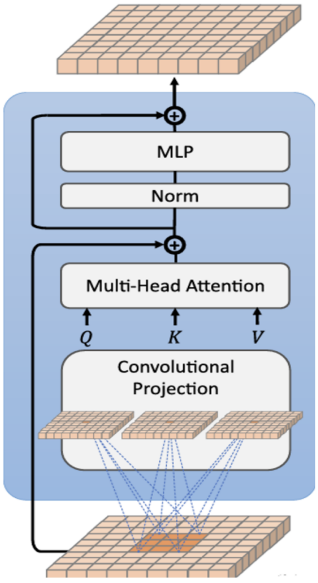


Figure 9.6.: Convolutional Vision Transformer Encoder Module [254].

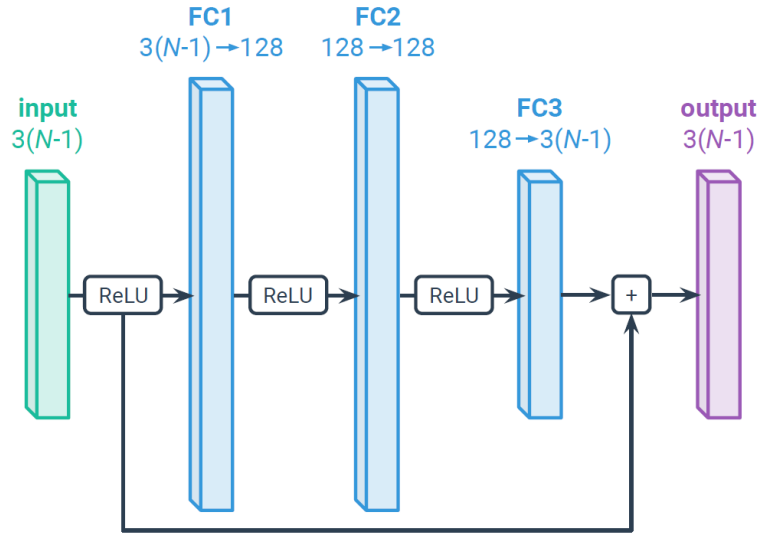


Figure 9.7.: Pose Refiner Network structure comprising three fully connected layers, with N representing the number of joints. Figure adapted from M. Fabbri et al. [248].

9.2.3 Pose Refiner Network

The Pose Refiner Network (PRN), illustrated in Figure 9.7, aims to enhance the accuracy of the 3D pose and ensure its completeness. PRN is trained using Mean Squared Error (MSE) loss, takes the normalized root-relative position of each joints as input. Also, we disable random joints during network training in order to increase robustness of the network to recover missing joints. The network architecture is straightforward, consisting of three fully connected layers with ReLU activation and a skip connection in between. Both the input and output values represent normalized root-relative representations of the same 3D posture, ranging from 0 to 1. During training, the input pose is subjected to Gaussian noise

(mean: 0m, variance: 0.08m), and some joints are arbitrarily eliminated with a probability of 0.1.

9.3 Implementation

The framework is implemented using PyTorch. The network was on trained on single NVIDIA GeForce GTX 1070 Ti GPU on Ubuntu 18.04. Adaptive moment estimate (Adam) is used for both the networks, Volumetric Heatmap Autoencoder and various Code Predictor Networks. The VHA is trained on 2000 epochs whereas CPN is trained on 5000 epochs using the MSE loss function. Both networks used 0.0001 as a learning rate, 0.7 for the decay rate, and 0.9 for the momentum, with the batch size of 1.

The input data is preprocessed prior to neural network training so as to converge more quickly. Note that the optical markers on the images are removed prior to the training in order to avoid the model sensitivity to optical markers instead of actual joint locations.

The voxelization of RGB-D volumetric space, coined by the A. Dai, et al. [255], the model considers a volumetric heatmap h, which is a 3D confidence map of size (D, W, H), where D represents the depth dimension, and H and W are represented as the height and width of the image respectively.

For a joint *j* with the 3D coordinate $u_j = (u_{1j}, u_{2j}, u_{3j})$, where u_{1j} is the distance of joint j from the camera, and u_{2j} and u_{3j} is the row and column position of joint in the image plane. The value of heatmap *h* of joint *j* at any location can be obtained by centering a fixed gaussian on u_j by the equation (9.1).

$$h_j(u) = e^{-\frac{\|u-u_j\|^2}{\sigma^2}} \tag{9.1}$$

In the context of multiple people *K* in an image, the heatmap *h* of joint *j* can be formed using the equation (9.2)

$$h_j(u) = max_K\{h_j^K(u)\} \tag{9.2}$$

The advantage of picking a volumetric heatmap representation over a 3D coordinate regression is that it represents uncertainty by extending into the third dimension, capturing uncertainty distributions throughout the volume in a discretized space. As opposed to the traditional 2D heatmaps, which are confined to an 2D image. The Figure 9.8 represents the few channels of a joint of volumetric heatmaps. The example is of the hip joint heatmap and we have taken each channel along the depth as a sequence and plotted it in the diagram. In ground truth data, joint predictions don't just estimate a distinctive location but rather a per voxel certainty, which makes it simpler for a network to learn the target position.

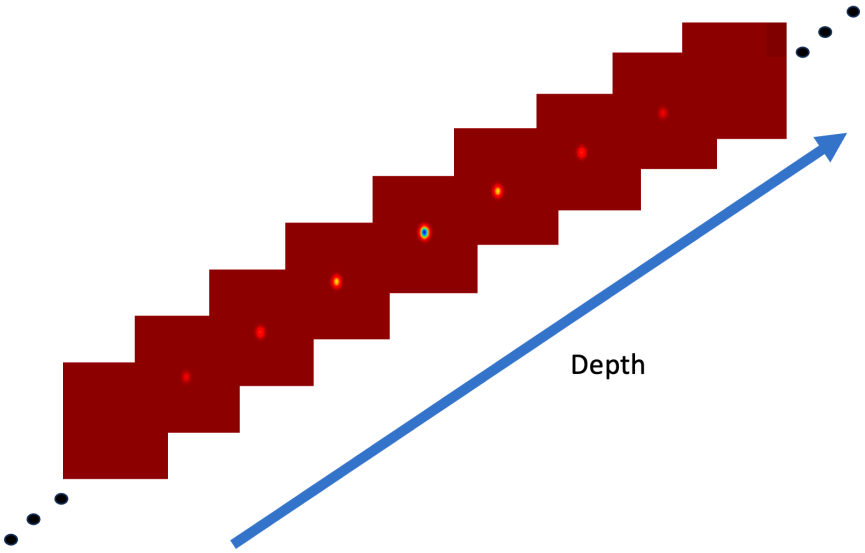


Figure 9.8.: The figure shows representation of a 3D joint volumetric heatmap, which shows the joints 2D heatmap uncertainties along the depth channels.

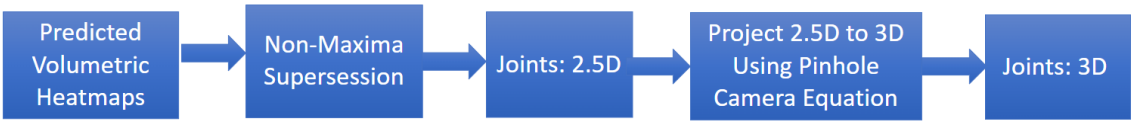


Figure 9.9.: Post processing step.

9.3.1 Post-processing

The Figure 9.9 gives an overview of the post-processing step. At inference time, the pseudo-3D coordinates of the body joints are obtained from the decoded volumetric heatmap through a local maxima search. By computing the non-maxima suppression along each volumetric heatmap to get the max location in the 2.5D coordinates. The location is decoded as a point x_{2D}, y_{2D} in the 2D depth channel at a camera distance (cam_dist). The pinhole camera equations eventually uncover the accurate three-dimensional coordinates of the detected joints if camera parameters are available using the following equation:

$$k = (-1)\sqrt{f_x^2 \cdot f_y^2 + f_x^2 \cdot c_y^2 - 2 \cdot f_x^2 \cdot c_y \cdot y_{2d} + f_x^2 \cdot y_{2d}^2 + f_y^2 \cdot c_x^2 - 2 \cdot f_y^2 \cdot c_x \cdot x_{2d} + f_y^2 \cdot x_{2d}^2} \tag{9.3}$$

$$x_{3d} = \frac{(f_y * cam_dist * c_x) - (f_y * cam_dist * x_2 d)}{k} \tag{9.4}$$

$$y_{3d} = \frac{(f_x * cam_dist * c_y) - (f_x * cam_dist * y_2 d)}{k} \tag{9.5}$$

$$z_{3d} = \frac{(f_y * cam_dist * f_x)}{k} \tag{9.6}$$

where focal length f = (f_x, f_y) and principal point (c_x, c_y)

9.4 Evaluation Results

We select the MPJPE error metric to evaluate the trained models. In the following subsections, we present qualitative and quantitative analysis on each driving datasets.

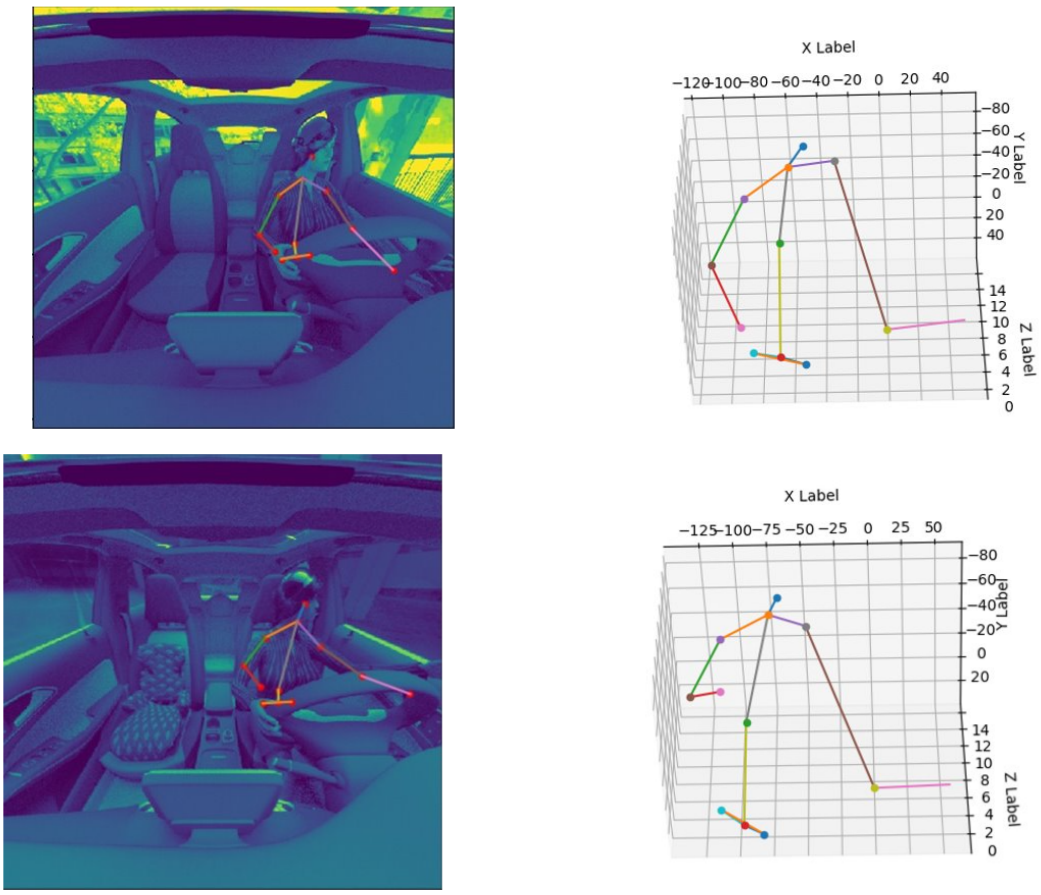


Figure 9.10.: The figure represents the output of our Code Predictor Model 9.2.2 on some of the example images of TiCaM dataset.

TiCaM

The TiCaM dataset comprises synthetic infrared (IR), depth, and RGB images capturing the front seat of a car. First, we generate volumetric heatmaps and trained the Volumetric Heatmaps Autoencoder (VHA) similar to procedure described in section 9.3. The volumetric heatmap is represented with size $(12 \times 128 \times 128 \times 128)$ where the dimesions represent 12 joints, height, width, and depth respectively. This constructed heatmap is again based on 2D joint keypoint annotations. The z-axis is encoded using depth map. Due to lack of

3D keypoint labels, depth values lead to inaccuracies, especially in lower joint obscured by car body parts like the steering. Training of the VHA was performed using using MSE loss, and the resulting codes were saved upon completion of the training process.

Subsequently, the Code Predictor Network (CPN) is trained, taking input from depth and infrared (IR) images sized at 512×512. The saved heatmap codes served as the ground truth for the provided input images. During inference, Code Predictor Network take both images ad input to produce the code. The code is then forwarded with the trained decoder segment of the VHA to generate the joint's volumetric heatmaps.

In the post-processing step, non-maximum suppression was applied to obtain joint locations in 2.5D. Due to the absence of camera intrinsic and extrinsic parameters, the root-relative joint location was calculated, with the midpoint of the hip joint serving as the root, facilitating the representation of the pose in 3D.

The figure 9.10 shows the qualitative results on the TiCaM test set. The model demonstrates robust 2D pose to illumination changes and car background. However, the 3D prediction suffered on joints where real depth data was unavailable such as left arm of the driver. The left arm was occluded by car steering wheel, resulting in incorrect or missing depth data.

SVIRO

The SVIRO dataset encompasses synthetic infrared (IR), depth, and RGB images capturing the interior rear seat of cars. Our training procedure to train VHA aligns with the methodology outlined in the preceding section 9.3. The dataset annotations solely provides 2D joint keypoints. To formulate the Volumetric Heatmaps, we encode joint locations by extracting 2D joint coordinates in image space. The camera distance is derived from the depth value corresponding to the joint's location in the depth map. It is important to acknowledge that depth locations for certain joints may be misleading due to occlusion by the car dashboard, leading us to exclude such joints during the construction of volumetric heatmaps.

The latent code generated using VHA are saved to serves as ground truth for the Code Predictor Network (CPN). In our training, we confine the focus to the CPN-Fusion model trained for the SVIRO dataset. Further, we regress pose for individual occupants by cropping the person using a fixed bounding box.

During the post-processing phase, in the absence of camera intrinsic and extrinsic parameters, we compute the root-relative joint locations. The hip right joint serves as the reference point for establishing the 3D pose visualization.

Figure 9.11 illustrates shows the qualitative results using CPN-Fusion on the SVIRO test set images. The first column shows the overlayed 2D output, while the second column displays the root-relative 3D joint coordinates in 2.5D space. The first row shows the networks robustness to the illumination changes in the input. The second row shows the network predicting the right shoulder despite being occluded. However, the third row shows

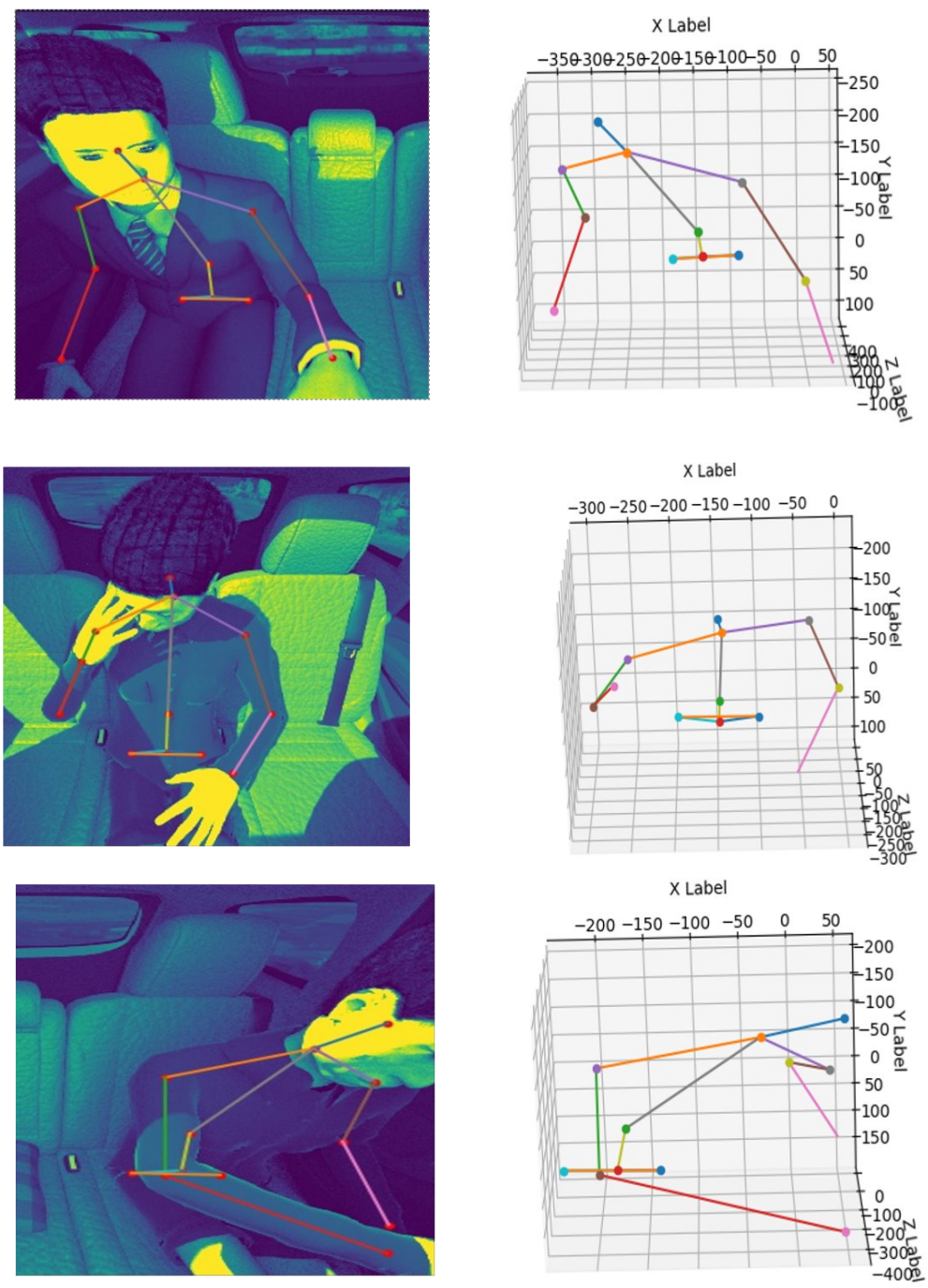


Figure 9.11.: Illustration of 3D pose outputs obtained through CPN-Fusion 9.2.2 applied to selected images from the SVIRO dataset.

that the model struggles to accurately estimate the depth values, such as the left elbow and shoulder of the driver. This limitation arises from the absence of real 3D coordinates for occluded joints. Thus, the depth value from depth map for occluded joints (due to other objects or self-occlusion) can be misleading.

DREICAP-3D

The DREICAP-3D dataset has been partitioned into three distinct subsets: a training set comprising 65%, a validation set with 25%, and a test set accounting for 5%. Initially, we generate 14 volumetric heatmaps corresponding to the 14 joints, each with dimensions $(128 \times 128 \times 128)$. The height and width undergo a downsampling of a factor of 4. We perform similar training procedure to train VHA and (CPN) networks.

	Head	Neck	Chest	Pelvis	Shoulders	Elbows	Wrists	Hips	Mean
IR	0.024	0.017	0.017	0.037	0.019	0.026	0.037	0.038	0.026
Depth	0.022	0.019	0.017	0.029	0.016	0.022	0.056	0.029	0.026
Fusion	**0.019**	**0.015**	**0.014**	**0.021**	**0.013**	**0.020**	**0.031**	**0.019**	**0.019**

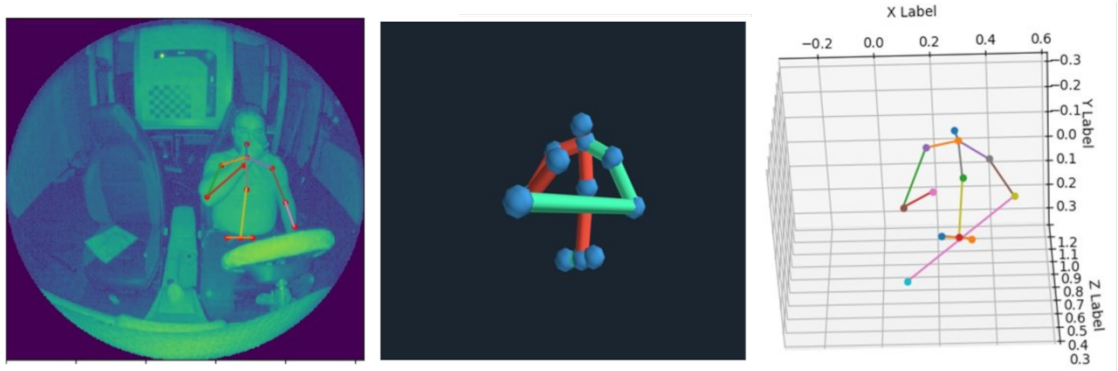
Table 9.1.: This table shows the MPJPE error in meters for each joint individually on the validation set of the DREICAP-3D dataset in various scenarios of our Code Predictor Network.

Prior to initiating the training of the Code Predictor Network, the optical marker were removed from both the infrared (IR) and depth imagee raw images. This pre-processing step holds vital significance, as the network has the potential to learn the optical marker as distinctive features in the images, potentially causing the model to become sensitive to the absence of markers in the input data.

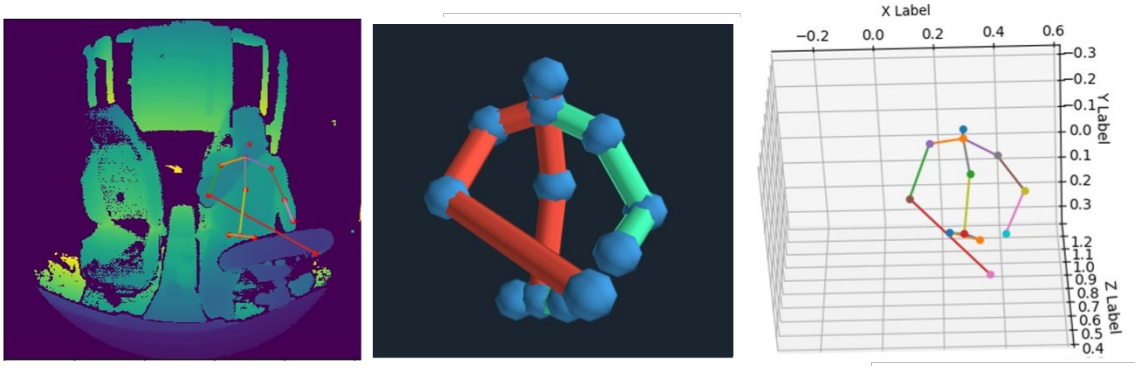
During the inference phase, the CPN takes the input image and predicts the code, which is then fed into the decoder portion of the VHA to obtain the volumetric heatmaps. Subsequently, these heatmaps undergo non-maximum suppression to extract 2D joint coordinates, which are then projected to 3D using the pinhole camera model. Refer to the post-processing section (9.9) for more details.

Table 9.1 showcases the Mean Per Joint Position Error (MPJPE) for each joint individually on the DREICAP-3D test dataset. The Table shows results for three trained variants of Code Predictor Network - CPN-IR, CPN-depth and CPN-Fusion. The table reveals that CPN-Fusion network, which integrates information from both raw fish-eye view IR and depth images outperforms other networks. The CPN-Fusion achieves overall accuracy of 0.019 meters (19 mm) on the test set.

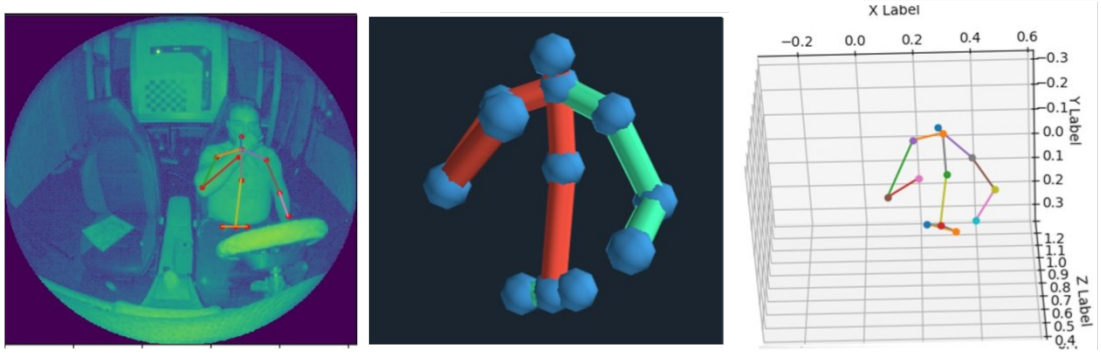
The Figure 9.12 shows 2D pose and 3D pose qualitative results for all varints of thes Code Predictor Networks. The CPN-IR which takes the IR image as input performs well for detecting the 2D joints location but fails to detect depth value accurately for the left wrist. The CPN-Depth (second row) which takes the depth image as input performs fails to detect the right wrist in image space. Finally, the CPN-Fusion detects both the wrist joints accurately both in the 2D and 3D space third row).



(a) CPN-IR



(b) CPN-Depth



(c) CPN-Fusion

Figure 9.12.: The figure represents the 2D/3D output from various Code Predictor Networks(CPN) on test sample taken from the DREICAP-3D dataset. The first column represents the 2D output. The second and third column represents the 3D output using Mlab and Pyplot respectively.

9.5 Ablation Study

Two ablation studies are conducted within the proposed framework. Initially, we delve into the impact of voxelization size and gaussian kernel on overall accuracy. Subsequently, we analyze the improvements resulting from the pose refinement network.

Hyper Parameters	Models					
	VHA-1	VHA-2	VHA-3	VHA-4	VHA-5	VHA-6
Downscale Height Downscale Width	$\frac{H}{8}$ $\frac{W}{8}$	$\frac{H}{8}$ $\frac{W}{8}$	$\frac{H}{4}$ $\frac{W}{4}$	$\frac{H}{4}$ $\frac{W}{4}$	$\frac{H}{4}$ $\frac{H}{4}$	$\frac{H}{4}$ $\frac{H}{4}$
No. Depth Channel	64	128	64	128	128	128
Sigma	4	4	4	2	4	8
F1 score	0.801	0.823	0.894	0.926	0.946	**0.986**

Table 9.2.: The table compares various VHA models trained with different voxelization factor on DREICAP-3D test set by F1 score. F1 score considers a predicted joint to be true positive if the distance from the corresponding ground truth joint is less than 1 voxel space.

9.5.1 Influence of Varied Voxelization Dimensions

In this section, we have detailed about the experiments performed on Volumetric Heatmap Autoencoder(VHA) to find its optimal downsampling factor. In the experiments, we train VHA on different downsampling factors as shown in table 9.2. The downsampling is performed in also performed in depth dimension. The experiments also investigates the impact of different sigma size while representing gaussian heatmap across depth channel. The model with 128 depth channels with heatmap represented in 8 depth channels performed best with F1 score of 0.986.

9.5.2 Improvement through Pose Refiner Network

The Pose Refiner Network further refines the 3D poses for the detected 12 joints. Figure 9.13 shows the comparison between the various Code Predictor Networks on raw images with and without using Pose Refiner Network. We compute the F1 score for a predicted joint, where it is considered “true positive” if the distance from the corresponding ground truth joint is less than a given threshold.

It is demonstrated that the Pose Refiner Network improves the efficiency of our various Code Predictor Networks. The qualitative result presented in Figure 9.14 shows that the Pose Refiner Network accurately the 3D joint coordinate. Figure 9.14a shows that the Pose Refiner Networks corrects the left wrist joint position, where it also detected the missing joint shown in Figure 9.14b.

9.6 Conclusion

The chapter presents an innovative approach that takes infrared and depth images to predict 3D driver pose. The proposed approach utilizes compressed volumetric heatmaps as target, which is generated using 3D ground truth pose data. Moreover, the volumetric heatmap autoencoder enables the increased performance and robustness of the network

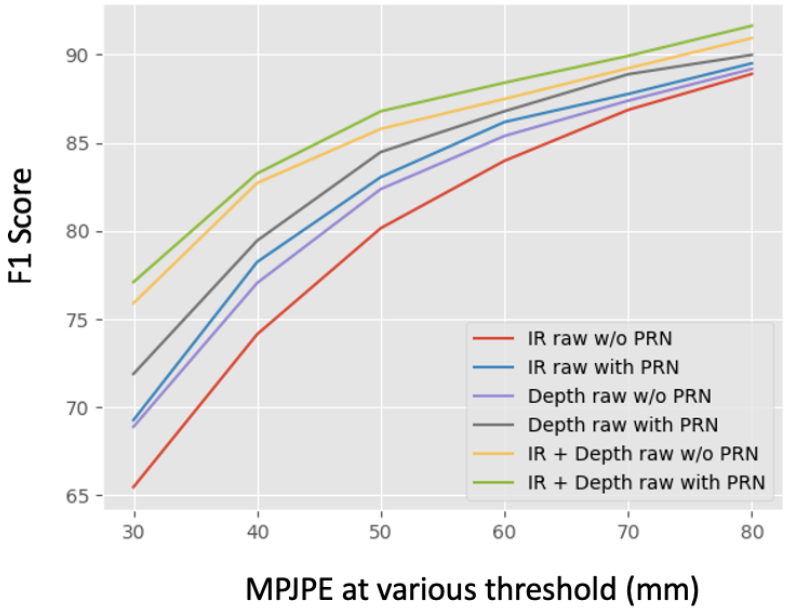
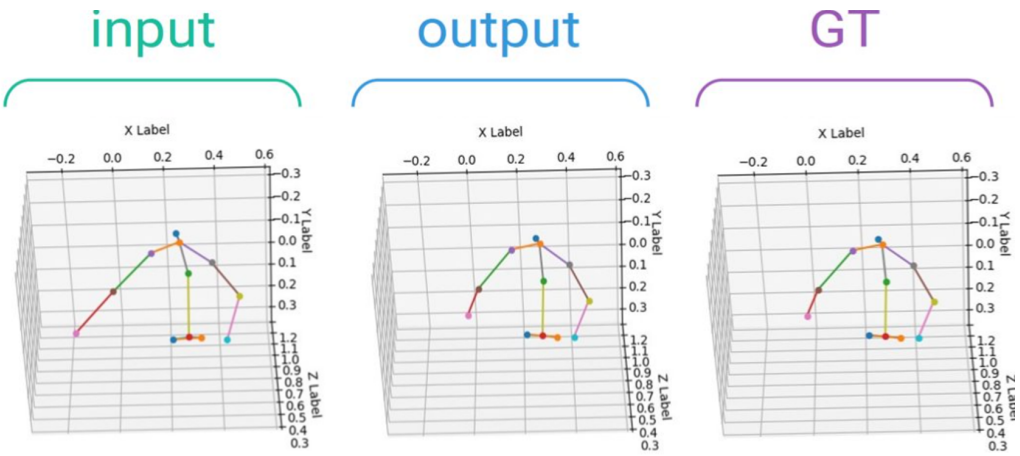


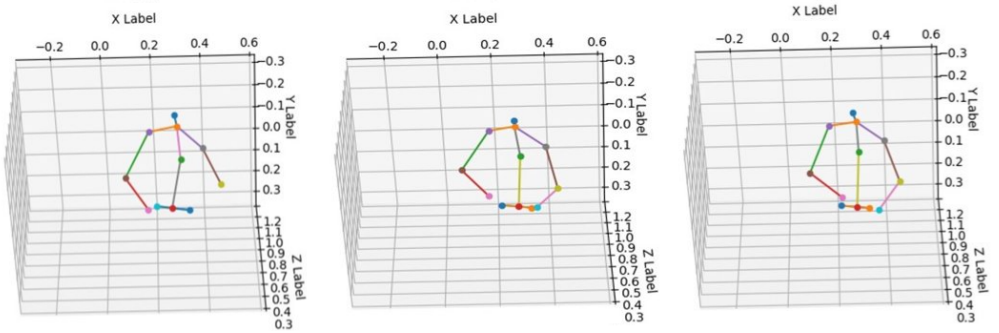
Figure 9.13.: Figure shows the comparison between the various Code Predictor Network on raw images with and without using Pose Refiner Network. F1 indicates that a predicted joint is considered "true positive" if the distance from the corresponding ground truth joint is less than a given threshold.

by using the compressed representation of sparse and extremely high dimensional output space. The presented work also compares Code Predictor Network that takes IR, depth, and fusion of both input modalities. This network allows predicting the code, which is further decoded using decoder part of the Volumetric Heatmap Auntoencoder to obtain the joint position's volumetric heatmaps. Also, Pose Refiner Network further refines the 3D poses or detects the missing joints which further improves the efficiency and performance of our model.

Our research demonstrates robust performance in accurately detecting the 3D driver pose across various datasets. Notably, our model, utilizing the sensor fusion stage of infrared (IR) and depth, exhibits superior performance compared to its single-modality Code Predictor Network (CPN) counterparts. It's important to note that our approach has been exclusively tested on a single human, specifically the driver in the scenario. However, the methodology can be readily extended to encompass other passenger seats within the car.



(a) Pose Refiner Network corrects the left wrist joint position.



(b) Pose Refiner Network detects the missing joint.

Figure 9.14.: Qualitative results of Pose Refiner model on the DREICAP-3D test dataset.

Driver Pose Estimation using Point Clouds

10

> *"I, a universe of atoms, an atom in the universe."*
>
> — **Richard Feynman**
> American physicist

The point cloud representation play a pivotal role in computer vision for understanding complex 3D environments. The point cloud data are often used for surface reconstruction, generating detailed 3D models of objects or scenes from sparse points. Primarily, the points within a point cloud lack a predefined order or structure. The points in the point cloud are not entirely independent; rather, they maintain relationships with neighboring points, constituting an intricate interdependence.

10.1 Overview

The recent advancements such as PointNet [256] and PointNet++ [257] have enabled convolutional neural networks to directly process point clouds, eliminating the need for the intermediary step of converting them into 3D voxels. This development has substantially simplified the study of point clouds utilizing deep learning methodologies [258, 259]. There by generating a surge of interest and the release of numerous publicly available datasets [260, 261, 262].

10.2 Motivation

The choice of point clouds as input modality for car driver pose estimation can justified on the following grounds listed below:

- **Varying Lighting Conditions:** Unlike traditional RGB-based methods, point clouds are less affected by changes in lighting conditions inside the car [263]. They provide precise depth data by ensuring robustness regardless of whether the car is parked in bright sunlight or dimly lit environments, which is essential for reliable pose detection.
- **Privacy Preservation**: Point cloud-based pose detection methods can preserve the privacy of occupants by avoiding the need to capture detailed RGB images or videos.

Instead, only the geometric structure of the scene is utilized for pose estimation, reducing privacy concerns associated with image-based approaches [264].

- **Occlusion Handling**: In a confined space like a car cabin, occupants may frequently occlude each other or themselves due to various objects present inside the car. Point clouds inherently capture the spatial relationships between objects and occupants, allowing for better handling of occlusions and more accurate pose estimation [265].
- **Real-time Sensor Fusion:** Cars are equipped with various sensors, including LiDAR, cameras, and radar. Modern sensor technologies, such as LiDAR or depth-sensing cameras, can generate point clouds in real-time with low latency. Point cloud data can be seamlessly integrated with information from these sensors to enhance pose detection accuracy and robustness. Sensor fusion techniques can further leverage the complementary strengths of different sensors to overcome individual sensor limitations and improve overall performance [266]. This enables real-time pose detection inside the car, which is crucial for applications like driver monitoring systems.
- **Safety Applications:** Accurate pose detection inside the car is crucial for safety applications such as occupant monitoring, driver attention assessment, and airbag deployment optimization. Point cloud-based approaches provide precise depth data which is necessary for the reliability of safety-critical tasks.

Overall, the cloud offer distinct advantages for pose detection tasks inside a car. Thus, the point clouds make a compelling choice as additional input modality for applications in automotive environments. Our contributions in the current chapter are succinctly summarized as follows:

- We present a PointConvNet [267] architecture incorporating random sampling in conjunction with a Local Feature Aggregation (LFA) module [268] to estimate 3D driver keypoint pose estimation using point clouds.
- We also present network architecture to regressing 3D joint rotation regression for in-cabin car driver pose using point clouds.

10.3 Proposed Methodology

The choice of using point cloud as input data, necessitates direct feature extraction on point cloud to obtain better 3D structural information. However, this often result in high computational cost. A common solution is to downsample the point cloud prior to the processing. Then deal with the key points obtained by down sampling instead of the original point cloud, so as to achieve the purpose of reducing the amount of calculation. To this end, we look in the following subsections two sampling methods: Random Sampling and Farthest Point Sampling.

The entire preprocessing workflow is illustrated in Figure 10.1.

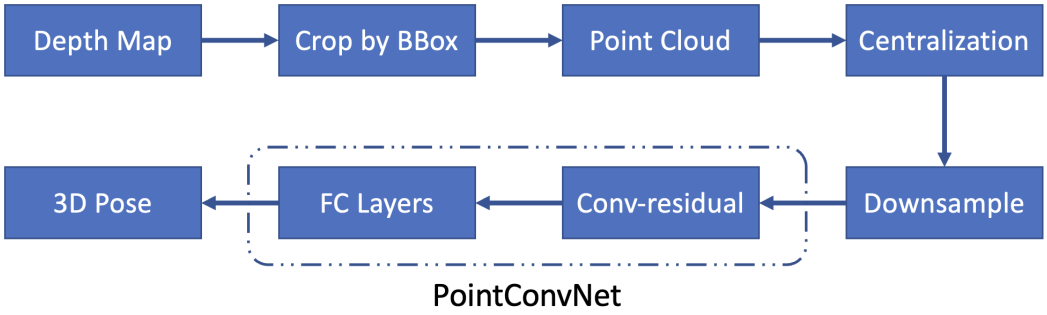


Figure 10.1.: The overview of proposed method

The preprocessing of point clouds is an essential task. The initial step involves the removal of background and noise points, centralization and followed by downsampling the point cloud. The selection of an appropriate downsampling technique presents a critical consideration, particularly when dealing with large-scale point clouds. This choice must take into account computational efficiency and other pertinent factors. In the next subsection, we breifly describe different downsampling methods and compare their benefits.

10.3.1 Sampling Techniques

The downsampling methods include Random Sampling(RS), Farthest Point Sampling, Inverse Density Importance Sampling (IDIS) [269], Generator-based Sampling (GS) [270], Continuous Relaxation based Sampling (CRS) [271, 272] and Policy Gradient based Sampling (PGS) [273]. Among them, RS has the lowest computational complexity of $O(1)$ and can be applied to large-scale point clouds; IDIS has $O(N)$ but is easily affected by outliers; FPS has $O(N^2)$; GS has higher computational complexity than FPS; CRS requires very large GPU memory and PGS has difficulty in converging the network when applied to large-scale point clouds. The memory consumption and computation time required for different sampling methods vary with the size of the point cloud. RS has significant advantages over other methods in terms of both computation time and memory consumption [268]. Nevertheless, we restrict our experiments to Randam Sampling and Farthest Point Sampling methods as explained in the following subsections.

Random Sampling

The principle of random sampling is very simple, and it is also the most common down-sampling method. First specify the number of downsampled points K, and then randomly select these K points from all N points in the entire input, where each point has the same probability of being selected. The calculation amount of random sampling is only related to the number of selected points K, and has nothing to do with the scale of the original point cloud N. So the computational complexity of random sampling is O(1). The characteristic

of random sampling is that it can control the number of downsampled point clouds, but the randomness is too large, and the important points may be eliminated.

Farthest Point Sampling

Farthest point sampling (FPS) is a very popular sampling algorithm [274]. In each sampling, the point that is farthest away from the (K-1) point set obtained in the previous sampling is selected. FPS can ensure that the sampled points have better coverage, and can control the number of sampling points. But the computational complexity of FPS is $O(N^2)$, and the amount of computation is squarely related to the number of points in the original point cloud. The pseudo code of FPS is shown in Algorithm 5.

Algorithm 5 Farthest Point Sampling

Input: Original point cloud P with n points
Output: A sampled point cloud S with m points

$S \leftarrow P0$
for all points in P-S **do**
    compute the distance between point to P0
    $L \leftarrow dist$
end for
$S \leftarrow P1$ (P1 is the point corresponding to the maximum value in L)
for all points in P-S **do**
    compute the distance between point to P1
    if dist<L[i] **then**
        L[i] = dist
    end if
end for
$S \leftarrow P2$ (P2 is the point corresponding to the maximum value in L)
Repeat the operation until m points are sampled

There are two issues to be considered, one is the initial point selection and the other is the distance metric used. We can randomly choose a point as the initial point, then the results of sampling point may be different every time. We can also choose a fixed point, such as the point farthest from the center of gravity of the point cloud, so that the result is the same every time. For point clouds, euclidean distance is generally used as the distance metric. And for triangular meshes, the geodesic distance is a better choice for distance metric.

10.3.2 Network Structure

When selecting the point cloud as the input data of the neural network, generally choose a sampling method with a certain number of sampling points, such as the random sampling and the farthest point sampling. And we use the random sampling with local feature aggregation module, proposed by Qingyong Hu et al. [268]. It aggregates better and

learn important local features, and thereby effectively alleviate the problem of losing useful information of point clouds caused by random sampling. We adopt a PointConvNet structure inspired by Zhijie Yao et al. [267], which adds residual connection to the traditional CNN structure. The complete network structure is shown in Figure 10.2.

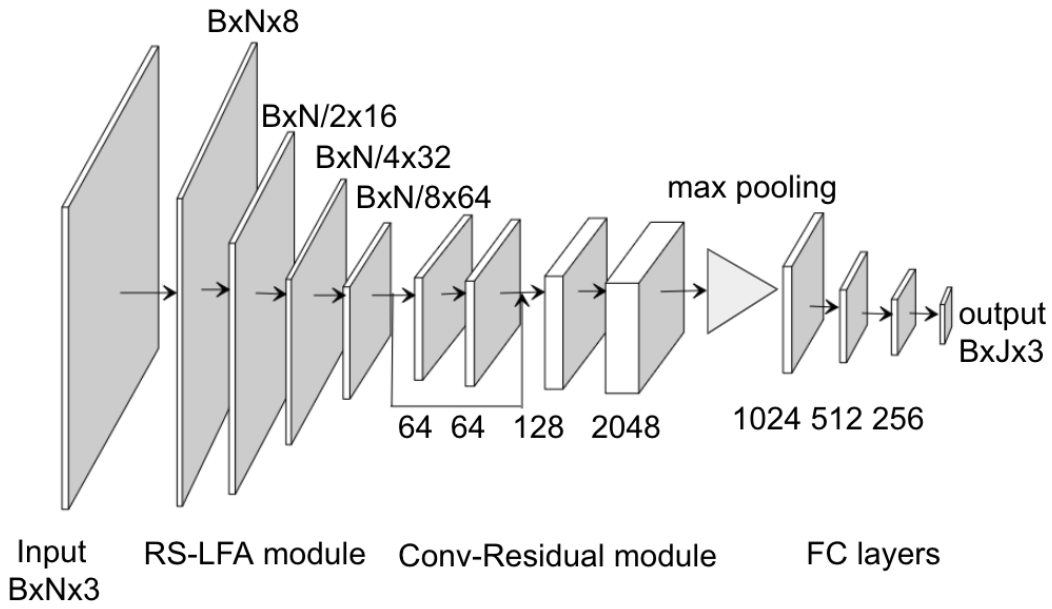


Figure 10.2.: The figure shows the PointConvNet's network structure with local feature aggregation (LFA) module

The input data are the point clouds with 3D coordinates. Firstly, we use a fully connected layer to increase the features to 8. Then followed by three random downsampling layers with local feature aggregation(LFA) modules. They can further increase the number of features while reducing the number of points. Next, convolution module with residual connection consisting of 4, 1×1 convolution layers are used to produce 2048 feature vector. The max pooling layer reduce the features, and before going through four fully connected(FC) layers resulting in 3D coordinates of J joints.

RS-LFA Module: The LFA module consists of local spatial encoding (LSE) [268] module, attentive pooling (AP) [268] module and dilated residual block [268]. In LSE module, we fist need to compute the 16 neighbor points by KNN algorithm, then combine the center point 3D coordinate, neighbor point 3D coordinate, relative coordinate and Euclidean distance together to compute the new neighbor point's feature. Finally combine the original neighbor point's feature and the new feature. The concatenation features are as twice large as the original ones. The AP module uses an attention mechanism to automatically learn and aggregate useful information from the concatenation features. It uses a fully connected network and a softmax function to learn the score of the concatenation features, and then uses the score as the weight to multiply the features. Finally, go through a shared multi-layer perceptron to adjust the size of features. The dilated residual block consists

of two LSE modules, two AP modules and one residual connection. It can double the size of the input feature by increasing the receptive field of each point. The RS module can reduce the number of points to half the original. So after three RS-LFA modules, the number of points change from 16384 to 2048 and the feature size become 64.

Conv-residual Module: Similar to the pointnet [256] and the joint 2d-3d network [267], we choose to use four convolution layers and a residual connection between the first two convolution layers with channel 64. Finally, a max pooling layer is used to aggregate features to ensure that the extracted global features of length 2048 do not change with the order of points.

Fully Connected Layers: We use a four-layer fully connected network to calculate the 3D coordinates of joints. And to prevent overfitting, we adopt the drop out trick before the second layer and set the probability to 0.5. When using datasets with different number of joints as training data, we need to modify the output size of the last fully connected layer.

10.4 Experimental Setup

We first generate point clouds from depth maps. So they are in the depth camera's coordinate system. In order to align the point clouds and 3D joint coordinates, we need to transform the 3D joint coordinates from world coordinate system to depth camera's coordinate system.

Kinect camera provides the rotation and translation from RGB camera coordinate system to IR camera coordinate system. And the camera extrinsic provides the rotation and translation from RGB to world coordinate system. Through Equation 10.1 and Equation 10.2, we can calculate the rotation and translation from IR to world coordinate system.

$$\mathrm{R}_{ir2world} = \mathrm{R}^{\mathrm{T}}_{rgb2ir} \cdot \mathrm{R}_{rgb2world} \tag{10.1}$$

$$\mathrm{t}_{ir2world} = (\mathrm{R}^{\mathrm{T}}_{rgb2ir} \cdot \mathrm{t}^{\mathrm{T}}_{rgb2world})^{\mathrm{T}} - (\mathrm{R}^{\mathrm{T}}_{rgb2ir} \cdot \mathrm{t}^{\mathrm{T}}_{rgb2ir})^{\mathrm{T}} \tag{10.2}$$

$$\begin{bmatrix} \mathrm{X}_c \\ \mathrm{Y}_c \\ \mathrm{Z}_c \end{bmatrix} = \mathrm{R}_{ir2world} \cdot \begin{bmatrix} \mathrm{X}_w \\ \mathrm{Y}_w \\ \mathrm{Z}_w \end{bmatrix} + \mathrm{t}_{ir2world} \tag{10.3}$$

According to the Equation 10.3, we can convert the point coordinates in the world coordinate system to the 3D coordinates in the depth camera coordinate system.

In order to make the data as clean as possible, we can use bounding box to remove background points. Then convert the 2D image coordinates of the points into 3D coordinates according to the Equation 10.4.

$$\begin{bmatrix} \mathsf{X} \\ \mathsf{Y} \\ \mathsf{Z} \end{bmatrix} \begin{bmatrix} \mathsf{f}_x & 0 & \mathsf{c}_x \\ 0 & \mathsf{f}_y & \mathsf{c}_y \\ 0 & 0 & 1 \end{bmatrix} = Z \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} \tag{10.4}$$

Then we need to centralize the point cloud to remove the effect of translation. That is, move the point cloud to the origin. Finally, stored the processed point cloud and the corresponding ground truth of joint coordinates in HDF5 format.

Training Procedure: In all experiments, we use Adam optimizer with the momentum 0.9 and the learning rate as 0.001 and decay rate is 0.7. The batch size is 32 and a total of 50 epochs are trained. The entire network is implemented by the TensorFlow framework and trained on Ubuntu 18.04 with one NVIDIA GeForce GTX 1070 Ti GPU.

10.5 Evaluation Results

We use mean average precision (mAP) evaluation metric to measure the accuracy of the estimated 3D joint keypoints. The calculation methods are shown in Equation [10.5-10.7]. To calculate the average precision, we need to first calculate the euclidean distance between the predicted 3D joint coordinates and the ground truth. Then we choose an appropriate distance as the threshold, and the prediction results with the distance less than the threshold are considered to be correct. Typically, the threshold is set to 10cm. Finally, we calculate the percentage of correct keypoints using mAP.

$$dist(a,b) = \sqrt{(\mathsf{a}_x - \mathsf{b}_x)^2 + (\mathsf{a}_y - \mathsf{b}_y)^2 + (\mathsf{a}_z - \mathsf{b}_z)^2} \tag{10.5}$$

$$AP(a,b) = \begin{cases} 1 & dist(a,b) <= 10cm \\ 0 & otherwise \end{cases} \tag{10.6}$$

$$mAP = \frac{\sum_{i=1}^{N} AP(\mathsf{p}_i, \mathsf{gt}_i)}{N} \tag{10.7}$$

10.5.1 ITOP Dataset

We perform our initial experiments on the ITOP dataset [275]. The network structure we used is the same as mentioned in the section 10.3.2. We replace the RS-LFA module with FPS sampling method for corresponding experiment. We only use data marked as valid as training set and test set. The training set consists of 18K frames and the test set has about 4.8K frames. The test results are as shown in Figure 10.3.

From the line chart, we can see that the farthest point sampling can be slightly more effectively with higher mAP, especially for joints with low mAP such as hands. Since the point cloud after removing the background points in ITOP dataset resulting in sparse data in few thousand points, the RS-LFA sampling method has limited accuracy on ITOP dataset [275]. The different downsampling methods can have different effectiveness of the model. Therefore, it is crucial to choose the appropriate downsampling method based on the dataset.

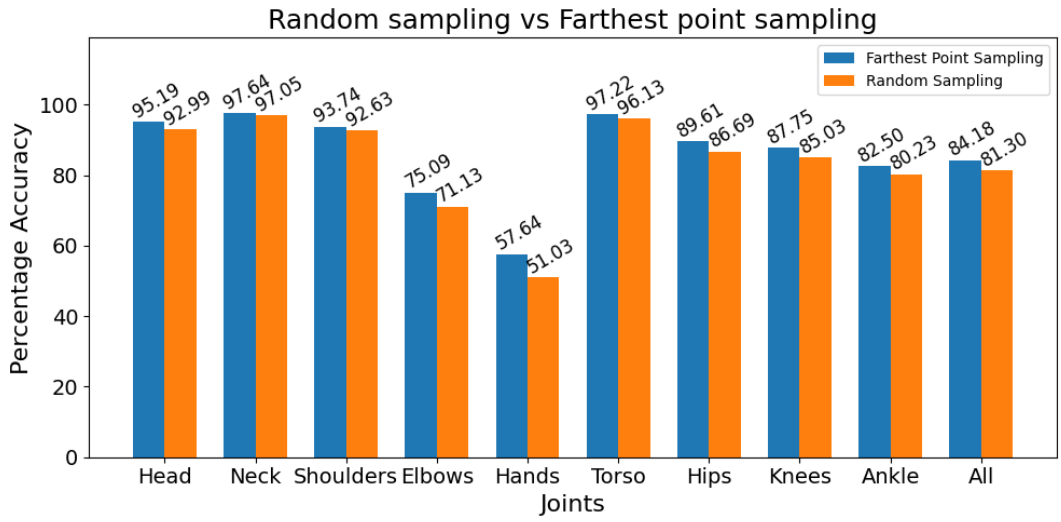
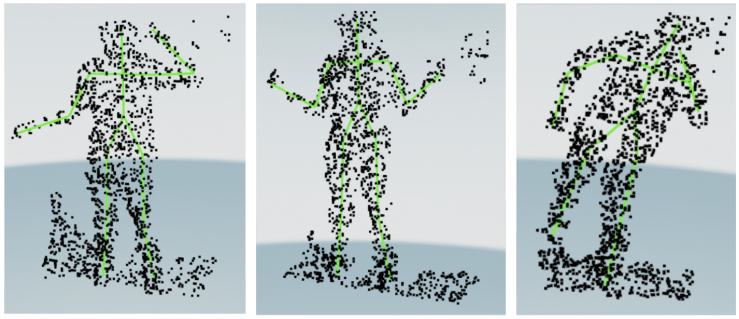


Figure 10.3.: Line chart of each joint's mAP for random sampling and farthest point sampling on the ITOP dataset

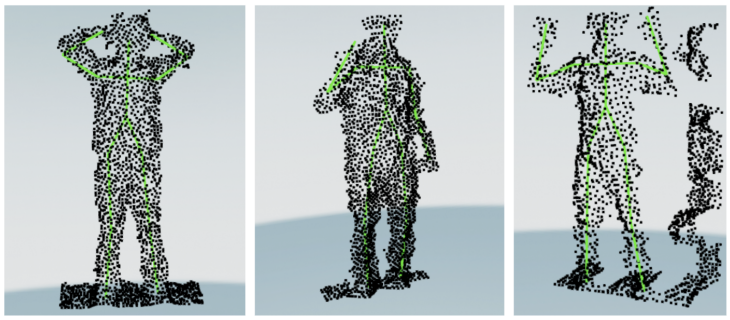
The Figure 10.4 shows that the arms are represented using more sparse points compared with the rest part of the human body. While in random sampling (RS), only a small number of points are selected for the arm part, and a large number of points are located on the body torso. While in the farthest point sampling (FPS), the selected points are more on the edge of the human body, so the network learn more global features. Although the number of sampling points are same, FPS results in more uniform points with respect to RS.

10.5.2 DREICAP-3D Dataset

We test the effect of random sampling and RS-LFA sampling on the DREICAP-3D dataset. We use person01 and person02 as the training data, person04, person06, person07 and person10 as the test data. We use cleaned groundtruth points generated using SOMA model. We select 19 joint keypoints out of 55 skeletal joint keypoints. In addition, we also compare our prediction with ground truth generated using Kinect azure body tracking [276]. The results of test data are shown in Table 10.1.



(a)



(b)

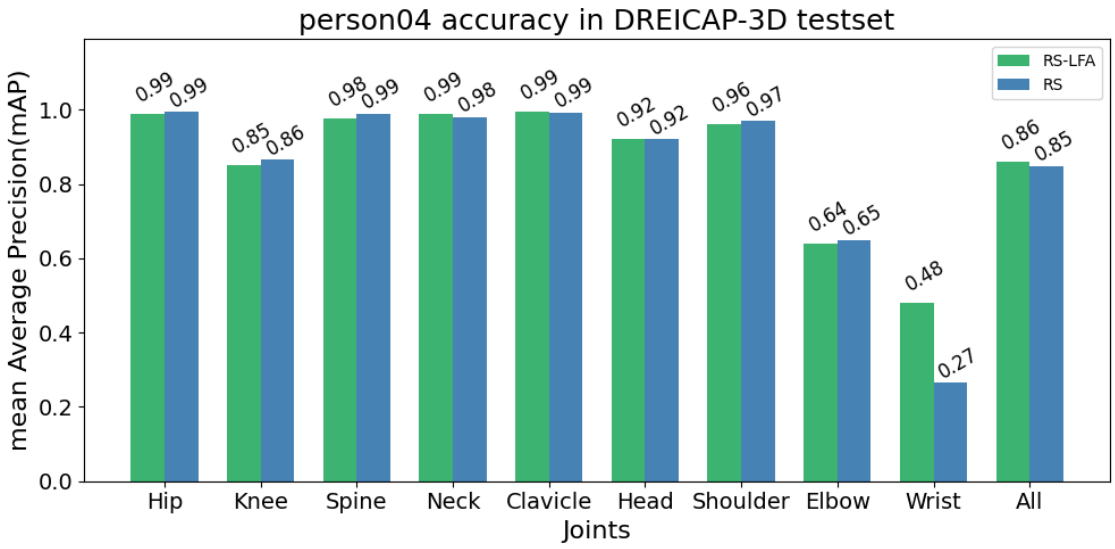
Figure 10.4.: All images are point clouds sampled with 2048 points (a) is the point cloud sampled by random sampling, (b) is the point cloud sampled by FPS

	RS	RS-LFA(SOMA)	RS-LFA(Kinect)
person04	0.858351	0.869263	0.921726
person06	0.823446	0.895770	0.936525
person07	0.931587	0.947287	0.945993
person10	0.901205	0.920180	0.953602
Mean	0.878647	0.908125	0.939461

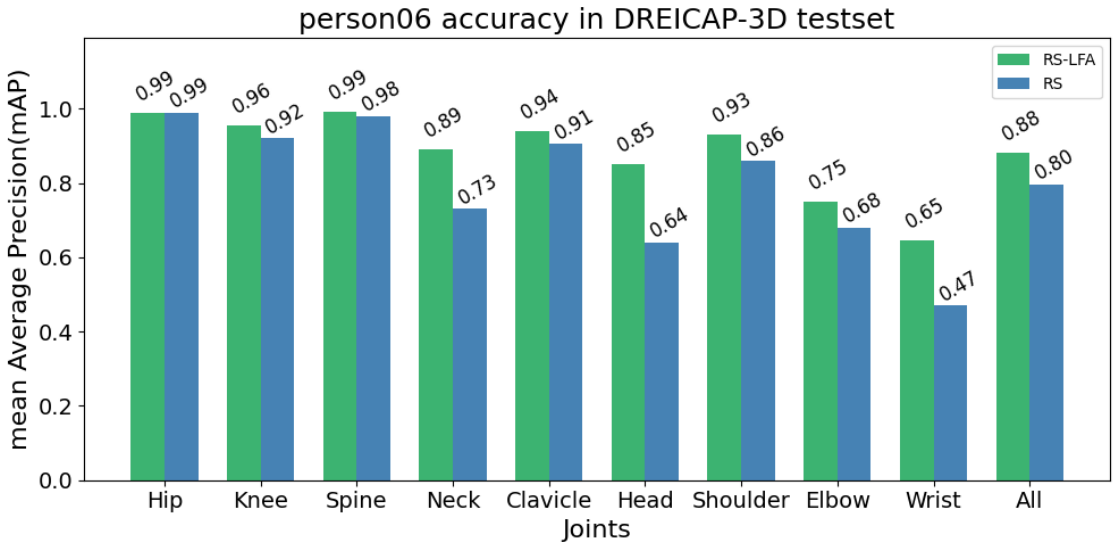
Table 10.1.: The Table provides a comprehensive overview of the model's performance across different sampling methods and ground truth generation techniques. The rows show mAP of each person in the test-set for random sampling (RS), RS-LFA(SOMA), and RS-LFA(Kinect) on the DREICAP-3D dataset. The last row presents the mean mAP values.

From the Table 10.1, RS-LFA shows effective improvement in the model prediction results. However, the degree of improvement varies for different people, which may be related to the recorded data quality being different for each individual.

For instance, the mAP value of each join keypoints for the participants person04 and person06 is as shown in the Figure 10.5a and Figure 10.5b. From the Figure 10.5a, we can see that the mAP of RS-LFA at spine1, left knee and right elbow are significantly smaller than RS, while at left elbow, left and right wrist are significantly better than RS. But as can be seen from Figure 10.5b, the mAP has improved in almost all joints. Therefore, we



(a) *person04* test sequences



(b) *person06* test sequences

Figure 10.5.: Bar chart shows mAP for each joint using random sampling(RS) and with local feature aggregation (RS-LFA) on the DREICAP-3D test dataset; the accuracy plot of person04(top) and person06(bottom) participant test sequences shows accuracy over different joints.

believe that RS-LFA is useful for improving the model prediction results, but the effect is different depending on the quality of the input data.

To compare whether the groundtruth generated by different methods have similar experimental results, we try to use the skeleton automatically generated by the kinect camera as the groundtruth. The Kinect skeleton contains only 15 points while the SOMA skeleton has a total of 19 points due to the poor quality of the knee's groundtruth in the Kinect skeleton and the lack of the jaw and a point on the spine. The compare results are shown in Table 10.1 and the compare results for each joint is shown in the Figure 10.6.

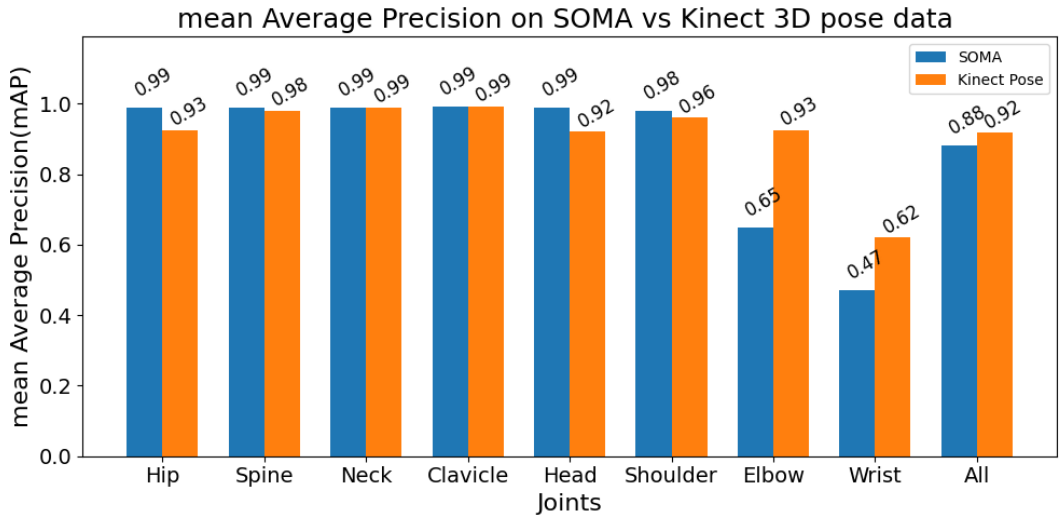
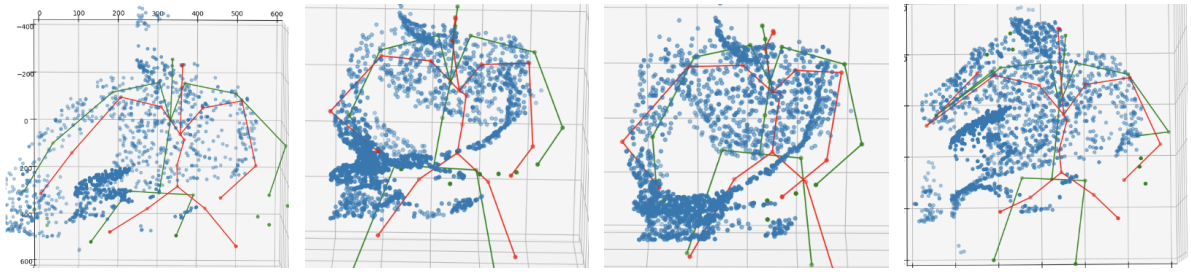


Figure 10.6.: The mAP of each joint for groundtruth generated by SOMA and Kinect 3D pose ground truth data in DREICAP-3D testset dataset(person04)



(a) (b) (c) (d)

Figure 10.7.: Comparison plots of the groundtruth generated using two different methods: SOMA(red) and Kinect(green)

We randomly selected four frames from the test data for visualization, as shown in Figure 10.7. The red skeleton is generated by the SOMA model, and the green skeleton is generated by the Kinect camera. In order to show the skeleton clearly, 2048 points were randomly selected for display in figures.

From the Figure 10.7, we can observe that the model trained on SOMA skeleton had limited accuracy on the points near the spine, while the model trained on Kinect skeleton data struggled to provide accurate keypoint position near hips and knees. According to the shape of the point cloud we can see that for the determination of the arm position is challenging.

In all previous experiments, we defined the threshold as 10 cm, because this value is the most commonly used threshold in the field of human pose estimation. We also tried to evaluate the prediction results under different thresholds. The Figure 10.8 shows the accuracy results for both models at various thresholds.

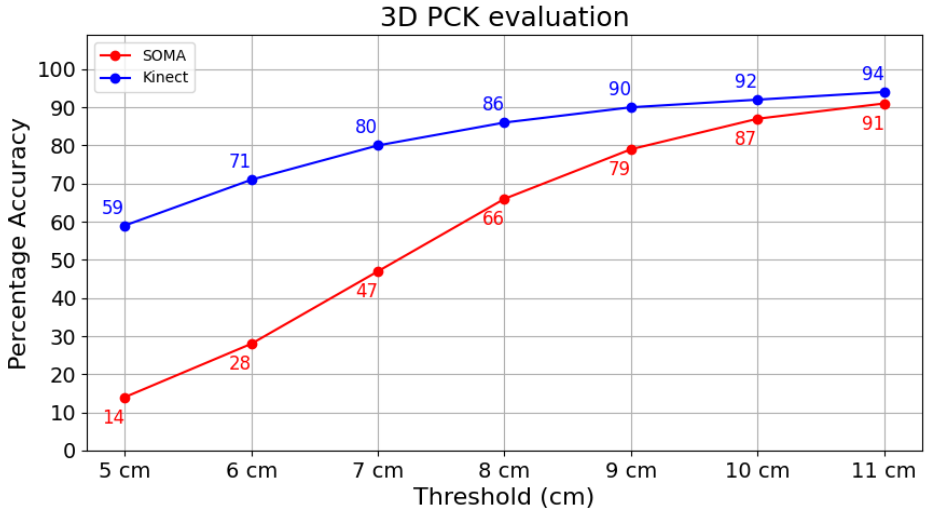


Figure 10.8.: The plot shows the 3D PCK evaluation results at different thresholds for SOMA method and Kinect method generated 3D pose ground truth data

	3D KeyPoints RS(SOMA)	3D Joint Rotations(SOMA)
person04	0.858351	0.798023
person06	0.823446	0.855395
person07	0.931587	0.877562
person10	0.901205	0.858194

Table 10.2.: The Table shows the mAP of each person in the test set for 3D KeyPoints using RS(SOMA) and 3D Joint Rotations(SOMA) over DREICAP-3D dataset

3D Joint Rotations: Although the quality of the SOMA skeleton needs to be further improved, generating the joint coordinates with the SOMA model allows us to obtain the local joint rotation. We further investigate the regression of pose joint angles, and then transform them into joint coordinates to calculate the mAP. The results are shown in Table 10.2. From the Table 10.2, we can see that the accuracy using mAPs metrics for joint rotations are inferior than the directly predicted joint coordinates. We believe this might be due to error accumulation during the forward kinematics to obtain 3D coordinates. It might occur also due to network structure inability to learn 3D joint rotations.

10.6 Conclusion

In this chapter, we present a robust driver pose estimation and the development of a PointConvNet with a random sampling and local feature aggregation (RS-LFA) module. Our work centers on the creation of a novel dataset DREICAP-3D tailored for driver pose estimation and the development of a PointConvNet with a random sampling and local feature aggregation (RS-LFA) module. To this end, we have harnessed the DREICAP-3D recorded using DFKI In-car cabin simulator to simulate a real driving environment.

The network architecture we employ integrates a random sampling module with the local feature aggregation (RS-LFA) module, a fundamental CNN structure equipped with a residual connection, and fully connected layers. This network structure is optimized for the direct processing of point clouds and leverages an efficient sampling method. Notably, the RS-LFA module enhances the aggregation and preservation of vital local features, effectively mitigating the loss of valuable information in point clouds often associated with random sampling.

Our experimental results using the ITOP dataset underscore the significance of a suitable downsampling method in enhancing the accuracy of human pose estimation. Moreover, our experiments on our dataset reveal that the LFA module substantially improves prediction accuracy. Nonetheless, a comparison between the ground truth 3D joint coordinates generated by the SOMA model and the Kinect camera reveals discrepancies in the arm's positioning. Consequently, one of our future directions is to enhance the dataset's 3D joint coordinates' quality, potentially through measures such as manual removal of frames that fall short of quality standards.

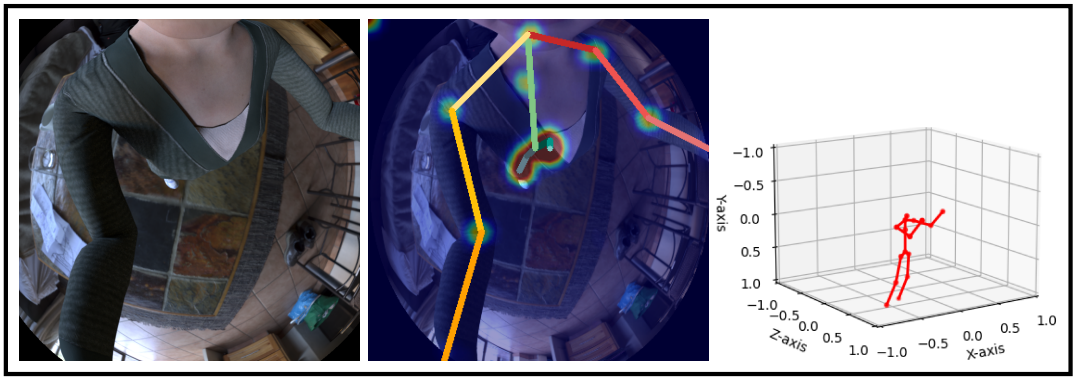
In our exploration, we also investigated an alternative approach involving the prediction of joint rotations as opposed to direct 3D keypoint regression. However, our experimental findings revealed no discernible improvement in performance. The limited accuracy in some of the joints warrants exploration of different network architectures as the sub-optimal results could also be due to the network architecture capacity to learn features predicting joint rotations. The lower results can also be attributed to error accumulation during the forward kinematics process. Consequently, further investigation is warranted, as refining this approach represents a crucial aspect of future research aimed at achieving higher accuracy for the overall body pose.

Acknowledgments

I would like to acknowledge the support and discussions from Dr. Bruno Mirbach and Dr. Jason Rambach, which significantly enriched the content and ideas presented here. I extend my gratitude to Qinzhuan Qian, Pascal Schneider, Xue Yang and Shubham Gupta for their contributions toward the driver pose dataset curation, preprocessing and codebase. The research work for funded under the project VIZTA under European Union's Horizon 2020 research and innovation program with grant agreement No 826600.

Part IV

Egocentric view: Self Occlusion and Partial Visibility



A visual representation of head-mounted fisheye-camera providing a first-person (Egocentric) view(left) from a synthetic dataset [71] with its corresponding 2D Pose(middle) and 3D Pose(right)

Overview

Overview

Overview

In this part, our focus shifts to the egocentric perspective of the camera. Our objective is to tackle the challenges associated with self-occlusion and partial visibility in first-person view images. Chapter 11 begins with an introduction to the fundamental principles crucial for understanding egocentric pose. Subsequently, a review of relevant research on 3D pose estimation from an egocentric viewpoint is provided. Following this, commonly used datasets with egocentric pose estimation are discussed. Next, the proposed methodology used in this study is explained in detail. The chapter ends with a thorough look at the experimental results of pose and its associated uncertainties.

Related publications

[1] Muhammad Umer Javaid. “Probabilistic 3D Egocentric Pose Estimation”. Master’s thesis. Informatik, RPTU Kaiserslautern-Landau, Aug. 2022.

[2] Arpit Lekhra. “Towards 3D EgoCentric Human Pose Estimation”. Master’s thesis. Informatik, RPTU Kaiserslautern-Landau, Mar. 2023.

Egocentric Human Pose Estimation with Head Mounted Camera

11

> *"It is not the mountain we conquer but ourselves."*
>
> — **Sir Edmund Hillary**
> Mount Everest Mountaineer

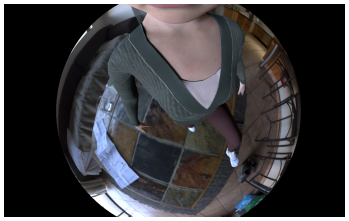
In the present chapter, we shift focus from an external(exo-centric) view to egocentric vision. While we have been previously engaged in understanding exo-centric perspectives, our current emphasis lies on comprehending human pose in egocentric vision. The recent surge of augmented reality (AR), virtual reality (VR), and mixed reality (MR), among other industries, is a result of technological advancements. In many applications, it's crucial to understand a person's actions, the wearer's body pose, remains inherently largely invisible to the camera. This shift acknowledges the importance of grasping the activities of individuals wearing cameras, a pivotal aspect in egocentric vision research. Existing efforts have concentrated on estimating the pose of hands and arms when they are within the camera's view. Nonetheless, this approach provides an incomplete perspective of the overall body posture and precludes any pose estimation in numerous frames where hands are not visible, which is common in various daily life activities.

11.1 Overview

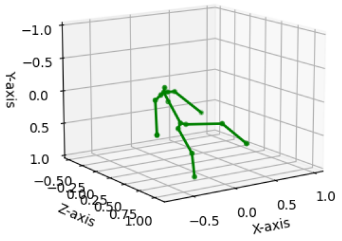
The goal of our research is to estimate robust 3D human pose estimation in ego-centric view. The experimental setup consists of a person wearing a Head Mounted Display(HMD) attached with a monocular downward-looking fish eye camera. This setup renders the environment from the very specific view of the user causing high perspective distortion, self-occlusion and partial visibility of joints. Figure 11.1, demonstrates the EgoCentric Pose setup. Estimating 2D or 3D pose from images collected from this first-person viewpoint is significantly more challenging than compared to the common external perspective. The solution is intended not only to deliver accurate pose estimates but also to incorporate awareness of uncertainties arising from factors such as self-occlusion or partial visibility of joints.



(a) Hardware Setup



(b) EgoCentric image



(c) 3D joint positions

Figure 11.1.: The general hardware configuration for EgoCentric Pose Estimation is shown in figure(a) [71]. The figure (b) illustrates how the EgoCentric image has a distorted view of the fisheye camera [71]. The figure (c) displays the subject's 3D joint position.

We present a novel feature-based, triple-branch network for ego-pose estimation. The backbone encodes the feature channels at a lower resolution, making it computationally efficient [277, 278, 279]. The Waterfall Atrous Spatial Pooling (WASP) [280] module, which combines the cascaded Atrous Convolution technique with the wider field of view acquired from the parallel arrangement of the Atrous Spatial Pyramid Pooling (ASPP) [281] module. The results of our studies show that the suggested methodology is effective at increasing the precision and efficacy of pose estimation.

Further, 6D rotation representation is selected for training to have a a continuous embedding in SO3 space [127]. Additionally, in some applications, such as forward or inverse kinematics necessitates the networks output to be orthogonal matrices. This demands incorporating an additional orthogonalization procedure during the network training. In this study, we incorporate the orthogonalization procedure to have a stable 6D joint rotations regression outputs [282].

The following are the key contributions of this thesis.

- A novel End-to-End pipeline for a features-based, triple-branch network for 3D Ego Pose estimation that delivers state-of-the-art performance, particularly when occlusions are high.
- The most efficient continuous rotations representation space for learning joint rotations in order to achieve more accurate 3D Ego poses and their complex geometric articulations.
- Proposes novel loss functions for learning angles that address the issue of rotation matrix orthogonality.
- Analyzing the model's uncertainty to ascertain its level of confidence in its 2D/3D pose and rotation predictions.

11.2 Related Works

In the current section, we delve into the related literature addressing the challenge of 3D ego pose estimation from single-view RGB images captured through mounted cameras on the body. Cha et al. [283] introduced an RNN-based technique for body pose estimation, employing two pinhole cameras fixed on the head. Xu et al. [284] propose a single-frame-based egocentric motion capture method using a single wide-view fisheye camera. Tome et al. [71] utilized a similar setup to record egocentric posture, employing an auto-encoder network that measures the degree of uncertainty in estimated heatmaps. Zhang et al. [285] recommended an automatic calibration module to further mitigate picture distortion effects. Hwang et al. [286] affixed an ultra-wide fisheye camera to the user's chest, simultaneously estimating body pose, camera rotation, and head pose.

Despite the recent surge in attention toward capturing people from an egocentric camera position for activity recognition, a significant portion of existing approaches predominantly focus on detecting upper body movements, such as hands, arms, or torso [287, 288, 289]. Capturing comprehensive 3D body motion using head-mounted cameras proves to be considerably more challenging. Several head-mounted capture systems rely on RGB-D input, primarily reconstructing hand, arm, and torso motions [290, 291]. Jiang and Grauman [292] attempted to reconstruct the entire body position from footage captured by a camera worn on the chest, relying on ego-motion calculations from the observed scene; however, their estimates exhibit inaccuracies and a high level of uncertainty.

Pioneering a different approach, Rhodin et al. [293] introduced a the first full-body capture through a helmet-mounted stereo fish-eye camera pair. Placing the cameras approximately 25 cm away from the user's head using telescoping sticks, this configuration, while somewhat inconvenient for the user, offered the advantage of capturing expansive field-of-view images encompassing the majority of the body [294].

11.3 Methodology

In this section, the methodology employed in this study is presented. Initially, we outline the baseline solution utilized for the extraction of image features, and subsequently, we introduce an auto-encoder network inspired by the work of [71]. We delve into the intricacies of our implementation, describing the process of generating feature maps using Wasp [295] and atrous spatial pooling [296]. Subsequently, we elucidate the architecture of our first-branch designed for 2D pose estimation. Proceeding further, we expound upon the second branch dedicated to 3D pose estimation and its influence on 2D poses. The final branch addresses angle rotations. A critical focus of the thesis pertains to devising a methods for learning joint rotation, particularly in scenarios where joints are occluded or inadequately posed.

To this end, we experiment two distinct network architectures: the XR-Ego network by Tome et al. [71] and our EgoPoseNet network architecture. Note that due to the

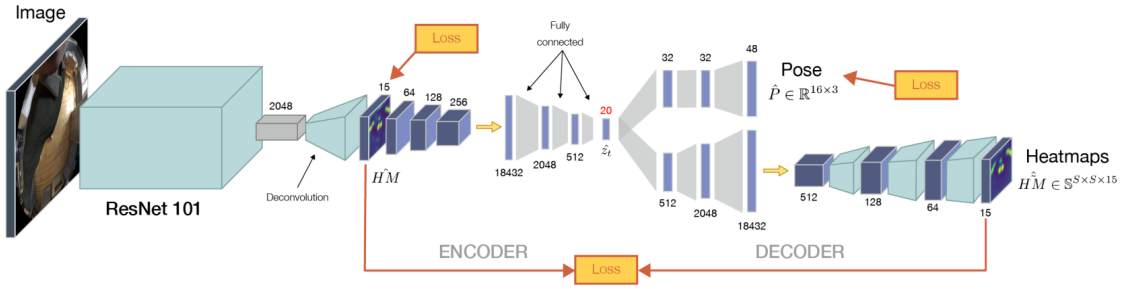


Figure 11.2.: The XR-Ego Network Architecture as presented by [71]. The lower branch is dedicated to reconstructing 2D heatmaps, while the upper branch is responsible for regressing 3D joint positions.

unavailability of either the code or predictions of xR-egopose [72] in the public domain, we rely on our re-implementation of xR-egopose network, providing a detailed account of our specific implementation.We delineate the diverse modules and training procedures employed in training each model in the following subsections.

11.3.1 xR-Ego Network

The architectural framework of the xR-Ego network comprises a two-stage, two-branch autoencoder as shown in Figure11.2. In the initial stage, the first branch is dedicated to the generation of 2D heatmaps, while the second branch is responsible for producing 3D joint positions. The first stage runs the input image through the ResNet backbone, followed by a deconvolution layer that generates 2D heatmaps denoted as H_i. The final pooling layer and fully connected layer of ResNet-101 are substituted with two deconvolution layers and a sigmoid function, serving the purpose of generating 2D heatmaps. Subsequently, the produced heatmap undergoes encoding at a reduced dimensionality via an autoencoder configuration, yielding the reconstructed 2D heatmap denoted as H_r.

In the second stage, the 2D heatmaps H_i are employed as inputs to predict 3D coordinates P_{pred}. This process involves the regression of 3D coordinates based on the information encapsulated in the generated 2D heatmaps.

Backbone

The backbone of the network comprises Resnet-101 with a deconvolution layer. Given the input image $\mathbf{I} \in \mathbb{R}^{368\times368\times3}$, the backbone generates $H_i \in \mathbb{R}^{48\times48\times15}$. Xavier [297] initialization is utilized to initialize the weights. The loss function L_{2D} is defined as the mean squared error between the ground truth 2D heatmaps H_g and the predicted heatmaps H_i, expressed in Equation 11.2.

$$\mathbf{L_{2D}} = \text{mse}(H_g, H_i) \tag{11.1}$$

Heatmap Autoencoder

The Encoder takes H_i as input, predicted by the backbone branch, and produces lower-dimension embeddings denoted as z of size 20. The encoder comprises three convolution layers with batch normalization, resulting in 512×6×6 channels. Dropout is applied with a probability of 0.5. Following the flattening of channels, a fully connected layer is applied, reducing the dimension from 512×6×6 to a latent space of 20. The decoder consists of two branches: one to regress the 3D pose from z, and another to reconstruct the input heatmaps.

The decoder takes the latent representation generated by the encoder as input and reconstructs 2D heatmaps $H_r \in \mathbb{R}^{48\times48\times15}$.

$$\mathbf{L_r} = \text{mse}(H_i, H_r) \tag{11.2}$$

3D Keypoint Pose Decoder

The 3D Keypoint Pose Decoder network branch processes the latent code z generated by the encoder, decoding to generate 3D joint positions $P_{pred} \in \mathbb{R}^{16\times3}$. The comprehensive autoencoder loss, denoted as L_{AE} (Equation 11.3), encompasses the l_2 loss between P_{pred} and the ground truth P_{gt}, the pose loss R, and the reconstruction loss L_r. The pose loss R comprises the cosine similarity loss θ and the l_1 loss L. The weights λ used in the loss function are set to $\lambda_{hm} = 10^{-3}, \lambda_p = 10^{-1}, \lambda_\theta = -10^{-2}, \lambda_L = 0.5$.

$$\begin{aligned}
\mathbf{L_{AE}} =& \lambda_p(\|P_{pred} - P_{gt}\|^2 + \mathbf{R}(P_{pred}, P_{gt})) + \lambda_{hm}\mathbf{L}_r \\
\mathbf{R}(P_{pred}, P_{gt}) =& \lambda_\theta\theta(P_{pred}, P_{gt}) + \lambda_L\mathbf{L}(P_{pred}, P_{gt}) \\
\theta(P_{pred}, P_{gt}) =& \sum_l^L \frac{P_{pred_l} \cdot P_{gt_l}}{\|P_{pred_l} \cdot P_{gt_l}\|} \\
\mathbf{L}(P_{pred}, P_{gt}) =& \sum_l^L \|P_{pred_l} - P_{gt_l}\|
\end{aligned} \tag{11.3}$$

Training

In the initial stage, the backbone and heatmap autoencoder undergo separate training. These two different modules are put together and the autoencoder is finetuned along with the backbone L_{2D} loss and the reconstruction loss L_r. Finally, the trained encoder of the autoencoder, denoted as L_{AE}, is used to train the 3D keypoint pose decoder branch.

11.3.2 EgoPoseNet Network

The contemporary literature on ego-centric pose estimation, a common limitation observed in recent networks is the absence of predictions for joint rotation angles. Upon scrutinizing

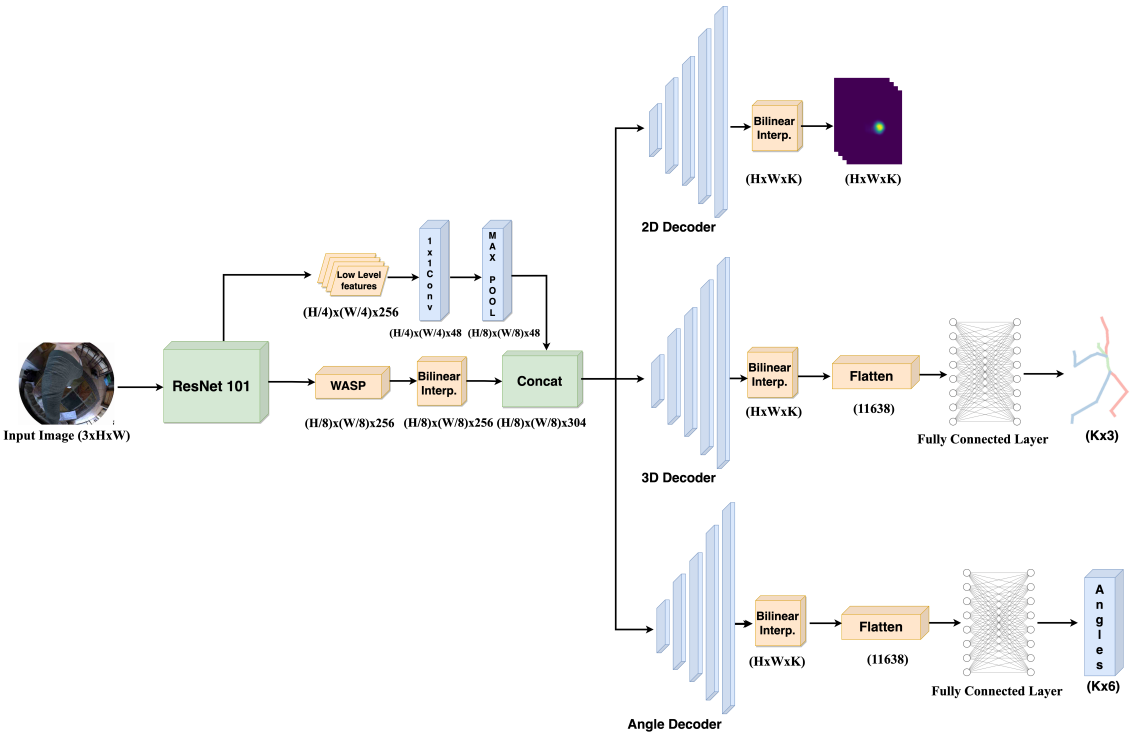


Figure 11.3.: The figure shows a triple branch feature-based network, EgoPoseNet. In addition to ResNet as the backbone, we integrating the WASP module [295], we process the input RGB image with dimensions $H \times W$ to generate 256 feature channels at a reduced resolution by a factor of eight. The first branch of EgoPoseNet produces $H \times W \times K$ 2D heatmaps, the second branch predicts $K \times 3$ 3D joint positions, and the third branch regresses a 6D representation of joint rotations with dimensions $K \times 6$.

the proposed dual-branch autoencoder (refer to Section 11.3.1), we identified a notable susceptibility to overfitting. This vulnerability arises due to the model's inherent complexity and the limited variance present in the xR-EgoPose training dataset. Additionally, the model faces challenges arising from insufficient information regarding the features. In response to these shortcomings, we introduce a novel feature-based triple-branch network. The architecture incorporates a feature-rich backbone and introduces an additional branch dedicated to the regression of joint rotation angles, aimed at predicting 3D joint rotations for each joint.

The input image undergoes processing through the backbone, which concatenates low-level ResNet-101 features with feature maps generated by the WASP module. The three branches of the network are responsible for estimating 2D heatmaps, 3D positions, and 6D representations of joint rotations for 16 joints, all sharing the same set of feature maps. We hypothesize that this shared feature representation contributes to enhanced learning, particularly when compared to lifting 3D pose exclusively from heatmaps. The latter approach tends to restrict learning to a specific camera setting, potentially leading to overfitting.

Subsequent sections provide detailed explanations for each module of the network.

Backbone

Motivated by the research conducted by Artacho *et al.*[295], we introduce a unified architecture for ego pose estimation. This architecture leverages the expansive Field of View (FOV) facilitated by atrous convolutions, combined with a cascade of convolutions arranged in a "Waterfall" configuration. The Waterfall Atrous Spatial Pooling (WASP) module is exploited to provide multiscale representations and enhance network efficiency with a more compact network size. To integrate this module into the network architecture, we substitute the final layer of ResNet-101 with the WASP module. This design choice aims to leverage the benefits of atrous convolutions for ego pose estimation while optimizing network efficiency through a compact and scalable configuration.

Heatmap Decoder

The Heatmap Decoder module is responsible for transforming the score maps generated by the Waterfall Atrous Spatial Pooling (WASP) module into heatmaps corresponding to various body joints. This decoding process involves receiving 256 feature maps from the WASP module and an additional 256 low-level feature maps from the first block of the ResNet backbone. Following a max pooling operation designed to align the dimensions of the inputs, the feature maps are concatenated and subsequently processed through a series of convolutional layers, dropout layers, and a final bilinear interpolation to resize them to the original input size. After performing a local max operation, the output comprises K heatmaps, where K is equal to 16 joints used for joint localization.

The training of the decoder involves utilizing the Mean Squared Error (MSE) loss denoted as L_h (Equation 11.4). This loss is computed as the MSE between the ground truth heatmap $H_{gt} \in \mathbb{R}^{48\times48\times16}$ and the predicted heatmap $H_{pred} \in \mathbb{R}^{48\times48\times16}$.

$$\mathbf{L_h} = \text{mse}(H_{gt}, H_{pred}) \tag{11.4}$$

3D Keypoints Pose Decoder

The 3D Keypoints Pose Decoder, akin to the Heatmap Decoder that combines low-level features generated by the Waterfall Atrous Spatial Pooling (WASP) module. Subsequently, convolutional layers, bilinear interpolation, and a max-pooling layer are applied in sequence. Following these operations, a flattened layer generates 11638 features, which are then forwarded to a fully connected layer for the regression of 16 3D joint positions.

Similar to the Heatmap Decoder, the training of this branch involves the utilization of the L_2 loss (Equation 11.5). This loss metric, denoted as L_p, computes the Mean Squared Error (MSE) between the ground truth pose $P_{3D} \in \mathbb{R}^{16\times3}$ and the predicted pose $\widehat{P}_{3D} \in \mathbb{R}^{16\times3}$.

$$\mathbf{L_p} = \text{mse}(P_{3D}, \widehat{P}_{3D}) \tag{11.5}$$

3D Rotation Angle Decoder

In Chapter 4, we delve into diverse representations for regressing 3D joint rotations. The discussion in Section 4.6.1 on 3D rotations underscores that conventional approaches like euler angles and quaternions often yield sub-optimal outcomes due to inherent discontinuity, making them prone to high errors.

Inspired by the findings in Chapter 4 and [127], we embrace a 6D continuous representation space for joint rotations. This choice enhances the capacity of deep neural networks to effectively learn rotations. Our subsequent focus addresses the challenge of ensuring orthogonality in rotation matrices while predicting 6D joint rotation representation.

Orthogonalization of Rotation Matrix: The ground truth data for 3D rotation angles in the xR-EgoPose dataset is represented in Euler angles. To facilitate training, we first convert these Euler angles to a Rotation Matrix in the "XYZ" rotation axis order. Subsequently, the Rotation Matrix is further transformed into a 6D representation for each of the 16 joints, to have continuous representation space for training the angle decoder branch.

We build upon the insights from the work of [127] to introduce two specialized loss functions for training rotations, forcing predictions to not only align with accuracy but also maintain orthogonality with the ground truth rotation matrices. We use the Gram-Schmidt-like mapping to enforce orthogonality of Rotation Matrix. While the 6D representation can be easily derived by Rotation Matrix, there are instances where the network struggles to learn the underlying cross product of $b_1 \times b_2$, resulting in a non-orthogonal rotation matrix [127]. This occurs when the dimension of the span of the input n-1 vectors to g_{GS} is less than n-1, preventing the Gram-Schmidt-like mapping from returning to SO(n). In the context of the 6D representation, this scenario arises when b_1 and b_2 are not linearly independent, leading one to be expressible as a linear combination of the other, and the span is confined to a one-dimensional subspace, resulting in a dimension of 1.

To address such scenarios, we apply additional two distinct loss functions during the optimization process. Firstly, the Rotation Matrix loss L_{rotmat} (Equation 11.6) checks the orthogonality of the matrix by verifying whether the dot product of the Rotation matrix R_{pred} with its transpose R_{pred}^T results in an Identity matrix I. Secondly, we use the Angle Loss $L_{AngleLoss}$ (Equation 11.7) to train the angle decoder. The $L_{AngleLoss}$ represents the distance between two matrices, the predicted rotation matrix R_{pred} and the ground truth rotation matrix R_{gt}.

$$\mathbf{L}_{rotmat} = \begin{bmatrix} 0 & if(R_{pred}.R_{pred}^T = I) \\ 1 & else \end{bmatrix} \tag{11.6}$$

$$\mathbf{L}_{AL} = \arccos \frac{tr(R_{pred}R_{gt}^T) - 1}{2}, \tag{11.7}$$

where tr denotes the trace of the matrix.

$$\mathbf{L}_{angles} = L_{mse} + (0.01 \times L_{AL}) + (0.1 \times L_{rotmat}), \tag{11.8}$$

where L_{angles} is the total loss for angle decoder branch and L_{mse} is l2 loss between 6D representation of ground truth Euler angle and 6D prediction by network.

Overall loss function of end-to-end network is as follows:

$$\mathbf{Loss} = (\lambda_h \times L_h) + (\lambda_p \times L_p) + (\lambda_a \times L_{angles}), \tag{11.9}$$

where $\lambda_h, \lambda_p, \lambda_a$ are the weights for heatmap loss, pose loss, and angle loss respectively.

Another key characteristic of our features-based network is that it can be used to train 2D poses, 3D poses, and angles independently or jointly, depending on the availability of ground truth pose data in a given dataset.

11.4 Implementation Details

The PyTorch framework [175] serves as the implementation platform for both networks, namely the Auto-Encoder and the three-branch based network featuring a backbone with Atrous convolutions. The training of these networks is conducted on four NVIDIA GeForce GTX 1080 Ti GPUs. The input images for both networks are re-scaled to dimensions of $368 \times 368 \times 3$. We train on both datasets namely xR-EgoPose [71] and Mo2Cap2 [298] The optimization process for both networks involves the use of the Adam optimizer.

In the Auto-Encoder network, the weights (λ) employed in the loss function were set to $\lambda_{hm} = 10^{-3}, \lambda_p = 10^{-1}, \lambda_\theta = -10^{-2}, \lambda_L = 0.5$. Conversely, for the Features-based network, the weights were configured as $\lambda_h = 0.4, \lambda_p = 0.4, \lambda_a = 0.2$.

11.5 Evaluation Results

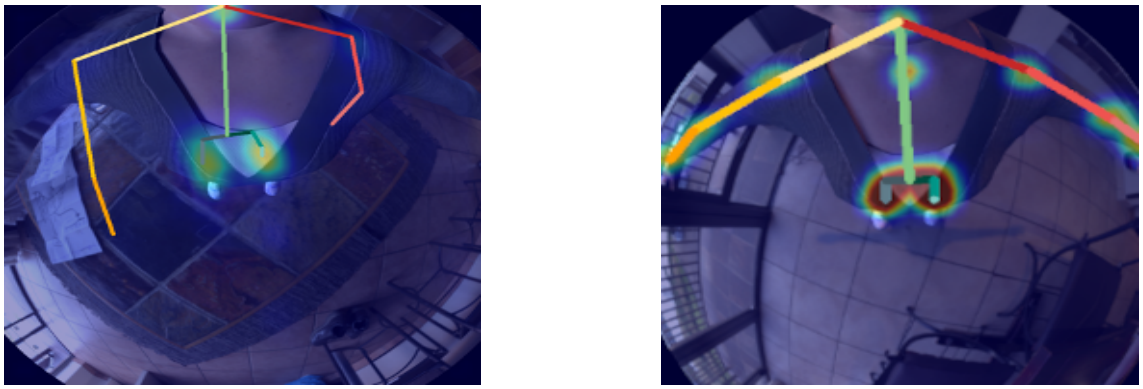
In this section, we present the results obtained on the xR-EgoPose and the Mo2Cap2 test datasets. Furthermore, we also explore the uncertainties associated with our model's predictions. Finally, we engage in an in-depth discussion on the distinct roles of each branch within the three-branch backbone and provide insights into the limitations of our approach.

The proposed approach is evaluated using the standard 3D keypoint pose accuaracy metrics of the xR-EgoPose dataset benchmark and the Mo2Cap2 dataset. For 3D pose evaluation metrics, we consider the Mean Per Joint Position Error (MPJPE) and Procrustes Alignment Mean Per Joint Position Error (PA-MPJPE). Additionally, we measure using the metric the Mean Pixel Error (MPE) to assess 2D poses and understand the influence of different branches on them. Note, since the camera itself is mounted on the person, measuring error at different scale is redundant. Thus, we do not use Percentage of Correct Keypoints(PCK and PCK-h) measures.

We measure the accuracy of rotations using Angle Error (AE) metric between ground truth and predicted rotation. Detailed discussions on these metrics follow in Appendix C.

Table 11.1 presents a quantitative comparison of our method with other state-of-the-art methods. Our proposed feature-based method, EgoPoseNet, provides competative performance with respect to state-of-the-art methods over MPJPE and PAMPJPE metrics at the time of writing. As there is no official implementation or weights provided by XR-EgoNet [71], we present the results of our implementation alongside the official scores given by the paper. The table highlights that our proposed network, EgoPoseNet, performs better than both our implementation and the official score of MPJPE for the XR-EgoNet network. Compared to XR-EgoNet[ours], EgoPoseNet reduces the error by 105.5 mm, while compared with XR-EgoNet [71], EgoPoseNet reduces the error by 0.9 mm. For PA-MPJPE, the error is reduced by 41.6 mm. EgoPoseNet is trained from scratch on a subset of Mo2Cap2 train data for 6 epochs, and despite being trained on a smaller dataset, it delivers comparable results on the Mo2Cap2 test set when compared to state-of-the-art methods that were trained on the complete dataset.

The qualitative results of EgoPoseNet on the XR-EgoPose dataset are presented in Figure 11.5. The first column showcases the input image, the second column displays the 2D results, and the final column presents the 3D results. Additionally, the second column includes heatmaps representing the Gaussian confidence in each joint's prediction. The first and last rows depict scenarios of severe self-occlusion of the left arm, while the left hand is self-occluded in row 5. Rows 3 and 4 have no visible wrists. Despite these challenges, all input images with significant lower-body occlusions, EgoPoseNet accurately predicted both 2D and 3D poses.



(a) xr-Ego Model **(b)** EgoPoseNet Model

Figure 11.4.: The Figure shows 2D heatmap uncertainty in model predictions. Fig(a) and Fig (b) show the comparsion of 2D heatmap regressed by xr-Ego and EgoPoseNet model respectively for partly visible wrist joints.

11.6 Ablation Studies

In this section, we conduct a detailed analysis of the influence of each component proposed by our approach on the overall performance. All ablation studies are carried out using

	xr-Ego		Mo2cap2	
	MPJPE	MPJPE-PA	MPJPE	MPJPE-PA
XR-EgoNet[ours]	162.8	82.3	-	86.85
XR-EgoNet [71]	58.2	-	-	-
Mo2Cap2 [298]	-	-	-	91.16
Wang et.al [299]	-	-	-	**83.17**
EgoPoseNet	**57.3**	**40.7**	176.49	98.13

Table 11.1.: Performance of EgoPoseNet network on xR-EgoPose test dataset [71] and Mo2Cap2 test dataset [298]. Our method outperforms the state-of-the-art methods on xR-EgoPose dataset and provide competitive results on Mo2Cap2 test set.

the selected XR-EgoPose test datasets. Subsequent sections present a comprehensive examination of the impact of each branch on both 2D and 3D pose estimation.

For the evaluation of 2D pose, we use the Mean Pixel Error (MPE) measured in pixels and the Mean Per Joint Position Error (MPJPE) measured in millimeters as the chosen evaluation metrics. The following subsections delve into the evaluation of angle errors, both with and without the incorporation of our newly introduced losses. Additionally, we explore the extent of reduction in instances of predicting invalid rotation matrices.

2D pose

The comparative analysis of our implementation of the XR-EgoNet model is presented in Table 11.2. Three distinct models within the EgoPoseNet architecture, developed progressively, are subjected to comparison. First is the EgoPoseNet single branch, exclusively focused on heatmap predictions. Next is the EgoPoseNet dual branch, featuring both a heatmap decoder and a poses decoder branch. Lastly, the EgoPoseNet triple branch, which integrates an additional angle decoder.

The angle decoder utilizes our novel loss function to generate scores. The results demonstrate a significant overall reduction of 57.15 pixels when compared to XR-EgoNet[ours], and a noteworthy decrease of 33.01 pixels in comparison to our single branch. This highlights the critical importance of incorporating 3D pose and angles in the training process.

	Neck	Shoulder	Elbow	Wrist	Hip	Knee	Ankle	All
XR-EgoNet[ours]	16.16	66.76	101.98	155.19	51.21	57.98	57.89	72.45
EgoPoseNet-single	24.22	35.84	64.05	96.76	37.94	39.61	39.64	48.31
EgoPoseNet-dual	24.28	29.93	49.77	83.35	21.56	21.43	22.08	36.06
EgoPoseNet-triple	**8.86**	**12.04**	**20.79**	**35.63**	**9.63**	**9.53**	**10.59**	**15.30**

Table 11.2.: The Table presents EgoPoseNet 2D pose evaluation results on xR-EgoPose test dataset [71] using Mean Pixel Error (MPE) metric. The Table presents a comparison of results for XR-EgoNet (our implementation), EgoPoseNet single branch, EgoPoseNet dual branch, and EgoPoseNet triple branch.

The qualitative assessment showed the robustness of EgoPose Net architecture over xr-EgoNet especially in the scenario of handling together of partial visibility and self-occlusion

of joints. The Figure 11.4 shows xr-Ego network failed to recognize arms due to absence of wrist in the image.

3D pose

The comparative analysis of our implementation of the XR-EgoNet model with the EgoPoseNet dual branch and the EgoPoseNet triple branch is presented. Here, the neck is excluded for comparison, as it is analogous to camera motion. The results reveal an overall reduction of 110.49 mm compared to XR-EgoNet[ours], and a notable decrease of 56.27 mm in comparison to our dual branch. This underscores the importance of angles in the training process.

	Shoulder	Elbow	Wrist	Hip	Knee	Ankle	MPJPE
XR-EgoNet[ours]	65.71	162.81	299.88	137.72	231.26	309.9	162.8
EgoPoseNet-dual	135.15	223.88	123.40	62.81	77.41	103.39	113.59
EgoPoseNet-triple	**29.53**	**60.03**	**77.34**	**53.7**	**59.2**	**79.64**	**57.32**

Table 11.3.: The Table shows EgoPoseNet performance on xR-EgoPose test dataset [71] for 3D pose using MPJPE error. The Table also compares the result for XR-EgoNet(our) implementation, EgoPoseNet dual branch, and EgoPoseNet triple branch.

Angle Error & Invalid Rotation Scenarios

The Table 11.4 presents a comparison of our EgoPoseNet network when trained without our two losses (row 1) and with them (row 2). It is evident that, on average, the angle loss for each joint is 1.5 degrees less when trained with our losses, underscoring their significance. Additionally, the stability of the network in angle predictions is depicted in Figure 11.7. We also computed the frequency of invalid rotation matrix scenarios with and without our losses on the XR-EgoPose dataset. The introduction of our new losses led to a 16% reduction in these invalid scenarios.

	Neck	Shoulder	Elbow	Wrist	Hip	Knee	Ankle	Mean
EgoPoseNet	$10.77°$	$19.53°$	$22.23°$	$18.22°$	$7.98°$	$7.74°$	$9.66°$	$11.85°$
EgoPoseNet-AL	$\mathbf{8.37°}$	$\mathbf{17.38°}$	$\mathbf{21.06°}$	$\mathbf{16.88°}$	$\mathbf{7.08°}$	$\mathbf{6.89°}$	$\mathbf{8.37°}$	$\mathbf{9.34°}$

Table 11.4.: Performance of our egocentric pose estimation network on xR-EgoPose test dataset [71] for 2D pose. The angle error measured in degrees for various skeleton joints are shown for two different models. The Table compares the result for EgoPoseNet(row 1) and EgoPoseNet-AL with additional angle losses(row2).

11.7 Uncertainty Estimation

Recently research methods have incorporated uncertainty estimation in computer vision tasks such as for object detection [96, 300, 301]. Uncertainty in pedestrian tracking has

been studied by [302, 303, 304]. The work of [305] predicts uncertainty on facial landmarks and [306] uses aleatoric uncertainty for human pose estimation.

Given practical constraints, utilizing an ensemble of multiple models is not feasible for our experiments. Consequently, we limit our investigations to the Monte Carlo dropout method, as detailed in the following section.

Monte-Carlo dropouts

We use Monte Carlo Dropout for modeling uncertainty (see section 2.5). Figure 6.3, shows the results of uncertainty in model predictions using the Monte Carlo Dropout method. The predictions are done for 150 images sequence, with n=20 forward passes. The shades of color represent model uncertainty with predictive mean plus/minus 2 standard deviations.

The Figure 11.6(a), (b), and (c) shows a well calibrated uncertainty for each dimension representing 3D keypoints, depicting the high confidence of the model learning 3D joint positions. The uncertainty plots show higher variance Z-axis as compared to other two-axis representing inherent depth ambiguity of joints with respect to camera.

The Figure 11.7 shows the computed mean and variance of angle difference between the ground truth rotation matrix and the predictions rotation matrix of right shoulder joint. The angle error during occlusion of right arm between frames 40 to frame 80 results in higher error. However, we observe that the variance is not well calibrated during this phase and needs further investigations.

The 2D heatmap uncertainty in model predictions is depicted in Figure 11.8. Figures 11.8(a) and 11.8(b) display frames featuring an overlay of 2D output heatmaps. Notably, Frame 79 stands out for having the lowest prediction error, attributed to the full visibility of the right arm, including the wrist joints. In contrast, Frame 67 reveals a substantial error in the right shoulder prediction, mainly due to the partial visibility of the left arm. While the model adeptly handles self-occlusion of lower leg joints, it grapples with accurately estimating rotation values in scenarios with partial visibility. Consequently, we contend that further investigation and research are essential to devise new techniques for effectively addressing partial visibility challenges in model predictions.

11.8 Conclusion

We present a framework for 3D egocentric pose estimation from a monocular perspective using a fisheye camera mounted on an individual's head. The research introduces a novel triple-branch network centered around features, emphasizing the significance of each branch in prediction outcomes. The study underscores the utility of the WASP module in enhancing the network with input information and improving occlusion handling. Furthermore, the thesis demonstrates that a 6D representation offers greater continuity and a more effective representation space for networks to learn joint rotations. Additionally, two innovative losses are introduced to facilitate the learning of joint angles, mitigating issues

associated with invalid rotation matrices. Lastly, the research introduces methodologies for gauging the confidence of the network in predicting 2D pose, 3D pose, and angles for 3D Egocentric Pose.

A a Residual Attention Network can leverage soft attention layers for enhanced focus on relevant features crucial for high performance sports pose estimation. The implementation of forward and inverse kinematics over the 6D representation, enables direct application of losses between predicted and constructed 3D poses and 3D joint rotations. Lastly, the conditional variational autoencoder [307], can provide uncertainty distribution during pose changes at each step of a motion sequence, providing a effective means to model uncertainty.

Acknowledgments

I would like to extend my sincere appreciation and acknowledgment for the support and discussions from Dr. Bertram Taetz, which significantly enriched the content and ideas presented here. I extend my gratitude to Arpit Lekhra for his contributions towards codebase and model evaluation. The research work is funded under the project DECODE.

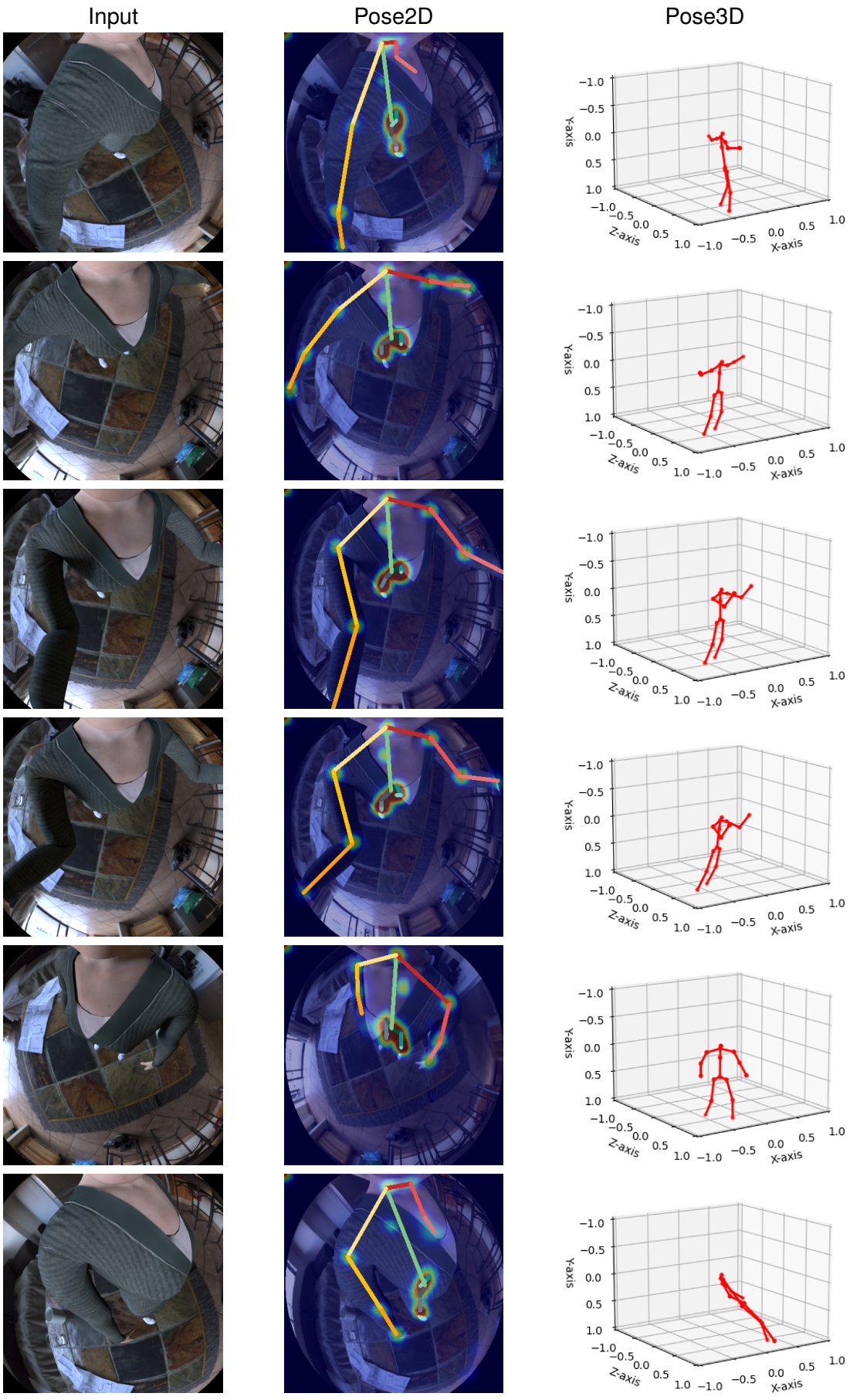
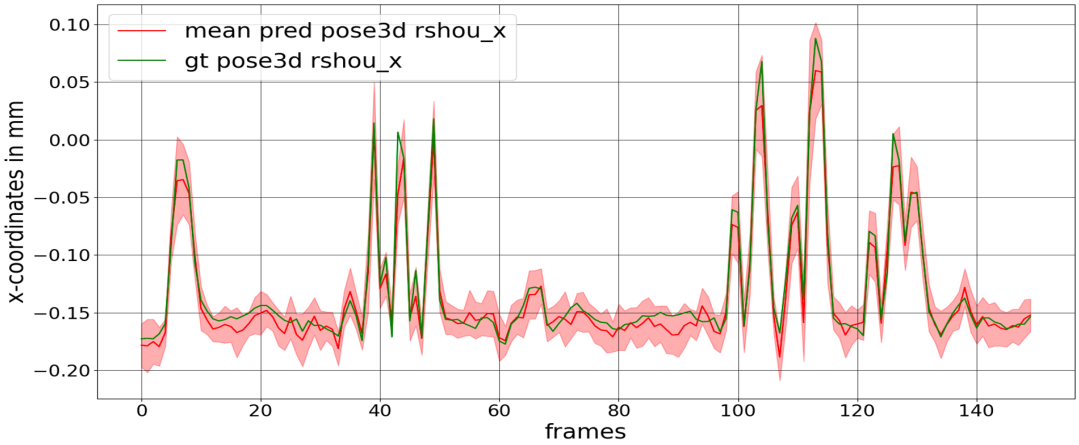
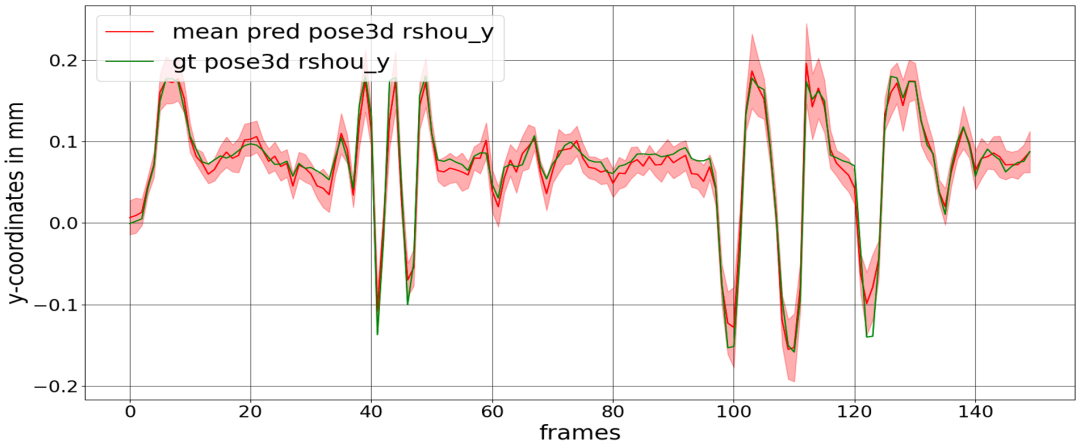


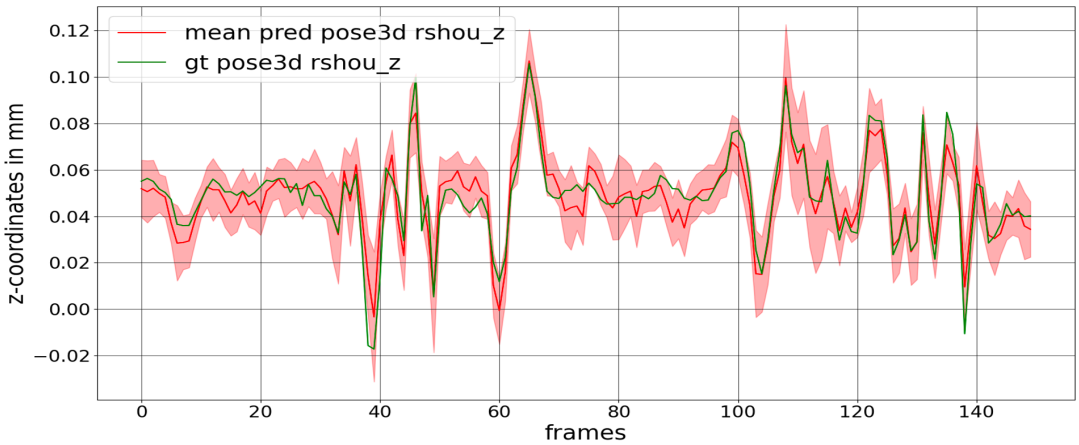
Figure 11.5.: The figures illustrate qualitative results showcasing 2D and 3D pose outputs on the XR-EgoPose dataset



(a) The 3D joint position prediction of x-coordinate for the right shoulder joint

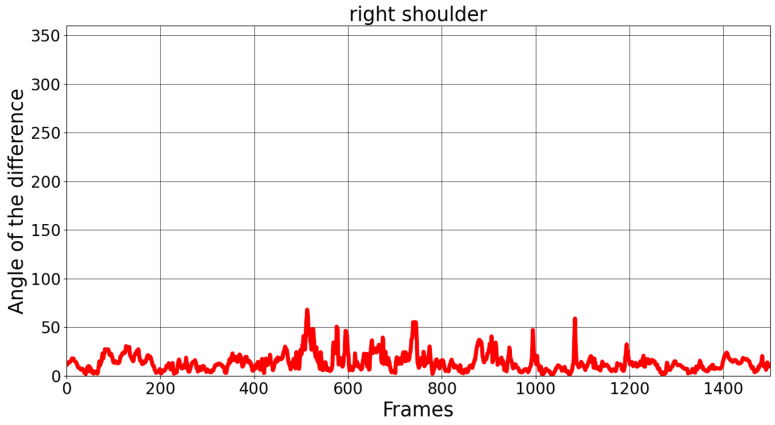


(b) The 3D joint position prediction of y-coordinate for the right shoulder joint

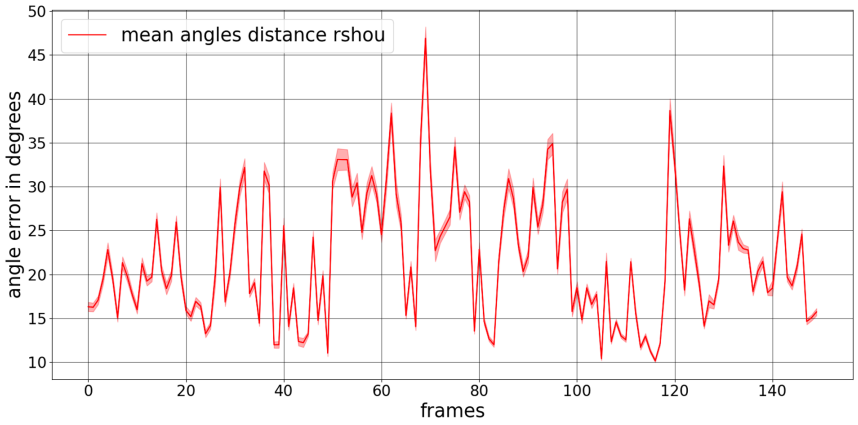


(c) The 3D joint position prediction of z-coordinate for the right shoulder joint

Figure 11.6.: Figure shows uncertainty in model predictions using the Monte Carlo Dropout method. The x-axis in all the figures shows the frame number. The y-axis shows the mean and variance of 3D joint predictions in red and ground truth in green in mm

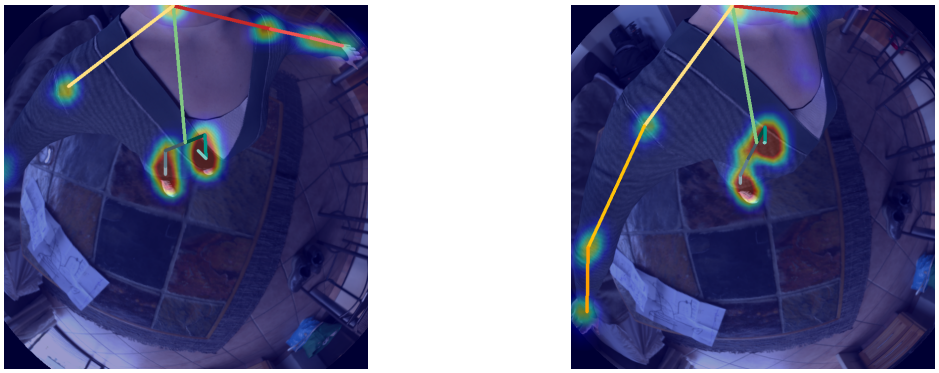


(a) The angle (axis-angle) error for right shoulder joint over a short sequence



(b) The uncertainty measured with angle error in a short sequence for the right shoulder joint

Figure 11.7.: Figure shows model predictions angle error. The x-axis in all the figure shows the frame number. The y-axis depicts angle error in red.



(a) Frame Number 67

(b) Frame Number 79

Figure 11.8.: The Figure shows 2D heatmap uncertainty in model predictions. Fig(a) and Fig (b) show the frames with 2D heatmap overlayed. The Frame 67 shows large error on right shoulder due partial visibility of left arm. Similary, the model prediction has lowest error on Frame 79 due to full visible of right arm including the wrist joints.

Part V

Conclusion

Conclusion

12

> *"You can't connect the dots looking forward; you can only connect them looking backward"*
>
> — **Steve Jobs**
> Apple Inc

In the concluding chapter, we delve and reflect on the journey undertaken and the insights gained. We draw upon the knowledge learned in the different parts of the thesis to form a cohesive and insightful conclusion. The subsequent section encapsulates a comprehensive summary, providing an overview of the accomplished objectives while addressing critical issues, and presenting key findings from different parts of the thesis.

12.1 Summary

In the current thesis, we explore different deep learning models designed for human pose estimation under partial observations. The inception of our investigation involved preliminary experiments, setting the stage for a more rigorous empirical inquiry into complex scenarios. With a meticulous approach to research design and implementation, our primary aim was to tackle the fundamental objectives of accurately estimating human pose parameters when confronted with the challenges of self-occlusion and partial visibility of joints in diverse camera-view setups.

Before we delve into modeling of specific partial observation scenarios, we first learn the range of motion of 3D joints using motion capture dataset. Further, these constraints are used to train by checking the validity of regress pose output in a valid configuration space [308]. The trained discriminators over synthetic data using lookup maps can be utilized as priors for human pose estimation and motion synthesis tasks. Additionally, a deep learning module 3D joint angle regression from monocular RGB images. This method extends beyond conventional 3D joint position estimation, providing a broader representation of learned 3D poses. Notably, it facilitates the prediction of twists along body segments, thereby improving 3D human body pose and shape estimation.

Next, we focused on self-occlusion occurring in exocentric-view scenes. We model self-occlusion in a springboard diving action scenario. The springboard diving is particularly challenging due to their high-speed and intricate body maneuvers resulting in self-occlusion from exo-centric monocular view. To address this, we breakdown the performance analysis

into two sub-problems: first, we learn the diver's body pose; second, we detect diving actions in continuous video footage. We present a DSV diving dataset, a collection of 450 diving videos, each captured at a frequency of 60 Hz, covering performances from various springboard heights. We train a DiveNet pose regression network [309] to estimates robust 2D joint keypoints of the diving athlete in each frame. The presented DiveNet action localization framework [310] excels in localizing diving actions in unconstrained environments with low latency. The framework also enabling precise measurement of dive heights using estimated temporal motion boundaries and projectile motion constraints. The DiveNet framework combined with localization module and pose regression module enables to provides the diver's center of mass (COM) trajectory and a homography computation method for aligning the COM trajectory in the diving motion plane with the image view. This provides us a metric-scale parameter representation devoid of complex camera calibration procedures.

The second challenging scenario involving partial observation deal with partial visibility of the human body within a visual scene. Partial visibility is another source of partial observation, which occurs often due to limited camera field of view. It hinders the learning of joint subsets, making it more challenging to recover the body pose.

We focus on driver pose tracking and monitoring scenario in-cabin car simulation environment with an aim to enhance driving safety. While RGB cameras may struggle in the dark, Time-of-Flight cameras can capture images using infrared light, allowing for improved visibility at night. This enhances overall surveillance capabilities, making it easier to monitor people inside the car during all lighting conditions. To this end, we present the driver pose dataset, capturing driver postures and activities in an in-car simulator environment. The dataset leverages the DFKI cabin simulator setup, featuring both a Kinect camera and an OptiTrack motion capture system. It encompasses 19 participants, 266 videos, and 37 types of actions, providing synchronized RGB, IR, and depth data. We generate ground truth using both the Kinect camera and neural network based mocap labeling method(SOMA) [237]. We also present 3D driver pose estimation inside a car using depth data and point clouds [311]. Competitive results are achieved in real-time on recorded driving scenes, with model accuracy benchmarked against generated ground-truth data.

The final part deals with the ego-centric visual environment, addressing partial visibility and self-occlusion concurrently. Ego-centric pose tracking provides self-pose estimation for individuals while moving with a mounted camera. We initially use a publicly available large synthetic ego-centric pose dataset (xr-Ego Dataset) to train ego-centric 3D pose regressors [71, 312]. Additionally, we learn the 3D joint rotations represented in 6D representation. While the trained approaches provide single solutions, we analyze the model uncertainty for challenging scenarios such occlusion.

The overall key findings and insights gained in pursuit of our research questions (presented in Chapter 1.3) are as follows:

- The data-driven strategy using Swing-Twist parameterization provides a more intuitive way to learn 3D joint rotation constraints. [Part 1, Research Question 1]

- The 6D representation provides more accurate 3D pose regression results due to its continuous representation for training deep learning networks. While regressing joint rotations using forward kinematics during training can further improve the accuracy of the 3D pose. [Part 1, Research Question 2]

- The state-of-the-art pose estimation models perform poorly on complex self-occlusion poses involved in diving sports. The DiveNet pose regression network trained on newly curated DSV dataset provide highly accurate results for diving training videos. The model compression techniques can be effective in model generalization and reduce overfitting to diving datasets. [Part 2, Research Question 1]

- DiveNet action localization network using temporal convolutions accurately localizes action with low latency using spatio-temporal features over video frames. The present method accurately demarcates the diving motion with over 90% classification accuracy and latency accuracy. The model necessitates the temporal context of 6-8 frames on both sides to accurately localize motion. The pose regression feature extraction has superior accuracy over generic object classification feature extractors. [Part 2, Research Question 2]

- The diving motion parameters such center of mass trajectory, key joint angles and angular rotation are useful performance metrics to benchmark diving motion. The homography computation helps recover true performance metric values invariant to camera viewpoints. The developed framework and analysis tools help visualize trainers key metrics, effectively compare multiple dives and provide instant feedback. [Part 2, Research Question 3]

- The partially visible pose can be annotated using deep learning methods [237] to resolve the issues of missing markers and ghost points during the motion capture. We present a novel DREICAP-3D, a driver pose and activity dataset providing synchronized RGB, IR and depth images. [Part 3, Research Question 1]

- The fusion of IR and depth maps provide an improved 3D pose estimation results with average error of 19 mm. The volumetric heatmaps autoencoder provides an effective way to have compressed representation of volumetric heatmaps representing 3D pose. Further, the presented PointConvNet with local feature aggregation module enables learning of 3D pose keypoints from point cloud data with a mAP of 86% with in a threshold distance of 10 cm. However, further research is required to enable the learning of 3D joint rotations from point clouds. [Part 3, Research Question 2]

- The presented triple branch network, EgoPoseNet demonstrated that learning 2D pose, 3D keypoints and 3D joint rotations complement each other. The qualitative results shows the model robustness to self-occlusion and partial views in ego-centric videos. [Part 4, Research Question 1]

- Quantifying model uncertainty through MC dropout [93] yields higher uncertainty estimates for joint rotations, particularly in frames with partial observations. However, additional research is necessary to obtain more stable and calibrated uncertainty. [Part 4, Research Question 2]

12.2 Future Work

In this section, we discuss the future outlook of human pose estimation handling partial observations. The current research can be extended into three different research directions as discussed below.

Deep Probabilistic Models: The uncertainty analysis conducted and included in the section of study although represent preliminary findings, necessitates further exploration into alternative methodologies that integrate aleatoric and epistemic uncertainties (deep ensembles [97]), or those methods that obviate the need for modifications to prediction networks. Additionally, the influence of viewpoint on uncertainty merits investigation as a prospective avenue for future research. Notably, the outcomes presented in this thesis pertain to the regression task of pose estimation. It would be compelling to investigate the performance of other probabilistic deep learning methodologies [313, 314, 307, 95] in the context of influence of self-occluding joints and partial visibility on explicitly visible joints. Recent works such as UPose3D [315] leverage temporal cues and 2D pose uncertainty in cross-view information. The uncertainty aware pose among frames in a video due to occlusion can be further enhance overall 3D pose estimation [316].

.

Continual Learning: Unlike conventional static models that assume a consistent distribution of pose data, continual learning acknowledges context specific human movements and body configurations. As individuals exhibit diverse postures over time, continual learning allows pose estimation models to dynamically evolve by incorporating new poses into their training data. This continual learning process enhances the model's ability to adapt to variations in human movements, improving accuracy and robustness in estimating poses across a range of activities and scenarios [317]. In essence, the integration of continual learning in human pose estimation ensures a more responsive and effective model over pretrained static models. It allows the model to actively learn the human body poses to bridge the domain gap between the training and test data [318].

Generative models: Generative modeling can be also very useful to generate or synthesize unseen human poses to overcome the lack of availability of data. Variational Autoencoders (VAEs) aims to learn a probabilistic mapping between the input data and a latent space. In the context of pose, VAEs can be trained to encode the variability in human poses into a

lower-dimensional latent space. This latent space can then be sampled to generate new, realistic poses. Another interesting approach would be to estimate given pose uncertainty by identifying if it is out of distribution sample by means of Autoencoder Attractors [319]. The recursive application of a trained autoencoder model, can also be interpreted as a human motion dynamical system with desirable properties for generalization and uncertainty estimation. The Conditional Generative Models use additional context or specific conditions into account while generating poses [307]. For example, a conditional generative model could be conditioned on to generate a specific class of dive out of 48 different classes [152], allowing it to generate poses specific to only a specific diving class. Lastly, the conditional variational autoencoder [307], could also effectively learn a human motion distribution to predict pose changes at each step of a predefined motion sequence, providing a effective way to model uncertainty. It's worth noting that the success of generative modeling in generating realistic human pose depends on the quality and diversity of the training data, the chosen architecture and training parameters of the generative model.

Integrating pose estimation using Generative Pretrained Transformer (GPT) models could also lead to novel applications, such as generating realistic human movements or enhancing virtual avatars' interactions with their environments [320, 321]. By leveraging generative model's ability to understand context and generate realistic outputs, pose estimation models could produce more accurate and nuanced representations of human poses, contributing to fields like animation, virtual reality, robotics, and healthcare. Additionally, advancements in Generative AI could lead to more efficient and robust pose estimation models, capable of handling diverse scenarios and generating realistic human movements in real-time.

To sum up, the presented work augments the dynamic landscape of human pose parameter estimation with contributions of specific frameworks for practical applications. The presented experiments provide a solid foundation for further exploration and progress in better understanding and estimating robust pose under partial observations. The work presents deep learning models exploiting contextual cues in a three different application scenario varying camera setups. However, the inherent complexities of regressing 3D pose and modeling pose uncertainty needs further research. Nevertheless, the ability of model to learn pose in exo-centric and ego-centric simultaneously will certainly provide new avenues of research towards novel ways of human-object interaction.

Appendices

Deep Learning Modules

A

A.1 Stacked Hourglass

The Stacked Hourglass Network (SHN) is a deep neural network architecture designed to process features at different scales [322]. The SHN architecture has been shown to be effective at capturing both the fine-grained and coarse-grained details of human poses. The architecture consists of a series of hourglass modules that are stacked on top of each other. Each hourglass module consists of a series of downsampling and upsampling operations, allowing the network to learn hierarchical features at different scales.

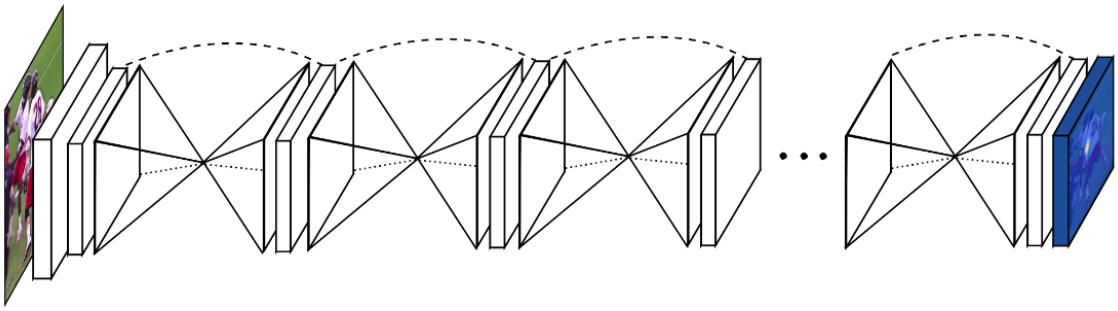


Figure A.1.: Stacked Hourglass network for pose estimation consisting of multiple stacked hourglass modules allowing for repeated bottom-up, top-down inference [322].

The module starts with a series of convolutional and max-pooling layers that gradually reduce the spatial resolution of the input image. At each max-pooling step, additional convolutional layers are applied to the pre-pooled resolution. The network then moves to the top-down sequence, where the features are upsampled and combined across different scales. To bring together information from two adjacent resolutions, nearest neighbor upsampling is performed on the lower resolution, followed by an elementwise addition of the two feature sets. The topology of the hourglass module is symmetric, with each layer present in the downsampling process having a corresponding layer in the upsampling process. After reaching the output resolution, the network makes the final predictions through two consecutive rounds of 1x1 convolutions. The output of the network is represented as a set of heatmaps, where each heatmap predicts the probability of a joint's presence at every pixel in the image. Figure A.2 shows a visual illustration of the hourglass module, excluding the final 1x1 layers.

The hourglass module consists of a series of convolutional and max-pooling layers that reduce the spatial resolution of the input image. This downsampling process helps the network learn features at a lower resolution, which can be useful for detecting large body

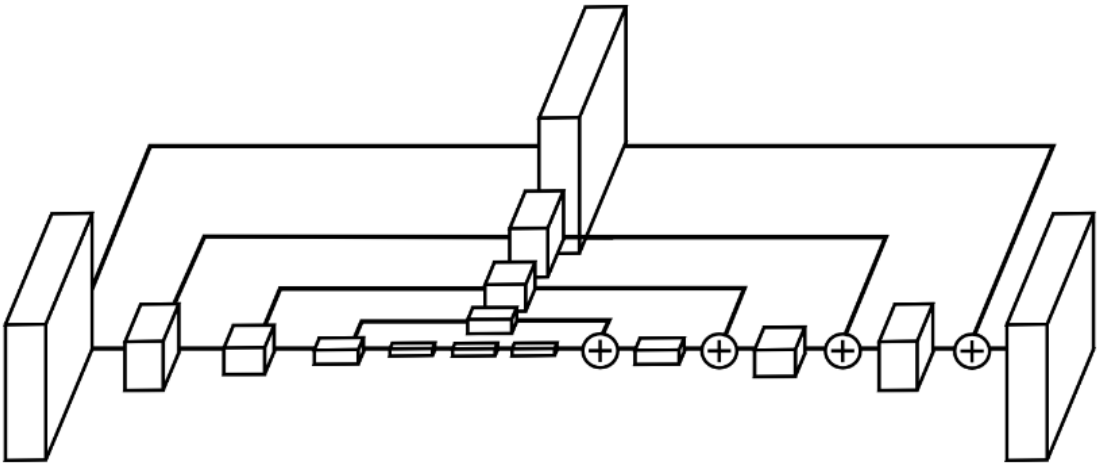


Figure A.2.: A single "hourglass" module is illustrated. As shown in this Figure, each box in the diagram corresponds to a residual module. Throughout the entire hourglass, there are the same amount of features [322].

parts such as the torso. After the downsampling process, the features are upsampled using transposed convolutions, which restore the original resolution of the input image. This allows the network to learn features at multiple scales, which can be useful for detecting fine-grained details such as the fingers and toes.

A.2 MobileHumanPose Model

MobileHumanPose [170] stands as a mobile-friendly 3D human pose regression model designed explicitly for real-time estimation on mobile devices. Leveraging MobileNetV2 [172] as its backbone network, incorporating a parametric activation function and skip connections, the model adopts a lightweight structure conducive to mobile deployment. Remarkably, it achieves comparable performance to state-of-the-art methods while maintaining a significantly smaller size, approximately seven times less.

In contrast to approaches by Moon et al. [179] and Chen et al. [323], which prioritize accuracy over efficiency, MobileHumanPose [170] follows an encoder-decoder structure. The encoder undertakes global feature extraction, while the decoder is dedicated to pose estimation [170]. The network architecture is illustrated in Figure A.3.

Backbone Network: The model explores various backbone networks, including ResNet [324] and MobileNetV2 [172]. Figure A.4 illustrates that MNasNet [325] yields the lowest Mean Per Joint Position Error (MPJPE). However, due to the lower number of parameters and comparable accuracy, MobileHumanPose opts for MobileNetV2 as its backbone. Modifications are applied to the MobileNetV2 structure by adjusting the channel size for the first four inverted residual blocks to enhance performance [170].

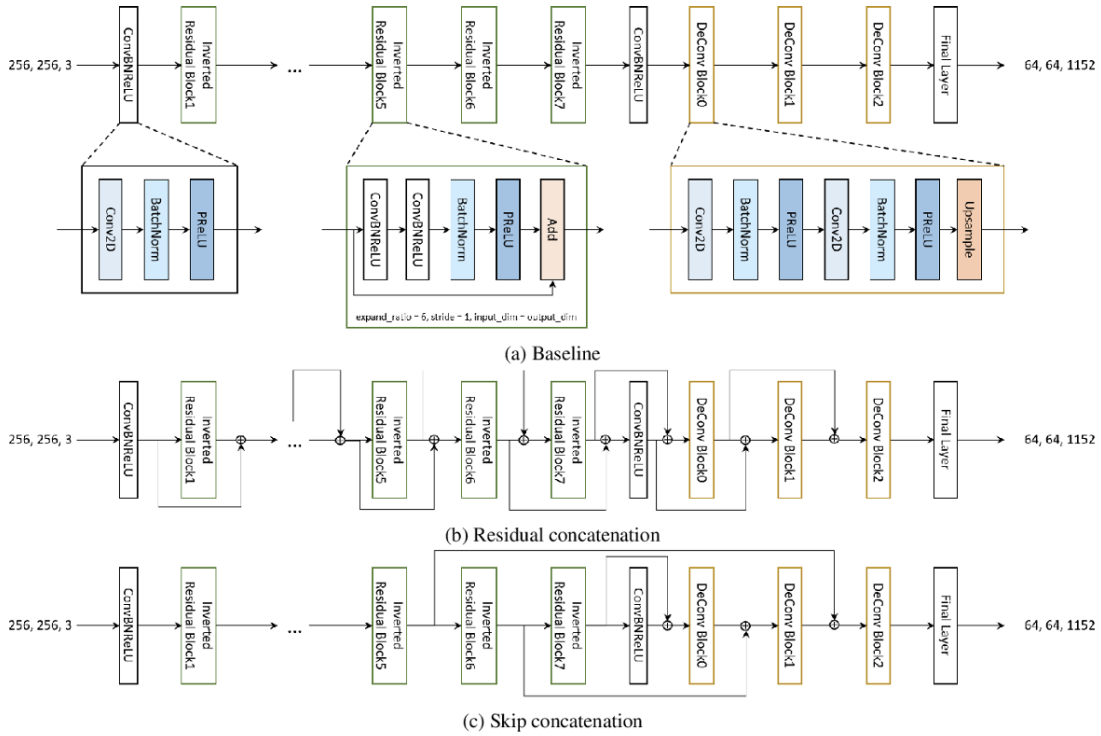


Figure A.3.: MobileHumanPose model architecture [170]

Backbone	Param.	FLOPS	Avg.
GhostNet	2.02M	1.21G	83.73
MobileNetV3	4.28M	1.39G	70.94
MobileNeXt	1.83M	1.39G	61.27
MobileNeXt*	1.83M	1.39G	61.19
MNasNet	4.23M	1.49G	55.85
MobileNetV2	3.36M	1.48G	56.38
MobileNetV2*	3.36M	1.48G	57.09
ResNet18	12.55M	3.46G	56.96
ResNet18*	12.55M	3.46G	53.45
ResNet50	28.42M	7.36G	54.80
ResNet50*	28.42M	7.36G	53.79

Figure A.4.: Comparison with various backbone networks. * denotes the corresponding network pre-trained on ImageNet. Metric: MPJPE (mm) [170]

Skip Concatenation: The model strategically employs FLOPS (Floating Point Operations) in the PyTorch [326] framework to minimize inference time. It utilizes two different concatenation structures, namely, residual and skip concatenation (Figure A.3(b) and A.3(c)). In residual concatenation, average pooling and bilinear functions are employed in the encoder and decoder parts, respectively. In skip concatenation, pure concatenation is used when the output dimensions are consistent. This design choice in skip concatenation enables the derivation of low-level features from the encoder to the decoder [170].

A.3 SPIN

SPIN [144] model is a frame-base method which combines both the optimisation-based and regression based approaches [144]. For regression based approach, HMR model [90] is used and for optimization based approach, SMPLify pipeline [17] is used. The model architecture is as shown in Figure A.5.

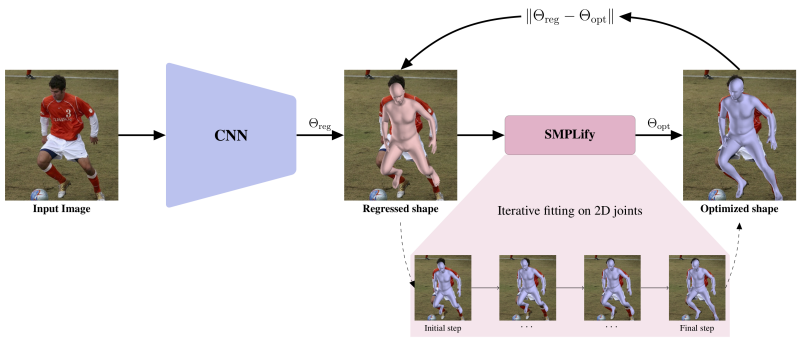


Figure A.5.: Architecture of SPIN [144].

Here, the input image is given to a CNN based regression (HMR) which regresses 3D human body and shape ($\Theta_{reg} := \{\theta_{reg}, \beta_{reg}\}$). These pose parameters are then utilised by iterative optimisation based algorithm, SMPLify. SMPLify uses the 2D re-projection loss to optimise the fitting iteratively. The optimised pose parameters ($\Theta_{opt} := \{\theta_{opt}, \beta_{opt}\}$) act as a strong supervision, compared to the 2D re-projection loss which is ambiguous and hence is a weaker supervision term. SPIN model is also self-improving because estimated model parameters from HMR model act as a good initialization for SMPLify (and vice-versa). Since, SMPLify utilises only the 2D keypoints for fitting, there is no need for 3D data, hence this model works even if there is less 3D supervision is provided. The HMR model also outputs camera parameters $\prod_{reg}$ which is used to compute 2D re-projection of 3D joints X_{reg} as defined in equation:

$$J_{reg} = \prod_{reg}(X_{reg}). \tag{A.1}$$

The mesh parameters are obtained by SMPL body model as $M_{reg} = M(\theta_{reg}, \beta_{reg})$. The losses used for supervision are as follows:

$$L_{2D} = ||J_{reg} - J_{gt}||, \tag{A.2a}$$

$$L_{SMPL} = ||\Theta_{reg} - \Theta_{opt}||, \tag{A.2b}$$

$$L_{3D} = ||X - \hat{X}||, \tag{A.2c}$$

$$L_{M} = ||M_{reg} - M_{opt}||. \tag{A.2d}$$

Here, L_{2D} is 2D re-projection loss, L_{3D} is a 3D supervision term which defines a mean squared loss whenever the 3D key-points (X) are available. L_{SMPL} is composed of SMPL

parameter losses. L_M is defined for mesh level supervision. SMPLify routine fits SMPL body model to 2D joints with an objective function:

$$E_J(\beta, \theta; K, J_{est}) + \lambda_\theta E_\theta(\theta) + \lambda_a E_a(\theta) + \lambda_\beta E_\beta(\beta), \tag{A.3}$$

where, β and θ are the SMPL body model parameters, J_{est} are the estimated 2D joints from 2D detector or ground-truth 2D locations and K is the camera parameter. The first term of the objective function $E_J(\beta, \theta; K, J_{est})$ penalises the euclidean distance between projected SMPL joints and 2D joints J_{est}. This objective function does not include the interpenetration error from the original SMPLify algorithm, because it is slow and does not weight much in accuracy of the poses. $E_\theta(\theta)$ is the pose prior loss which penalises the impossible human poses. $E_a(\theta)$ is the angle prior loss which puts a constraint on unnatural knee and elbow joint rotations. $E_\beta(\beta)$ constraints body shape values from taking large values. SMPLify algorithm is performed in two steps, in the first step, camera translation and body orientation is estimated and body model parameters are fixed. The loss for this step is called as camera-fitting loss, which computes the euclidean loss between the estimated camera translation with the initial camera translation. In the second step, using the estimated camera translation, SMPLify fits SMPL body model to 2D joints minimising the objective defined in equation A.3, which is called as body-fitting loss.

A.4 VIBE

VIBE [327] is a temporal model which is built on top of the SPIN model. It uses the features generated from the SPIN model and processes them to a temporal encoder which learns the latent representation from the past and future frames. The model uses GRUs [328] to learn the temporal representations. These features are further trained by regressor, similar to SPIN model to get the 3D pose as output. The model further uses a motion discriminator which is trained on AMASS dataset to distinguish between fake and real poses. AMASS [329] is a collection of all the major Motion Capture datasets and provides SMPL parameters for each of them. Temporal SMPLify is used as optimisation in this approach, in which SMPLify optimisation approach is used in temporal manner to account for the temporal features from the temporal encoder.

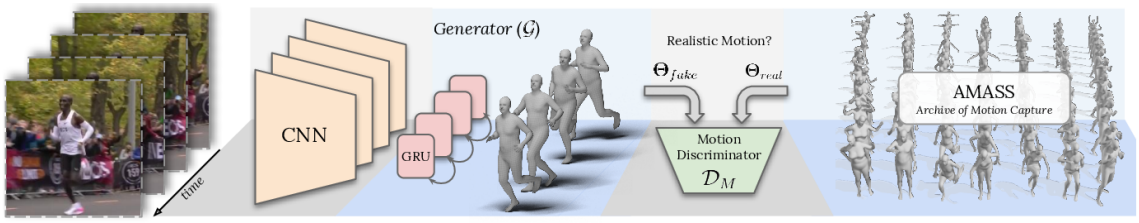


Figure A.6.: Architecture of VIBE. [327]

Similar to SPIN model, VIBE combines 3D, 2D re-projection, and SMPL losses to deal with the weak-supervision, which constrains the model to a more confined representation space. Additionally, it uses adversarial loss to constrain the learned representation to plausible human poses. The discriminator also utilises the self-attention method [330] which combines the all the important hidden state information instead of the using just the last hidden state output.

A.5 SOMA Model

The SOMA [237] model can automatically label the original point cloud obtained by marker-based optical motion capture(mocap) and generate the corresponding human body model. The whole process is shown in Figure A.7. We first generate a synthetic dataset for training the SOMA model. Then label the target mocap point cloud. Finally generate the SMPL-X human body model that fits the point cloud.

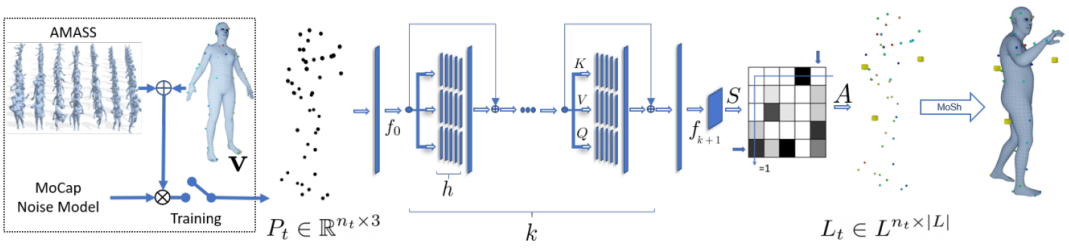


Figure A.7.: The entire pipeline of SOMA model [237]. From generating synthetic training data to labeling the point cloud and computing the corresponding SMPL-X model

The network architecture of the SOMA model is shown in the Figure A.8a. The median centered point cloud is used as input. It first goes through a Conv1D-block and a Relu activation function, followed by an 8-layer self-attention block and a Relu activation function, next is a Conv1D-block and a Relu layer, and finally a Conv1D-layer and an Optimal Transport layer. The Optimal Transport layer uses the itertive Sinkhorn normalization proposed by Sinkhorn et al [331]. It is mainly to make the predicted non-squared score matrix satisfy the constraints of the point cloud labeling problem, that is, for $S \in \mathbb{R}^{n_t \times M}$, $\sum_{i=1}^{n_t} s_{ij} = 1, \forall j \in [1, M]$ and $\sum_{j=1}^{M} s_{ij} = 1, \forall i \in [1, n_t]$ [331] for the available points and labels. As for ghost points and labels, we can add an extra last row and column and then iteratively normalize the matrix along rows and columns. The entire pipeline is shown as Figure A.9 [332]. The detailed self-attention block architecture is shown in Figure A.8b [237]. The input feature of the self-attention block is 125. It first reshapes to $3 \times 5 \times 25$, then goes through a scaled dot product layer and reshape the feature to 125 again. Next is a Conv1D layer and a concatenate layer. The concatenation layer concatenates the output of Conv1D with the input of the self-attention block. After that are two Conv1D-block, which contain a batch normalization layer and a Relu activation function between them. Multiple self-attention blocks connected by residual operations. The scaled dot product attention

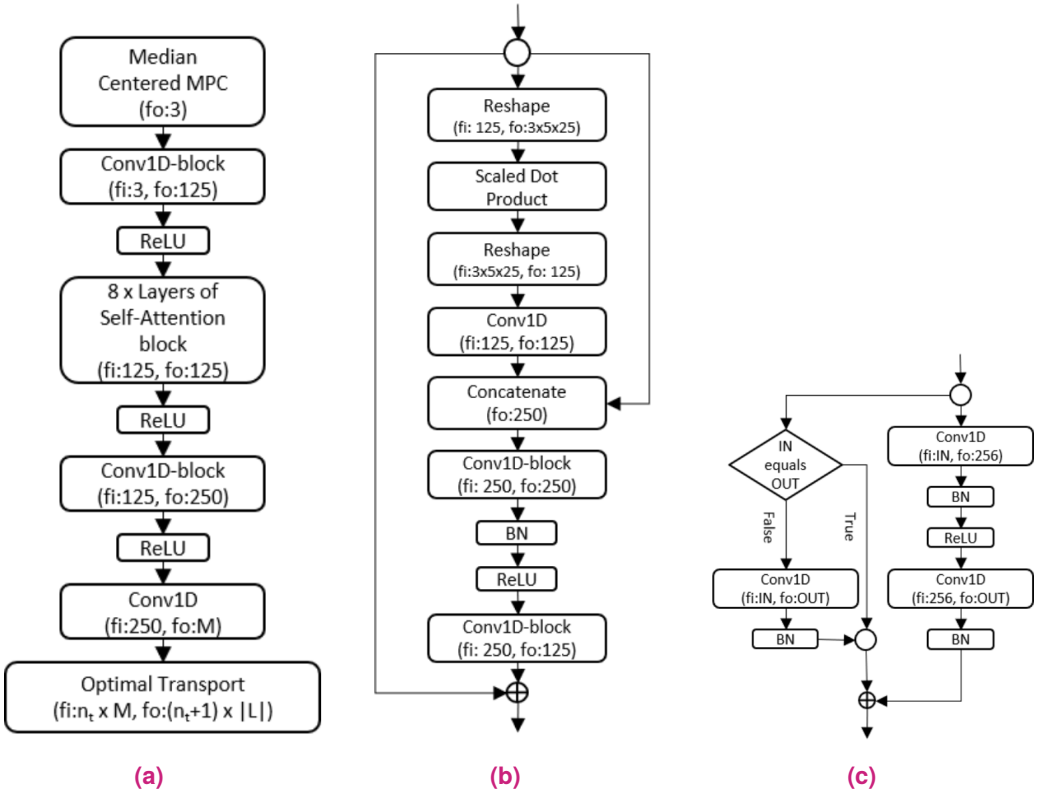


Figure A.8.: Architecture of SOMA model(left), self-attention block(middle) and Conv1D-block(right) [237]

[333] is an attention mechanism and the structure is shown in Figure A.10a [333]. The attention calculation function is defined by the Equation A.4 [237].

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_{model}}})V \tag{A.4}$$

where Q is the query, K is the key, V is the value and d_{model} means that the dot product is scaled down by $\sqrt{d_{model}}$.

Figure A.10b shows the multi-head attention, which consists of several attention layers running in parallel. The use of multi-head attention increases the capacity of the network, has a local and global view of the point cloud, and extracts the features of the point cloud from multiple levels.
The Conv1D-block is shown in Figure A.8c [237]. The input of the Conv1D-block goes through two Conv1D layers with batch normalization, The input of a Conv1D block goes through two Conv1D layers with batch normalization, which are connected with a ReLU activation function. The obtained result is summed with the input by the residual operation. If the number of features is different, use one Conv1D layers with batch normalization to change the number of features and then sum.

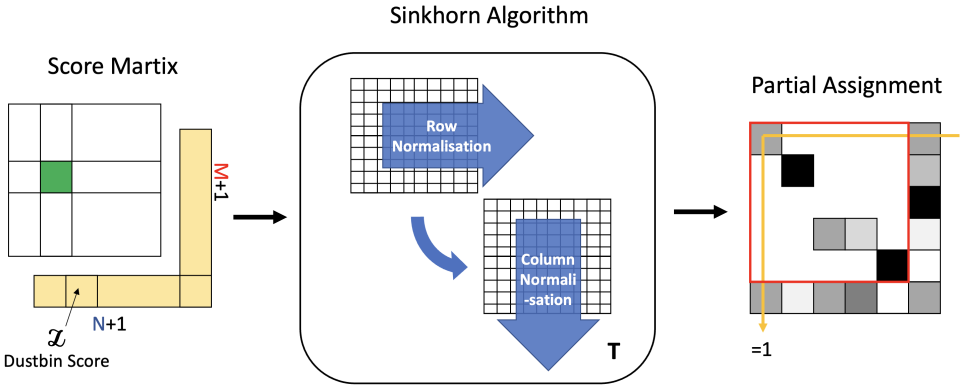


Figure A.9.: Illustration of the Optimal Transport layer processing [332]

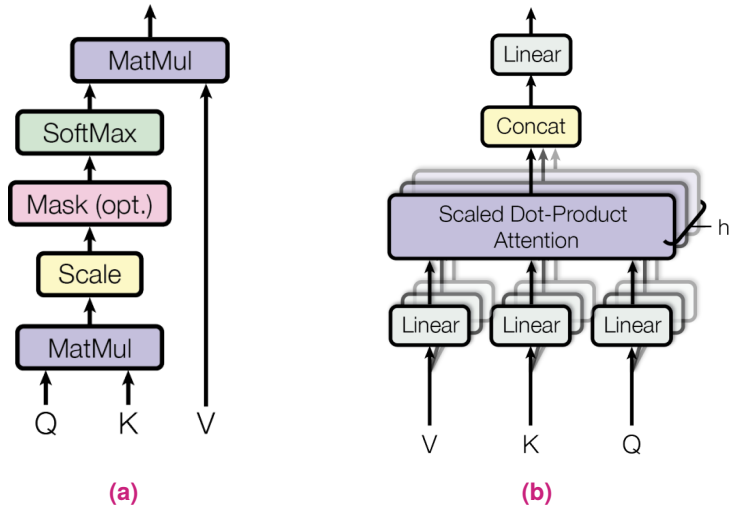


Figure A.10.: Architecture of the scaled dot product attention(left) and the multi-head attention(right) [333]

The loss function to train SOMA model is as shown in equation A.6. $L = c_l L_A \sum c_{reg} L_{reg}$. L_A is the loss of the assignment matrix and L_{reg} is the loss of L_2 regularization on the model parameters. Shown as Equation A.5 and Equation A.6 [237]

$$L_A = \frac{-1}{\sum_{i,j} G'_{i,j}} \sum_{i,j} W_{i,j} \cdot G'_{i,j} \cdot log(A'_{i,j}) \tag{A.5}$$

$$L_{reg} = \|\phi\|_2^2 \tag{A.6}$$

For detailed understanding of SOMA training procedure, readers can refer [237] for more details.

A.6 PointNet++

Compared with PointNet [256], PointNet++ [257] has made great progress in many aspects. PointNet[256] only uses local features in the segmentation task, and only considers the features of each point itself, ignoring the relationship between points and neighbors. PointNet++ [257] effectively extracts local features through sampling and grouping. Although the global features obtained by max pooling are simple, it causes information loss. It uses the hierarchical feature learning framework to obtain local-global features of different scales. Different from directly connecting point features and global features, it uses the encoder-decoder structure for segmentation tasks, and uses skip connections to concatenate the local-global features of the corresponding layers, making the generated features more distinctive. The hierarchical feature learning architecture of PointNet++ [257] is shown in Figure A.11.

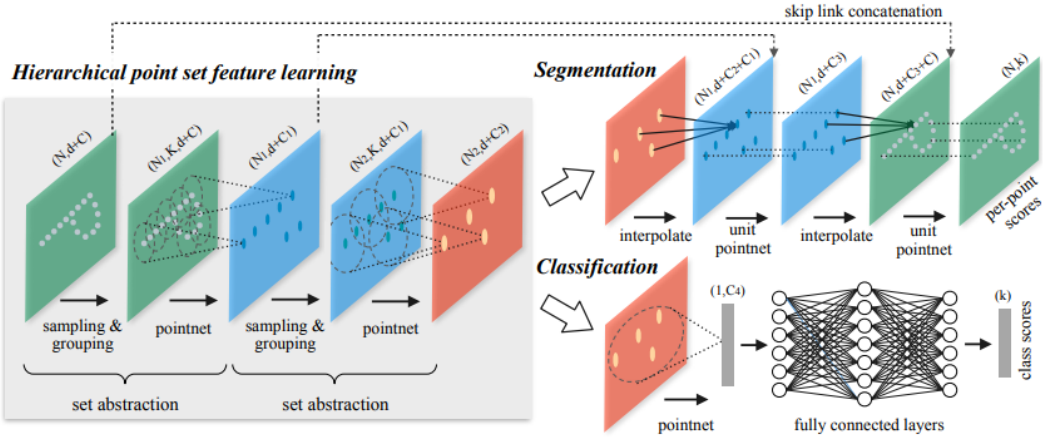


Figure A.11.: The hierarchical feature learning architecture of PointNet++ [257]

We can generally regard the network structure of PointNet++ as an encoder-decoder structure. Encoder is a downsampling process, using multiple set abstraction modules to downsample the input point cloud at multiple levels. The set abstraction module consists of three modules: sampling, grouping, and pointnet. The Sampling module uses the farthest point sampling (FPS) method to downsample the point cloud. FPS can be understood as making the sampling points as far as possible. The advantage of this sampling is that the downsampling results can be more uniform. We can also treat the N_1 downsampled points as key points found from the original point cloud with N points. The grouping module is to take each key point as the center, find K neighbor points, and form a patch, that is to say, a total of N_1 patches are generated. Different methods can be used to find neighbor points, such as K-nearest neighbors and query ball. And query ball is more suitable for local recognition, because it can limit the distance from the neighbor point to the key point. Both the sample and group modules operate on point coordinates, so if there is still feature information, the coordinates and features can be concatenated. The PointNet module needs to perform convolution and pooling operations on the previously obtained features.

PointNet++ [257] also considers the problem of non-uniform sampling density that may occur for sparse point clouds. It proposes two solutions, multi-scale grouping(MSG) and multi-resolution grouping(MRG). MSG is to take query balls of different radius for each key point, and obtain multiple concentric balls of different sizes, that is, multiple patches with the same center but different scales, extract features from these patches, and concatenate them. MRG concats two kinds of features, one of which is to aggregate the features of the patches of the previous set abstraction, and the other is to directly extract the features of the current patch through a single PointNet module.

Decoder adopts different structures for classification and segmentation tasks as needed. The classification task decoder is to send the global feature obtained by the encoder into the fully connected network, and finally obtain classification scores through a softmax function. The Segmentation task decoder realizes upsampling through interpolation and skip connection, and finally obtains per-point scores.

Name	No. of Filters	Kernel Size	Stride	Output
ResNet-101	–	–	–	$15 \times 48 \times 48$
Conv-1	64	4×4	2	$64 \times 24 \times 24$
Conv-bn-relu	64	4×4	1	$64 \times 24 \times 24$
Linear-3(z)	–	–	–	20

Table A.1.: Heatmap Encoder network architecture of EgoposeNet. The encoder receives $15 \times 48 \times 48$ 2D heatmaps from ResNet-101. It uses three convolution layers, each followed by batch normalization, leaky ReLU with a negative slope of 0.2, and a dropout layer with a 0.5 probability. The output after convolution 1 is $64 \times 24 \times 24$, convolution 2 is $128 \times 12 \times 12$, and convolution 3 is $512 \times 6 \times 6$. Afterward, a flattened layer is applied followed by a fully connected layer to generate a latent z of size 20.

Name	No. of Filters	Kernel Size	Stride	Output
WASP	–	–	–	$256 \times 23 \times 23$
ResNet-101-low-level	–	–	–	$256 \times 92 \times 92$
MaxPool1(low-level)	–	3×3	2	$48 \times 46 \times 46$
Bilinear-Intp(low-level)	–	–	–	$256 \times 46 \times 46$
Concat(low-level,WASP)	–	–	–	$304 \times 46 \times 46$
Conv-1	256	3×3	1	$256 \times 23 \times 23$
Conv-bn-relu	256	1×1	1	$256 \times 23 \times 23$
Conv-2	256	3×3	1	$256 \times 23 \times 23$
Conv-bn-relu	256	1×1	1	$256 \times 23 \times 23$
Conv-3	22	1×1	1	$16 \times 46 \times 46$

Table A.2.: Heatmap Decoder branch network architecture of EgoposeNet. The decoder gets $256 \times 23 \times 23$ feature map from the WASP module and $256 \times 92 \times 92$ low-level features from layer 1 of ResNet-101. MaxPool1 and Bilinear Interpolation is performed on low-level features followed by a concatenation of low-level features and WASP features and three convolution layers. The final output is $16 \times 46 \times 46$ 2D heatmaps.

Name	No. of Filters	Kernel Size	Stride	Output
WASP	–	–	–	$256 \times 23 \times 23$
ResNet-101-low-level	–	–	–	$256 \times 92 \times 92$
MaxPool1(low-level)	–	3×3	2	$48 \times 46 \times 46$
Bilinear-Intp(low-level)	–	–	–	$256 \times 46 \times 46$
Concat(low-level,WASP)	–	–	–	$304 \times 46 \times 46$
Conv-1	256	3×3	1	$256 \times 23 \times 23$
Conv-bn-relu	256	1×1	1	$256 \times 23 \times 23$
Conv-2	256	3×3	1	$256 \times 23 \times 23$
Conv-bn-relu	256	1×1	1	$256 \times 23 \times 23$
Conv-3	22	1×1	1	$22 \times 46 \times 46$
MaxPool2	–	3×3	2	$22 \times 23 \times 23$
flatten	–	–	–	11638
Linear-1	–	–	–	1000
Linear-2	–	–	–	500
Linear-3	–	–	–	48

Table A.3.: Pose Decoder branch network architecture of EgoposeNet. The decoder gets $256 \times 23 \times 23$ feature map from the WASP module and $256 \times 92 \times 92$ low-level features from layer 1 of ResNet-101. MaxPool1 and Bilinear Interpolation is performed on low-level features followed by a concatenation of low-level features and WASP features and three convolution layers. Further MaxPool2 layer is applied followed by flattened layer to generate 11638 output. A fully connected layer downsamples 11638 to the final output of 48 which signifies 3D joint positions of 16 joints.

A.7 EgoPoseNet

The Table A.1 and Table A.2 show the network architecture details of 2D heatmap encoder and 2D heatmap decoder of the triple network branch respectively. The architecture of the Pose Decoder(see table A.3) and Angle Decoder are the same, except for the last linear layer. The last layer of the pose decoder is replaced with another linear layer which outputs 96 neurons. The 96 is 16×6 6D rotation representation, while the output 48 by pose decoder is 16×3 3D joint positions.

A.8 WASP

The Waterfall Atrous Spatial Pooling architecture (WASP) module allows the backbone to capitalize on both the increased FOV associated with atrous or dilated convolutions and the efficiency afforded by the cascade approach. Structurally, WASP comprises four branches arranged in a waterfall-like pattern, each possessing distinct FOVs as illustrated in Figure A.12. The atrous convolutions within WASP initiate at a low rate of 6 and progressively increase in succeeding branches with rates of 6, 12, 18, and 24. The utilization of smaller filter sizes in this design contributes to increased efficiency. Moreover,

each branch generates multiscale characteristics that are subsequently merged to produce a more comprehensive representation.

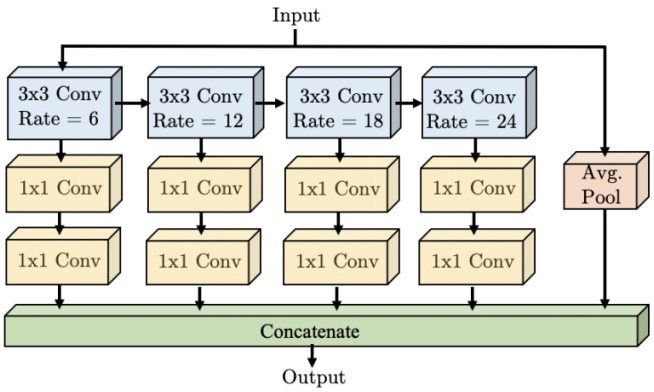


Figure A.12.: The Waterfall Atrous Spatial Pooling Architecture as presented by [295].

Datasets

B

A large corpus of data is required for deep learning models to generalize well. There has been surge of pose estimation datasets. The Table B.1 provides an brief overview of human pose and action datasets describing various pose annotations and input modalities.

Dataset	Images	Video	Multi Person	Multi View	Cam Type	2D KPS	3DP KPS	3D Rots	Body Shape	IR	Action	Dataset Type
LSP [334]	10,000	-	-	-	RGB	14	-	-	-	-	-	Real
PENN [335]	163,841	2,326	-	-	RGB	13	-	15	-	-	-	Real
MPII [176]	28,883	-	Yes	-	RGB	16	-	-	-	-	-	Real
FLIC [336]	15,247	-	Yes	-	RGB	7	-	-	-	-	-	Real
COCO [101]	3,30,000	-	Yes	-	RGB	17	-	-	-	-		Real
OP_Foot[337]	14,000	-	-	-	RGB	6	-	-	-	-		Real
DSV [309]	11,219	100	-	-	RGB	16	-	-	-	-	-	Real
IAT [309]	≈6000	108	-	-	RGB	16	-	-	-	-	-	Real
Sports Cap [154]	41,749	740	-	-	RGB	25	-	-	-	-	-	Real
3DPW [338]	53,796	60	Yes	-	RGB	24	24	21	Yes	-	-	Real
H36M [102]	3,600,000	720	-	4	RGB	32	32	-	-	-	16	Real
MPI-INF-3DHP [104]	1,837,216	87	-	14	RGB	28	28	-	-	-	-	Real
TotalCapture [103]	1,600,000	298	-	8	RGB	27	27	-	-	-	-	Real
MuPOTs-3D [20]	493,841	20	Yes	14	RGB	17	17	-	-	-	-	Real
ITOP [339]	50,000	300	-	2	Depth	15	15	-	-	-	15	Real
k2hpd [340]	100,000	300	-	-	Depth	19	-	-	-	-	10	Real
NTU RGB+D[341]	-	114480	Yes	3	RGBD	25	25	21	-	-	120	Real
Baracca [342]	>9000	240	-	2	RGBD	15	15	-	-	Yes	8	Real
DriveAndAct [343]	-	29	-	5	RGBD	26	26	-	-	Yes	81	Real
DREICAP-3D	-	337	-	-	RGBD	17	17	17	Yes	Yes	37	Real
TiCAM-Real [231]	-	110	Yes	-	RGBD	-	-	-	-	Yes	Yes	Real
TiCAM-Syn [231]	-	1	Yes	-	RGBD	52	-	-	-	Yes	Yes	Synthetic
SVIRO [232]	-	8	Yes	-	RGBD	52	-	-	-	Yes	Yes	Synthetic
JTA [344]	50000	512	Yes	-	RGBD	22	22	-	-	-	-	Synthetic
Surreal [345]	6,000,000	-	-	-	RGBD	24	-	-	Yes	-	-	Synthetic
Mo2Cap2 [298]	535,000	-	-	-	RGBD	15	15	-	-	-	-	Synthetic
Xr-EgoPose [71]	380,000	-	-	-	RGBD	64	64	51	-	-	-	Synthetic
Unreal-Ego [346]	450,000	4839	-	2	RGB	32	32	-	-	-	-	Synthetic
EgoCap [347]	100,000	-	-	2	RGB	17	17	-	-	-	-	Real
EgoGlass [348]	170,000	-	-	2	RGB	13	13	-	-	-	-	Real
EgoPW [349]	318,000	97	-	2	RGB	17	17	-	-	-	20	Real
EgoFish3D [350]	75,000	30	-	-	RGB	15	15	-	-	-	10	Real
Diving48 [152]	-	18000	-	-	RGB	-	-	-	-	-	48	Real
HMDB51 [351]	-	6766	-	-	RGB	-	-	-	-	-	51	Real
Kinectics [352]	-	500,000	-	-	RGB	-	-	-	-	-	600	Real
UCF-101 [353]	-	13320	-	-	RGB	-	-	-	-	-	101	Real

Table B.1.: The table provides overview of different human pose and activity datasets used in the literature.

Furthermore, the joints annotated across the pose datasets vary in terms of number of joints and the skeletal joint definition. A considerable effort was made to align different datasets into a common skeleton reference.

Joint Rotation Representation

C

In this chapter, we briefly cover the different rotation parameters commonly used in the literature. The 3D rotations representation provide a concise and precise means to represent the spatial orientation of objects. The accurate representation of 3D rotations is fundamental across numerous fields. In computer graphics, it ensures the realistic rendering of scenes,objects and animations in virtual reality. In robotics, precise representation facilitates robot manipulations, path planning, and control, enabling robots to navigate and interact effectively in their environments. Further, the representations also play a vital role in computer vision for different vision tasks such as object detection, tracking, and pose estimation.

The following subsections provides an overview of each representation.

C.1 Rotation Matrix

It captures the rotation of the vector from frame u to v. It is represented as follows:

$$R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}, \tag{C.1}$$

where $R \in \mathbb{R}^{3\times3} \in SO(3)$, $SO(3)$ is defined as the special orthogonal group in 3D. Rotation matrix has unique representation but this representation lacks the geometric intuition. It also needs 9 components to represent 3 DoF which is inefficient.

C.2 Euler angles

Euler angles define the rotation as a sequence of 3 consecutive rotations around 3 predefined axes. There are different possible orders for the rotations. It is efficient in terms of parameters because it only needs 3 parameters, the angles around the predefined axes. This representation captures the intuitive geometric interpretation, for instance if we consider that the predefined axes are aligned with the major joint axes, the angles provide the joint angles. However, euler angles are not unique and suffer from gimbal lock. In gimbal lock, there is a loss of one DoF if two rotation axes become parallel to each other.

Euler angle representation for 3D rotations can be expressed as follows:

$$R = R_z(\psi) \cdot R_y(\theta) \cdot R_x(\phi)$$

where:

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

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

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

And R is the resulting rotation matrix, representing the combined rotation about the X, Y, and Z axes.

C.3 Unit Quaternions

A unit quaternion is a 4-dimensional orientation representation expressed as follows:

$$q = \begin{pmatrix} q_w \\ q_x \\ q_y \\ q_z \end{pmatrix} = q = \begin{pmatrix} q_w \\ q_v \end{pmatrix}, \tag{C.2}$$

where $q \in \mathbb{R}^4$ and $||q|| = 1$.

This representation does not have singularities. However, this has 4 parameters which is taken care by adding an additional non-linear constraint, $||q|| = 1$.

C.4 Axis angle

This representation expresses 3D rotation by an angle θ and a rotation vector $n \in \mathbb{R}^3, ||n|| = 1$. The representation is expressed as follows:

$$\eta := \theta n \in \mathbb{R}^3. \tag{C.3}$$

This representation has minimal number of 3 parameters. However, the representation is not unique and also suffers from singularity (case when $\theta = 0°$, axis is not defined in this case.)

C.5 Exponential Map

The Exponential map is used to represent 3D rotations as elements of a Lie group, such as the Special Orthogonal group $SO(3)$ [116]. In this representation, a rotation matrix R can be parameterized by a vector ω in 3D space, where ω represents the axis of rotation scaled by the rotation angle θ. The Exponential map for 3D rotations takes this axis-angle representation and maps it to a rotation matrix. Mathematically, given a rotation axis ω and an angle θ, the Exponential map can be expressed as $R = \exp(\theta\hat{\omega})$, where $\hat{\omega}$ is the skew-symmetric matrix associated with the vector ω. This representation allows for efficient manipulation and interpolation of rotations in 3D space, making it particularly useful in applications such as computer graphics, robotics, and computer vision.

Let $\mathbf{v} = (v_1, v_2, v_3)$ be a unit vector representing the axis of rotation, and let θ be the angle of rotation about that axis. The exponential map $\text{Exp}(\cdot)$ maps this axis-angle representation to a rotation matrix $\mathbf{R}$ as follows:

$$\mathbf{R} = \text{Exp}(\mathbf{v}, \theta)$$

The exponential map is defined using Rodrigues' rotation formula:

$$\text{Exp}(\mathbf{v}, \theta) = \mathbf{I} + \sin(\theta)\mathbf{A} + (1 - \cos(\theta))\mathbf{A}^2$$

where: $\mathbf{I}$ is the 3×3 identity matrix, $\mathbf{A}$ is the skew-symmetric matrix associated with $\mathbf{v}$, defined as:

$$\mathbf{A} = \begin{bmatrix} 0 & -v_3 & v_2 \\ v_3 & 0 & -v_1 \\ -v_2 & v_1 & 0 \end{bmatrix}$$

The resulting rotation matrix $\mathbf{R}$ represents the rotation about the axis $\mathbf{v}$ by an angle θ.

The exponential map offers several advantages across mathematics and its applications. Primarily, it efficiently represents small-angle rotations, particularly in 3D spaces, by parameterizing rotation matrices using axis-angle parameters, facilitating straightforward manipulation and interpolation of rotations, vital in fields like computer graphics and robotics.

C.6 6D Rotation Representation

Authors of [354] argue that all representations of 3D rotations such as Quaternions and Euler angles are discontinuous in real Euclidean space of four or fewer dimensions, which is difficult for neural networks to learn. They propose a more continuous representation for 3D rotation in 6D.

A neural network typically has continuous units. Neural networks learn a representation space R which is then mapped to the original space X and loss is computed between

this original space and input. This is shown in figure C.1. A mapping function f maps from intermediate representation space R to original space X as, $f : R \to X$ and g maps from X to R as, $g : X \to R$. To define a representation between R and X, f must be a left inverse function of g, that is: $\forall x \to X, f(g(x)) = x$ and if g is continuous then this representation also becomes continuous.

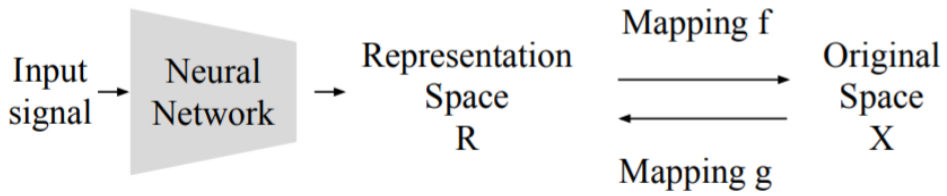


Figure C.1.: Definition of Continuous Representation and its relation with neural network. [354]

To give an example of discontinuity, consider a matrix M of special orthogonal group $SO(n)$. M satisfies the property of $SO(n)$:

1. $MM^T = M^T M = I$
2. $det(M) = 1$

For 2D rotation, $M \to SO(2)$ and can be defined as:

$$\begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ sin(\theta) & \cos(\theta) \end{bmatrix}. \tag{C.4}$$

Here, $\theta \to R$, where R belongs to a set of angles, suppose, $R = [0, 2\pi]$. Now as aforementioned, g represents a mapping from original space of $SO(2)$ to the angular representation space R. And this mapping is a discontinuous for zero rotation (identity matrix), where M becomes:

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \longrightarrow \begin{bmatrix} \cos(0) & -\sin(2\pi) \\ \sin(0) & \cos(2\pi) \end{bmatrix}. \tag{C.5}$$

The limit of g is undefined, giving two values of angles: 0 and 2π respectively. The same can be seen in figure C.2. However, if the matrix $M \to SO(2)$ is represented by its first column vector $[\cos(\theta), \sin(\theta)]^T$, then the representation becomes continuous. Similarly, for 3D rotations, if the matrix $M \to SO(3)$ is defined by its first two column vectors, the 3D rotation is defined by a 6D continuous representation.

Euler angles exhibit a discontinuity where the identity rotation I occurs, with one-directional limit at $\theta = 0$ and the other at $\theta = 2\pi$. Similarly, quaternions and axis angles encounter a break in continuity at 180-degree rotations. To address this, we implement an orthogonalization process directly within the representation.

In the realm of continuous representation, involving $n \times (n-1)$ dimensions for n-dimensional rotations, we define the original space X as $SO(n)$, and the representation

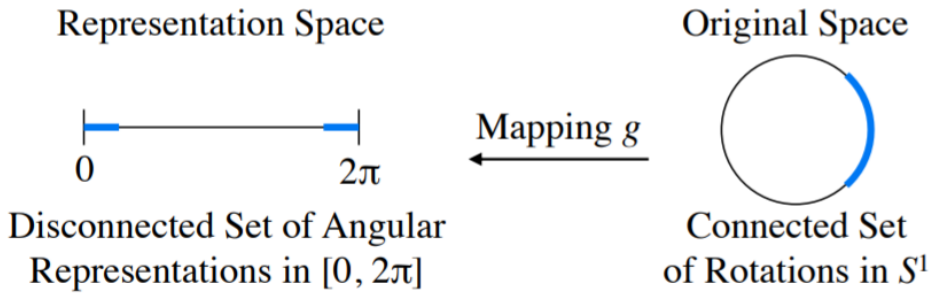


Figure C.2.: Visual representation of 2D rotation from original space in SO(2) to angular representation space. [354]

space as $R = R^{n\times(n-1)}$. A mapping g_{GS} to the representation space is introduced, simplifying the process by discarding the final column vector of the input matrix, as expressed in Equation C.6.

The Gram-Schmidt-like mapping f_{GS} to the original space is illustrated in Equation C.7, where a_i and b_i are column vectors for $i = 1, 2, ..., n$. This process aids in achieving a more coherent and continuous representation of rotations.

$$g_{GS}(\begin{bmatrix} | & | & | \\ a_1 & .. & a_n \\ | & | & | \end{bmatrix}) = \begin{bmatrix} | & | & | \\ a_1 & .. & a_{n-1} \\ | & | & | \end{bmatrix} \tag{C.6}$$

$$f_{GS}(\begin{bmatrix} | & | & | \\ a_1 & .. & a_{n-1} \\ | & | & | \end{bmatrix}) = \begin{bmatrix} | & | & | \\ b_1 & .. & b_n \\ | & | & | \end{bmatrix} \tag{C.7}$$

$$b_i = \begin{bmatrix} N(a_i) & if(i = 1) \\ N(a_i - y) & if(2 \leq i < n) \\ z & if(i = n) \end{bmatrix}^T \tag{C.8}$$

, where $y = \sum_{j=1}^{i-1}(b_j.a_i)b_j$ and $z = det \begin{bmatrix} | & | & | & e_1 \\ b_1 & .. & b_n & | \\ | & | & | & e_n \end{bmatrix}$

Here, the function $N()$ signifies a normalization function, and $e_1, ..., e_n$ represent the n canonical basis vectors of the Euclidean space.

With n=3, the mapping g_{GS} makes continuous representation at $n \times (n-1)$ equals 6 dimensions.

Evaluation Metrics

D

Evaluation metrics play a crucial role in quantitatively assessing the performance of neural networks. In the context of human pose estimation, several standard metrics are commonly employed. In this chapter, we delve into the evaluation metrics used for this purpose, namely Mean Pixel Error (MPE), Percentage of Correct Key-points (PCK), Mean Per Joint Position Error (MPJPE), Procrustes Analysis Mean Per Joint Position Error (PA-MPJPE), Angle Error (AE), and Inference Time.

D.1 Mean Pixel Error (MPE)

Mean Pixel Error is a metric utilized to gauge the accuracy of 2D joint positions in pixels. It represents the average keypoint error across all joints and poses. The Euclidean distance between the predicted and ground truth points is calculated for each joint. Given n frames, k joints, $y \in \mathbb{R}^{k\times 3}$ as ground truth joint positions, and $\hat{y} \in \mathbb{R}^{k\times 3}$ as predicted joint positions, the MPE is computed as follows:

$$MPE(\text{pixels}) = \frac{1}{n}\frac{1}{k}\sum_{i=1}^{n}\sum_{j=1}^{k}||\hat{y}_{i,j} - y_{i,j}||_2 \tag{D.1}$$

D.2 Percentage of Correct Key-points (PCK)

The percentage of correct key points is used to measure the accuracy of pose estimation models in the pixel space. The metric indicates the proportion of keypoints that are correctly identified by the algorithm out of all the keypoints detected. A higher percentage of correct keypoints generally indicates better performance of the feature detection algorithm.

The variant of the measure PCK is PCK-*h*, where h is typically a threshold distance that determines whether a keypoint is considered correctly localized or not. If the distance between a detected keypoint and its ground truth position is within h, it is counted as a correct detection; otherwise, it is counted as an incorrect detection. The formula for PCK_h can be expressed as:

$$\text{PCK}_h = \frac{N_{\text{correct}}}{N_{\text{total}}} \times 100\%$$

Where: - N_{total} is the total number of keypoints detected by the algorithm. - N_{correct} is the number of keypoints whose distance from the ground truth is less than or equal to h.

D.3 Mean Per Joint Position Error (MPJPE)

This is one of the widely used evaluation metric for 3D HPE, which defines the euclidean distance between the estimated 3D joints and ground truth 3D joints, averaged over all joints in millimeters(mm). For a set of frames, it is averaged over the MPJPEs of all frames. The formulation to define MPJPE is given as:

$$E_{MPJPE}(f,S) = \frac{1}{N_S}\sum_{i=1}^{N_S} ||m^{(f)}_{\mathbf{f},S}(i) - m^{(f)}_{\mathbf{gt},S}(i)||_2, \tag{D.2}$$

where, N_S is the number of joints in skeleton S with m_{gt} as ground-truth 3D joint positions and m_f as predicted 3D joint positions.

D.4 Mean Per Joint Position Error after Procrustes Alignment (PAMPJPE)

Procrustes Analysis is performed in the post-processing step, which computes the error after rigid transformation between the predictions and ground-truth. Procrustes alignment performs the similarity transform to align the predicted 3D joints to a common frame. It involves three steps: scaling, translation and rotation such that there is a minimum sum of squared distance between corresponding predicted and ground-truth positions. In many literature, this is also called as reconstruction error.

D.5 Angle error

To compute the difference in angular representation, geodesic error is used. This expresses the minimum angular difference between two rotations in axis-angle form, as defined below:

$$AE = cos^{-1}((M^{''}_{00} + M^{''}_{11} + M^{''}_{22} - 1)/2), \tag{D.3}$$

where, $M^{''} = MM^{'-1}$. M and $M^{'}$ are the corresponding input matrix and output matrix.

D.6 Inference Time

Inference time measures the runtime of models, both before and after compression, on mobile devices. It is recorded in milliseconds.

References

[1] Eadweard Muybridge. *The human figure in motion*. Courier Corporation, 2012.

[2] Gunnar Johansson. “Visual perception of biological motion and a model for its analysis”. In: *Perception & psychophysics* 14 (1973), pp. 201–211.

[3] David J Sturman. “A brief history of motion capture for computer character animation”. In: *SIGGRAPH94, Course9* (1994).

[4] D Gordon E Robertson, Graham E Caldwell, Joseph Hamill, Gary Kamen, and Saunders Whittlesey. *Research methods in biomechanics*. Human kinetics, 2013.

[5] Bruce Guenther Baumgart. *Geometric modeling for computer vision.* Stanford University, 1974.

[6] Aristides G Requicha. “Representations for rigid solids: Theory, methods, and systems”. In: *ACM Computing Surveys (CSUR)* 12.4 (1980), pp. 437–464.

[7] Michael Girard and Anthony A Maciejewski. “Computational modeling for the computer animation of legged figures”. In: *ACM SIGGRAPH Computer Graphics* 19.3 (1985), pp. 263–270.

[8] Ronan Boulic, Nadia Magnenat Thalmann, and Daniel Thalmann. “A global human walking model with real-time kinematic personification”. In: *The visual computer* 6 (1990), pp. 344–358.

[9] Jianmin Zhao and Norman I Badler. “Inverse kinematics positioning using nonlinear programming for highly articulated figures”. In: *ACM Transactions on Graphics (TOG)* 13.4 (1994), pp. 313–336.

[10] Giacomo Rizzolatti, Luciano Fadiga, Vittorio Gallese, and Leonardo Fogassi. “Premotor cortex and the recognition of motor actions”. In: *Cognitive brain research* 3.2 (1996), pp. 131–141.

[11] Timothy F Cootes, Christopher J Taylor, David H Cooper, and Jim Graham. “Active shape models-their training and application”. In: *Computer vision and image understanding* 61.1 (1995), pp. 38–59.

[12] Tim Cootes, ER Baldock, and J Graham. “An introduction to active shape models”. In: *Image processing and analysis* 328 (2000), pp. 223–248.

[13] Timothy F Cootes, Gareth J Edwards, and Christopher J Taylor. “Active appearance models”. In: *Computer Vision—ECCV’98: 5th European Conference on Computer Vision Freiburg, Germany, June 2–6, 1998 Proceedings, Volume II 5*. Springer. 1998, pp. 484–498.

[14] Jialue Fan, Wei Xu, Ying Wu, and Yihong Gong. “Human tracking using convolutional neural networks”. In: *IEEE transactions on Neural Networks* 21.10 (2010), pp. 1610–1623.

[15] Shuiwang Ji, Wei Xu, Ming Yang, and Kai Yu. “3D convolutional neural networks for human action recognition”. In: *IEEE transactions on pattern analysis and machine intelligence* 35.1 (2012), pp. 221–231.

[16] Alexander Toshev and Christian Szegedy. “Deeppose: Human pose estimation via deep neural networks”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2014, pp. 1653–1660.

[17] Federica Bogo, Angjoo Kanazawa, Christoph Lassner, Peter Gehler, Javier Romero, and Michael J Black. “Keep it SMPL: Automatic estimation of 3D human pose and shape from a single image”. In: *European Conference on Computer Vision*. Springer. 2016, pp. 561–578.

[18] Angjoo Kanazawa, Michael J. Black, David W. Jacobs, and Jitendra Malik. “End-to-end Recovery of Human Shape and Pose”. In: *Computer Vision and Pattern Regognition (CVPR)*. 2018.

[19] Bruce Xiaohan Nie, Ping Wei, and Song-Chun Zhu. “Monocular 3D human pose estimation by predicting depth on joints”. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, pp. 3447–3455.

[20] Dushyant Mehta, Oleksandr Sotnychenko, Franziska Mueller, Weipeng Xu, Srinath Sridhar, Gerard Pons-Moll, and Christian Theobalt. “Single-Shot Multi-Person 3D Pose Estimation From Monocular RGB”. In: *3D Vision (3DV), 2018 Sixth International Conference on*. IEEE. Sept. 2018.

[21] Muhammed Kocabas, Nikos Athanasiou, and Michael Black. “VIBE: Video Inference for Human Body Pose and Shape Estimation”. In: 2020, pp. 5252–5262.

[22] Gyeongsik Moon and Kyoung Mu Lee. “I2L-MeshNet: Image-to-Lixel Prediction Network for Accurate 3D Human Pose and Mesh Estimation from a Single RGB Image”. In: *ArXiv* abs/2008.03713 (2020).

[23] Andrew Garbett, Ziedune Degutyte, James Hodge, and Arlene Astell. “Towards understanding people’s experiences of ai computer vision fitness instructor apps”. In: *Designing interactive systems conference 2021*. 2021, pp. 1619–1637.

[24] Amit Nagarkoti, Revant Teotia, Amith K Mahale, and Pankaj K Das. “Realtime indoor workout analysis using machine learning & computer vision”. In: *2019 41st Annual international conference of the IEEE engineering in medicine and biology society (EMBC)*. IEEE. 2019, pp. 1440–1443.

[25] Junfeng Gao, Yong Yang, Pan Lin, Dong Sun Park, et al. *Computer vision in healthcare applications*. 2018.

[26] Lucas Mourot, Ludovic Hoyet, François Le Clerc, François Schnitzler, and Pierre Hellier. “A Survey on Deep Learning for Skeleton-Based Human Animation”. In: *Computer Graphics Forum*. Vol. 41. 1. Wiley Online Library. 2022, pp. 122–157.

[27] Pei Xu and Ioannis Karamouzas. “A GAN-Like Approach for Physics-Based Imitation Learning and Interactive Character Control”. In: *Proceedings of the ACM on Computer Graphics and Interactive Techniques* 4.3 (2021), pp. 1–22.

[28] Ali Ahmad Malik, Tariq Masood, and Arne Bilberg. “Virtual reality in manufacturing: immersive and collaborative artificial-reality in design of human-robot workspace”. In: *International Journal of Computer Integrated Manufacturing* 33.1 (2020), pp. 22–37.

[29] Dejun Zhang, Yiqi Wu, Mingyue Guo, and Yilin Chen. “Deep learning methods for 3D human pose estimation under different supervision paradigms: a survey”. In: *Electronics* 10.18 (2021), p. 2267.

[30] Jinbao Wang, Shujie Tan, Xiantong Zhen, Shuo Xu, Feng Zheng, Zhenyu He, and Ling Shao. “Deep 3D human pose estimation: A review”. In: *Computer Vision and Image Understanding* 210 (2021), p. 103225.

[31] Ce Zheng, Wenhan Wu, Chen Chen, Taojiannan Yang, Sijie Zhu, Ju Shen, Nasser Kehtarnavaz, and Mubarak Shah. “Deep learning-based human pose estimation: A survey”. In: *ACM Computing Surveys* 56.1 (2023), pp. 1–37.

[32] Tewodros Legesse Munea, Yalew Zelalem Jembre, Halefom Tekle Weldegebriel, Longbiao Chen, Chenxi Huang, and Chenhui Yang. “The progress of human pose estimation: A survey and taxonomy of models applied in 2D human pose estimation”. In: *IEEE Access* 8 (2020), pp. 133330–133348.

[33] Shradha Dubey and Manish Dixit. “A comprehensive survey on human pose estimation approaches”. In: *Multimedia Systems* 29.1 (2023), pp. 167–195.

[34] Zhihan Lv, Fabio Poiesi, Qi Dong, Jaime Lloret, and Houbing Song. “Deep learning for intelligent human–computer interaction”. In: *Applied Sciences* 12.22 (2022), p. 11457.

[35] Christian Keilstrup Ingwersen, Christian Møller Mikkelstrup, Janus Nørtoft Jensen, Morten Rieger Hannemose, and Anders Bjorholm Dahl. “SportsPose - A Dynamic 3D Sports Pose Dataset”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*. June 2023, pp. 5219–5228.

[36] Zhanyuan Huang, Yang Liu, Yajun Fang, and Berthold KP Horn. “Video-based fall detection for seniors with human pose estimation”. In: *2018 4th international conference on Universal Village (UV)*. IEEE. 2018, pp. 1–4.

[37] Debasmita Mukherjee, Kashish Gupta, Li Hsin Chang, and Homayoun Najjaran. “A survey of robot learning strategies for human-robot collaboration in industrial settings”. In: *Robotics and Computer-Integrated Manufacturing* 73 (2022), p. 102231.

[38] Yang Xing, Chen Lv, Dongpu Cao, and Peng Hang. “Toward human-vehicle collaboration: Review and perspectives on human-centered collaborative automated driving”. In: *Transportation research part C: emerging technologies* 128 (2021), p. 103199.

[39] Xiaochen Liu, Yurong Jiang, Kyu-Han Kim, and Ramesh Govindan. “Grab: Fast and accurate sensor processing for cashier-free shopping”. In: *arXiv preprint arXiv:2001.01033* (2020).

[40] Haoye Dong, Xiaodan Liang, Xiaohui Shen, Bochao Wang, Hanjiang Lai, Jia Zhu, Zhiting Hu, and Jian Yin. “Towards multi-pose guided virtual try-on network”. In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2019, pp. 9026–9035.

[41] Yann Desmarais, Denis Mottet, Pierre Slangen, and Philippe Montesinos. “A review of 3D human pose estimation algorithms for markerless motion capture”. In: *Computer Vision and Image Understanding* 212 (2021), p. 103275.

[42] Helge Rhodin, Jörg Spörri, Isinsu Katircioglu, Victor Constantin, Frédéric Meyer, Erich Müller, Mathieu Salzmann, and Pascal Fua. “Learning monocular 3d human pose estimation from multi-view images”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 8437–8446.

[43] OptiTrack. *OptiTrack*. https://optitrack.com/. Accessed on December 10, 2023.

[44] Vicon. *Vicon*. https://www.vicon.com/. Accessed on December 10, 2023.

[45] QualiSys. *Qualisys*. https://www.qualisys.com/. Accessed on December 10, 2023.

[46] Thei. *Thei Markerless Motion Capture*. https://www.theiamarkerless.ca/. Accessed on December 10, 2023.

[47] SciTrack. *SciTrack*. http://sci-track.com/. Accessed on December 10, 2023.

[48] Yinghao Huang, Manuel Kaufmann, Emre Aksan, Michael J Black, Otmar Hilliges, and Gerard Pons-Moll. “Deep inertial poser: Learning to reconstruct human pose from sparse inertial measurements in real time”. In: *ACM Transactions on Graphics (TOG)* 37.6 (2018), pp. 1–15.

[49] Liujun Liu, Jiewen Yang, Ye Lin, Peixuan Zhang, and Lihua Zhang. “3D human pose estimation with single image and inertial measurement unit (IMU) sequence”. In: *Pattern Recognition* (2023), p. 110175.

[50] Fuyang Huang, Ailing Zeng, Minhao Liu, Qiuxia Lai, and Qiang Xu. “DeepFuse: An IMU-aware network for real-time 3D human pose estimation from multi-view image”. In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. 2020, pp. 429–438.

[51] Jinbao Wang, Shujie Tan, Xiantong Zhen, Shuo Xu, Feng Zheng, Zhenyu He, and Ling Shao. “Deep 3D human pose estimation: A review”. In: *Computer Vision and Image Understanding* 210 (2021), p. 103225. ISSN: 1077-3142.

[52] FreePik. *FreePik Images*. https://freepik.com/. Accessed on December 10, 2023.

[53] Laurie Needham, Murray Evans, Darren P Cosker, Logan Wade, Polly M McGuigan, James L Bilzon, and Steffi L Colyer. “The accuracy of several pose estimation methods for 3D joint centre localisation”. In: *Scientific reports* 11.1 (2021), p. 20673.

[54] Run Zhou Ye, Arun Subramanian, Daniel Diedrich, Heidi Lindroth, Brian Pickering, and Vitaly Herasevich. “Effects of image quality on the accuracy human pose estimation and detection of eye lid opening/closing using OpenPose and DLib”. In: *Journal of Imaging* 8.12 (2022), p. 330.

[55] Timo Von Marcard, Roberto Henschel, Michael J Black, Bodo Rosenhahn, and Gerard 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)*. 2018, pp. 601–617.

[56] Yiming Zhao, Denys Rozumnyi, Jie Song, Otmar Hilliges, Marc Pollefeys, and Martin R Oswald. “Human from Blur: Human Pose Tracking from Blurry Images”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2023, pp. 14905–14915.

[57] Imry Kissos, Lior Fritz, Matan Goldman, Omer Meir, Eduard Oks, and Mark Kliger. “Beyond weak perspective for monocular 3d human pose estimation”. In: *Computer Vision–ECCV 2020 Workshops: Glasgow, UK, August 23–28, 2020, Proceedings, Part II 16*. Springer. 2020, pp. 541–554.

[58] Nicolo Carissimi, Paolo Rota, Cigdem Beyan, and Vittorio Murino. “Filling the gaps: Predicting missing joints of human poses using denoising autoencoders”. In: *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*. 2018, pp. 0–0.

[59] Angel Martínez-González, Michael Villamizar, Olivier Canévet, and Jean-Marc Odobez. “Investigating depth domain adaptation for efficient human pose estimation”. In: *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*. 2018, pp. 0–0.

[60] Adrián Núñez-Marcos, Gorka Azkune, and Ignacio Arganda-Carreras. “Egocentric vision-based action recognition: a survey”. In: *Neurocomputing* 472 (2022), pp. 175–197.

[61] Zhe Cao, Gines Hidalgo, Tomas Simon, Shih-En Wei, and Yaser Sheikh. “OpenPose: Real-time Multi-Person 2D Pose Estimation Using Part Affinity Fields”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 43.1 (2021), pp. 172–186.

[62] Dushyant Mehta, Srinath Sridhar, Oleksandr Sotnychenko, Helge Rhodin, Mohammad Shafiei, Hans-Peter Seidel, Weipeng Xu, Dan Casas, and Christian Theobalt. “VNect: Real-Time 3D Human Pose Estimation with a Single RGB Camera”. In: *ACM Trans. Graph.* 36.4 (July 2017).

[63] Long Zhao, Xi Peng, Yu Tian, Mubbasir Kapadia, and Dimitris N Metaxas. “Semantic Graph Convolutional Networks for 3D Human Pose Regression”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 3425–3435.

[64] Hongsuk Choi, Gyeongsik Moon, and Kyoung Mu Lee. “Pose2mesh: Graph convolutional network for 3d human pose and mesh recovery from a 2d human pose”. In: *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part VII 16*. Springer. 2020, pp. 769–787.

[65] Yanrui Bin, Zhao-Min Chen, Xiu-Shen Wei, Xinya Chen, Changxin Gao, and Nong Sang. “Structure-aware human pose estimation with graph convolutional networks”. In: *Pattern Recognition* 106 (2020), p. 107410.

[66] Hai Ci, Chunyu Wang, Xiaoxuan Ma, and Yizhou Wang. “Optimizing network structure for 3d human pose estimation”. In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2019, pp. 2262–2271.

[67] Ce Zheng, Sijie Zhu, Matias Mendieta, Taojiannan Yang, Chen Chen, and Zhengming Ding. “3d human pose estimation with spatial and temporal transformers”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 11656–11665.

[68] N Dinesh Reddy, Laurent Guigues, Leonid Pishchulin, Jayan Eledath, and Srinivasa 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*. 2021, pp. 15190–15200.

[69] Dushyant Mehta, Oleksandr Sotnychenko, Franziska Mueller, Weipeng Xu, Mohamed Elgharib, Pascal Fua, Hans-Peter Seidel, Helge Rhodin, Gerard Pons-Moll, and Christian Theobalt. “XNect: Real-time multi-person 3D motion capture with a single RGB camera”. In: *Acm Transactions On Graphics (TOG)* 39.4 (2020), pp. 82–1.

[70] Yu Cheng, Bo Wang, Bo Yang, and Robby T Tan. “Monocular 3D multi-person pose estimation by integrating top-down and bottom-up networks”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 7649–7659.

[71] Denis Tome, Patrick Peluse, Lourdes Agapito, and Hernan Badino. “xR-EgoPose: Egocentric 3D Human Pose from an HMD Camera”. In: Oct. 2019.

[72] Jian Wang, Lingjie Liu, Weipeng Xu, Kripasindhu Sarkar, Diogo Luvizon, and Christian Theobalt. “Estimating Egocentric 3D Human Pose in the Wild with External Weak Supervision”. In: *CoRR* abs/2201.07929 (2022).

[73] Hiroyasu Akada, Jian Wang, Soshi Shimada, Masaki Takahashi, Christian Theobalt, and Vladislav Golyanik. “UnrealEgo: A New Dataset for Robust Egocentric 3D Human Motion Capture”. In: (2022).

[74] Abdolrahim Kadkhodamohammadi and Nicolas Padoy. “A generalizable approach for multi-view 3d human pose regression”. In: *Machine Vision and Applications* 32.1 (2020), pp. 1–14.

[75] Hanbyul Joo, Tomas Simon, Xulong Li, Hao Liu, Lei Tan, Lin Gui, Sean Banerjee, Timothy Godisart, Bart Nabbe, Iain Matthews, et al. “Panoptic studio: A massively multiview system for social interaction capture”. In: *IEEE transactions on pattern analysis and machine intelligence* 41.1 (2017), pp. 190–204.

[76] Karim Iskakov, Egor Burkov, Victor Lempitsky, and Yury Malkov. “Learnable triangulation of human pose”. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2019, pp. 7718–7727.

[77] Z. Cao, G. Hidalgo Martinez, T. Simon, S. Wei, and Y. A. Sheikh. “OpenPose: Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2019).

[78] Xingyi Zhou, Xiao Sun, Wei Zhang, Shuang Liang, and Yichen Wei. “Deep kinematic pose regression”. In: *European Conference on Computer Vision*. Springer. 2016, pp. 186–201.

[79] Jia-Ching Cheng and Jose M. F. Moura. “Capture and representation of human walking in live video sequences”. In: *IEEE Transactions on Multimedia* 1.2 (1999), pp. 144–156.

[80] Yucheng Chen, Yingli Tian, and Mingyi He. “Monocular human pose estimation: A survey of deep learning-based methods”. In: *Computer Vision and Image Understanding* 192 (2020), p. 102897.

[81] Dragomir Anguelov, Praveen Srinivasan, Daphne Koller, Sebastian Thrun, Jim Rodgers, and James Davis. “SCAPE: shape completion and animation of people”. In: *ACM SIGGRAPH 2005 Papers*. 2005, pp. 408–416.

[82] Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J Black. “SMPL: A skinned multi-person linear model”. In: *ACM transactions on graphics (TOG)* 34.6 (2015), p. 248.

[83] Ahmed A A Osman, Timo Bolkart, and Michael J. Black. “STAR: A Spare Trained Articulated Human Body Regressor”. In: *European Conference on Computer Vision (ECCV)*. 2020.

[84] Dragomir Anguelov, Praveen Srinivasan, Daphne Koller, Sebastian Thrun, Jim Rodgers, and James Davis. “SCAPE: Shape Completion and Animation of People”. In: *ACM SIGGRAPH 2005 Papers*. SIGGRAPH ’05. Los Angeles, California: Association for Computing Machinery, 2005, pp. 408–416. ISBN: 9781450378253.

[85] Ahmed AA Osman, Timo Bolkart, and Michael J Black. “Star: Sparse trained articulated human body regressor”. In: *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part VI 16*. Springer. 2020, pp. 598–613.

[86] Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J. Black. “SMPL: a skinned multi-person linear model”. In: *ACM Trans. Graph.* 34 (2015), 248:1–248:16.

[87] Xingyi Zhou, Qixing Huang, Xiao Sun, Xiangyang Xue, and Yichen Wei. “Towards 3d human pose estimation in the wild: a weakly-supervised approach”. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, pp. 398–407.

[88] Xiao Sun, Bin Xiao, Fangyin Wei, Shuang Liang, and Yichen Wei. “Integral human pose regression”. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 529–545.

[89] Dario Pavllo, David Grangier, and Michael Auli. “Quaternet: A quaternion-based recurrent model for human motion”. In: *arXiv preprint arXiv:1805.06485* (2018).

[90] Angjoo Kanazawa, Michael J Black, David W Jacobs, and Jitendra Malik. “End-to-end recovery of human shape and pose”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 7122–7131.

[91] Isinsu Katircioglu, Bugra Tekin, Mathieu Salzmann, Vincent Lepetit, and Pascal Fua. “Learning latent representations of 3d human pose with deep neural networks”. In: *International Journal of Computer Vision* 126.12 (2018), pp. 1326–1341.

[92] Yarin Gal and Zoubin Ghahramani. “Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning”. In: *Proceedings of The 33rd International Conference on Machine Learning* (June 2015).

[93] Yarin Gal and Zoubin Ghahramani. “Dropout as a bayesian approximation: Representing model uncertainty in deep learning”. In: *international conference on machine learning*. 2016, pp. 1050–1059.

[94] Alex Kendall and Yarin Gal. “What uncertainties do we need in bayesian deep learning for computer vision?” In: *Advances in neural information processing systems*. 2017, pp. 5574–5584.

[95] Lennart Bramlage, Michelle Karg, and Cristóbal Curio. “Plausible Uncertainties for Human Pose Regression”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2023, pp. 15133–15142.

[96] Michael Truong Le, Frederik Diehl, Thomas Brunner, and Alois Knol. “Uncertainty estimation for deep neural object detectors in safety-critical applications”. In: *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. IEEE. 2018, pp. 3873–3878.

[97] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. “Simple and scalable predictive uncertainty estimation using deep ensembles”. In: *Advances in neural information processing systems* 30 (2017), pp. 6402–6413.

[98] Gino van den Bergen. “Rotational Joint Limits in Quaternion Space”. In: *Game Engine Gems* 3 (2016).

[99] Lorna Herda, Raquel Urtasun, and Pascal Fua. “Hierarchical implicit surface joint limits for human body tracking”. In: *Computer Vision and Image Understanding* 99.2 (2005), pp. 189–209.

[100] Paolo Baerlocher and Ronan Boulic. “Parametrization and range of motion of the ball-and-socket joint”. In: *Deformable avatars*. Springer, 2001, pp. 180–190.

[101] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. “Microsoft coco: Common objects in context”. In: *European conference on computer vision*. Springer. 2014, pp. 740–755.

[102] Catalin Ionescu, Dragos Papava, Vlad Olaru, and Cristian Sminchisescu. “Human3.6M: Large Scale Datasets and Predictive Methods for 3D Human Sensing in Natural Environments”. In: *IEEE transactions on pattern analysis and machine intelligence* 36 (2013).

[103] Matthew Trumble, Andrew Gilbert, Charles Malleson, Adrian Hilton, and John Collomosse. “Total Capture: 3D Human Pose Estimation Fusing Video and Inertial Sensors”. In: 2017.

[104] Dushyant Mehta, Helge Rhodin, Dan Casas, Pascal Fua, Oleksandr Sotnychenko, Weipeng Xu, and Christian Theobalt. “Monocular 3D Human Pose Estimation In The Wild Using Improved CNN Supervision”. In: *3D Vision (3DV), 2017 Fifth International Conference on*. IEEE. 2017.

[105] Naureen Mahmood, Nima Ghorbani, Nikolaus F. Troje, Gerard Pons-Moll, and Michael J. Black. “AMASS: Archive of Motion Capture as Surface Shapes”. In: *International Conference on Computer Vision*. Oct. 2019, pp. 5442–5451.

[106] Gül Varol, Javier Romero, Xavier Martin, Naureen Mahmood, Michael J. Black, Ivan Laptev, and Cordelia Schmid. “Learning from Synthetic Humans”. In: *CoRR* abs/1701.01370 (2017).

[107] Dario Pavllo, David Grangier, and Michael Auli. “QuaterNet: A Quaternion-based Recurrent Model for Human Motion”. In: *CoRR* abs/1805.06485 (2018).

[108] Elhayek Ahmed, Kovalenko Onorina, Murthy Pramod, Jameel Nawaz Malik Muhammad, and Didier Stricker. “Fully Automatic Multi-person Human Motion Capture for VR Applications”. In: *EuroVR (EuroVR-2018)*. 2018.

[109] Ijaz Akhter and Michael J Black. “Pose-conditioned joint angle limits for 3D human pose reconstruction”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 1446–1455.

[110] Shafagh Keyvanian, Michelle J Johnson, and Nadia Figueroa. “Learning Realistic Joint Space Boundaries for Range of Motion Analysis of Healthy and Impaired Human Arms”. In: *2023 IEEE-RAS 22nd International Conference on Humanoid Robots (Humanoids)*. IEEE. 2023, pp. 1–8.

[111] Morten Engell-Nørregård and Kenny Erleben. “Estimation of Joint types and Joint Limits from Motion capture data”. In: (2009).

[112] James U Korein. *A geometric investigation of reach*. MIT press, 1986.

[113] Wei Shao and Victor Ng-Thow-Hing. “A general joint component framework for realistic articulation in human characters”. In: *Proceedings of the 2003 symposium on Interactive 3D graphics*. ACM. 2003, pp. 11–18.

[114] Lorna Herda, Raquel Urtasun, Pascal Fua, and Andrew Hanson. “Automatic determination of shoulder joint limits using quaternion field boundaries”. In: *The International Journal of Robotics Research* 22.6 (2003), pp. 419–436.

[115] Catalin Ionescu, Dragos Papava, Vlad Olaru, and Cristian Sminchisescu. “Human3. 6m: Large scale datasets and predictive methods for 3d human sensing in natural environments”. In: *IEEE transactions on pattern analysis and machine intelligence* 36.7 (2014), pp. 1325–1339.

[116] F Sebastian Grassia. “Practical parameterization of rotations using the exponential map”. In: *Journal of graphics tools* 3.3 (1998), pp. 29–48.

[117] Georgios Pavlakos, Xiaowei Zhou, Konstantinos G Derpanis, and Kostas Daniilidis. “Coarse-to-fine volumetric prediction for single-image 3D human pose”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 7025–7034.

[118] Dushyant Mehta, Srinath Sridhar, Oleksandr Sotnychenko, Helge Rhodin, Mohammad Shafiei, Hans-Peter Seidel, Weipeng Xu, Dan Casas, and Christian Theobalt. “Vnect: Real-time 3d human pose estimation with a single rgb camera”. In: *ACM Transactions on Graphics (TOG)* 36.4 (2017).

[119] Ching-Hang Chen, Ambrish Tyagi, Amit Agrawal, Dylan Drover, Rohith MV, Stefan Stojanov, and James M Rehg. “Unsupervised 3D Pose Estimation with Geometric Self-Supervision”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 5714–5724.

[120] Bruce Xiaohan Nie, Ping Wei, and Song-Chun Zhu. “Monocular 3d human pose estimation by predicting depth on joints”. In: *2017 IEEE International Conference on Computer Vision (ICCV)*. IEEE. 2017, pp. 3467–3475.

[121] Xingyi Zhou, Qixing Huang, Xiao Sun, Xiangyang Xue, and Yichen Wei. “Towards 3d human pose estimation in the wild: a weakly-supervised approach”. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, pp. 398–407.

[122] Benjamin Ummenhofer, Huizhong Zhou, Jonas Uhrig, Nikolaus Mayer, Eddy Ilg, Alexey Dosovitskiy, and Thomas Brox. “DeMoN: Depth and Motion Network for Learning Monocular Stereo”. In: *CoRR* abs/1612.02401 (2016).

[123] Thanh-Toan Do, Ming Cai, Trung Pham, and Ian D. Reid. “Deep-6DPose: Recovering 6D Object Pose from a Single RGB Image”. In: *CoRR* abs/1802.10367 (2018).

[124] Yu Xiang, Tanner Schmidt, Venkatraman Narayanan, and Dieter Fox. “PoseCNN: A Convolutional Neural Network for 6D Object Pose Estimation in Cluttered Scenes”. In: *CoRR* abs/1711.00199 (2017).

[125] Ge Gao, Mikko Lauri, Jianwei Zhang, and Simone Frintrop. “Occlusion Resistant Object Rotation Regression from Point Cloud Segments”. In: *CoRR* abs/1808.05498 (2018).

[126] Ruben Villegas, Jimei Yang, Duygu Ceylan, and Honglak Lee. “Neural Kinematic Networks for Unsupervised Motion Retargetting”. In: *CoRR* abs/1804.05653 (2018).

[127] Yi Zhou, Connelly Barnes, Jingwan Lu, Jimei Yang, and Hao Li. “On the Continuity of Rotation Representations in Neural Networks”. In: *CoRR* abs/1812.07035 (2018).

[128] Zeye Wu and Wujun Che. “3D Human Pose Lifting: From Joint Position to Joint Rotation”. In: *Chinese Conference on Image and Graphics Technologies*. Springer. 2019, pp. 228–237.

[129] Qingfu Wan, Wei Zhang, and Xiangyang Xue. “DeepSkeleton: Skeleton Map for 3D Human Pose Regression”. In: *CoRR* abs/1711.10796 (2017).

[130] Leonid Sigal, Alexandru O Balan, and Michael J Black. “Humaneva: Synchronized video and motion capture dataset and baseline algorithm for evaluation of articulated human motion”. In: *IEEE transactions on pattern analysis and machine intelligence* 36.7 (2013).

[131] Catalin Ionescu, Dragos Papava, Vlad Olaru, and Cristian Sminchisescu. “Human3. 6m: Large scale datasets and predictive methods for 3d human sensing in natural environments”. In: *IEEE transactions on pattern analysis and machine intelligence* 36.7 (2013).

[132] Matthew Trumble, Andrew Gilbert, Charles Malleson, Adrian Hilton, and John Collomosse. “Total Capture: 3D Human Pose Estimation Fusing Video and Inertial Sensors.” In: *British Machine Vision Conference (BMVC)*. Vol. 2. 2017, p. 3.

[133] Dushyant Mehta, Helge Rhodin, Dan Casas, Pascal Fua, Oleksandr Sotnychenko, Weipeng Xu, and Christian Theobalt. “Monocular 3d human pose estimation in the wild using improved cnn supervision”. In: *2017 International Conference on 3D Vision (3DV)*. IEEE. 2017, pp. 506–516.

[134] Matteo Fabbri, Fabio Lanzi, Simone Calderara, Andrea Palazzi, Roberto Vezzani, and Rita Cucchiara. “Learning to detect and track visible and occluded body joints in a virtual world”. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 430–446.

[135] Xingyi Zhou, Xiao Sun, Wei Zhang, Shuang Liang, and Yichen Wei. “Deep kinematic pose regression”. In: *European Conference on Computer Vision* (2016).

[136] Pramod Murthy, Hammad T. Butt, Sandesh Hiremath, Alireza Khoshhal, and Didier. Stricker. “Learning 3D Joint Constraints from Vision based Motion Capture Datasets”. In: *International Conference on Machine Vision Applications*. 2019.

[137] Isinsu Katircioglu, Bugra Tekin, Mathieu Salzmann, Vincent Lepetit, and Pascal Fua. “Learning Latent Representations of 3D Human Pose with Deep Neural Networks”. In: *International Journal of Computer Vision* 126.12 (Dec. 2018), pp. 1326–1341.

[138] Alejandro Newell, Kaiyu Yang, and Jia Deng. “Stacked hourglass networks for human pose estimation”. In: *European conference on computer vision*. Springer. 2016, pp. 483–499.

[139] Xiao Sun, Bin Xiao, Fangyin Wei, Shuang Liang, and Yichen Wei. “Integral human pose regression”. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 529–545.

[140] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. “Automatic differentiation in PyTorch”. In: *NIPS-W*. 2017.

[141] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. “Deep residual learning for image recognition”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[142] Mykhaylo Andriluka, Leonid Pishchulin, Peter Gehler, and Bernt Schiele. “2d human pose estimation: New benchmark and state of the art analysis”. In: *Proceedings of the IEEE Conference on computer Vision and Pattern Recognition*. 2014, pp. 3686–3693.

[143] Sam Johnson and Mark Everingham. “Learning Effective Human Pose Estimation from Inaccurate Annotation”. In: *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*. 2011.

[144] Nikos Kolotouros, Georgios Pavlakos, Michael J Black, and Kostas Daniilidis. “Learning to reconstruct 3D human pose and shape via model-fitting in the loop”. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2019, pp. 2252–2261.

[145] Keze Wang, Liang Lin, Chenhan Jiang, Chen Qian, and Pengxu Wei. “3D human pose machines with self-supervised learning”. In: *IEEE transactions on pattern analysis and machine intelligence* (2019).

[146] Julieta Martinez, Rayat Hossain, Javier Romero, and James J Little. “A simple yet effective baseline for 3d human pose estimation”. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, pp. 2640–2649.

[147] Van-Hung Le. “3-D Human Pose Estimation in Traditional Martial Art Videos”. In: *International Journal of Machine Learning and Computing* 10.2 (2020).

[148] Weichen Zhang, Zhiguang Liu, Liuyang Zhou, Howard Leung, and Antoni B Chan. “Martial arts, dancing and sports dataset: A challenging stereo and multi-view dataset for 3d human pose estimation”. In: *Image and Vision Computing* 61 (2017), pp. 22–39.

[149] Jason Y Zhang, Panna Felsen, Angjoo Kanazawa, and Jitendra Malik. “Predicting 3d human dynamics from video”. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2019, pp. 7114–7123.

[150] Aiden Nibali, Zhen He, Stuart Morgan, and Daniel Greenwood. “Extraction and classification of diving clips from continuous video footage”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*. 2017, pp. 38–48.

[151] Gagan Kanojia, Sudhakar Kumawat, and Shanmuganathan Raman. “Attentive spatio-temporal representation learning for diving classification”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 2019, pp. 0–0.

[152] Yingwei Li, Yi Li, and Nuno Vasconcelos. “Resound: Towards action recognition without representation bias”. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 513–528.

[153] Ke Zhang, Hanbo Ying, Hong-Ning Dai, Lin Li, Yuanyuan Peng, Keyi Guo, and Hongfang Yu. “Compacting Deep Neural Networks for Internet of Things: Methods and Applications”. In: *IEEE Internet of Things Journal* 8.15 (2021), pp. 11935–11959.

[154] Xin Chen, Anqi Pang, Wei Yang, Yuexin Ma, Lan Xu, and Jingyi Yu. “SportsCap: Monocular 3D human motion capture and fine-grained understanding in challenging sports videos”. In: *International Journal of Computer Vision* 129.10 (2021), pp. 2846–2864.

[155] Alejandro Newell, Kaiyu Yang, and Jia Deng. “Stacked hourglass networks for human pose estimation”. In: *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part VIII 14*. Springer. 2016, pp. 483–499.

[156] Jingdong Wang, Ke Sun, Tianheng Cheng, Borui Jiang, Chaorui Deng, Yang Zhao, Dong Liu, Yadong Mu, Mingkui Tan, Xinggang Wang, et al. “Deep high-resolution representation learning for visual recognition”. In: *IEEE transactions on pattern analysis and machine intelligence* 43.10 (2020), pp. 3349–3364.

[157] Changqian Yu, Bin Xiao, Changxin Gao, Lu Yuan, Lei Zhang, Nong Sang, and Jingdong Wang. “Lite-hrnet: A lightweight high-resolution network”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2021, pp. 10440–10450.

[158] Bowen Cheng, Bin Xiao, Jingdong Wang, Honghui Shi, Thomas S Huang, and Lei Zhang. “Higherhrnet: Scale-aware representation learning for bottom-up human pose estimation”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 5386–5395.

[159] Bruno Artacho and Andreas Savakis. “Unipose: Unified human pose estimation in single images and videos”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 7035–7044.

[160] Yufei Xu, Jing Zhang, Qiming Zhang, and Dacheng Tao. “Vitpose: Simple vision transformer baselines for human pose estimation”. In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 38571–38584.

[161] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. “An image is worth 16x16 words: Transformers for image recognition at scale”. In: *arXiv preprint arXiv:2010.11929* (2020).

[162] Weian Mao, Yongtao Ge, Chunhua Shen, Zhi Tian, Xinlong Wang, Zhibin Wang, and Anton van den Hengel. “Poseur: Direct human pose regression with transformers”. In: *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part VI*. Springer. 2022, pp. 72–88.

[163] Katja Ludwig, Sebastian Scherer, Moritz Einfalt, and Rainer Lienhart. “Self-Supervised Learning for Human Pose Estimation in Sports”. In: *2021 IEEE International Conference on Multimedia & Expo Workshops (ICMEW)*. 2021, pp. 1–6.

[164] Takumi Kitamura, Hitoshi Teshima, Diego Thomas, and Hiroshi Kawasaki. “Refining OpenPose with a new sports dataset for robust 2D pose estimation”. In: *2022 IEEE/CVF Winter Conference on Applications of Computer Vision Workshops (WACVW)*. 2022, pp. 672–681.

[165] James Hong, Matthew Fisher, Michaël Gharbi, and Kayvon Fatahalian. *Video Pose Distillation for Few-Shot, Fine-Grained Sports Action Recognition*. 2021.

[166] Nicola Giulietti, Alessia Caputo, Paolo Chiariotti, and Paolo Castellini. “SwimmerNET: Underwater 2D Swimmer Pose Estimation Exploiting Fully Convolutional Neural Networks”. In: *Sensors* 23.4 (2023), p. 2364.

[167] Jinglin Xu, Yongming Rao, Xumin Yu, Guangyi Chen, Jie Zhou, and Jiwen Lu. *FineDiving: A Fine-grained Dataset for Procedure-aware Action Quality Assessment*. 2022.

[168] Sina Honari, Victor Constantin, Helge Rhodin, Mathieu Salzmann, and Pascal Fua. “Temporal Representation Learning on Monocular Videos for 3D Human Pose Estimation”. In: *IEEE Transactions on Pattern Analysis & Machine Intelligence* (2022), pp. 1–12.

[169] Valentin Bazarevsky, Ivan Grishchenko, Karthik Raveendran, Tyler Zhu, Fan Zhang, and Matthias Grundmann. “BlazePose: On-device Real-time Body Pose tracking”. In: (2020).

[170] Sangbum Choi, Seokeon Choi, and Changick Kim. “MobileHumanPose: Toward real-time 3D human pose estimation in mobile devices”. In: 2021, pp. 2328–2338.

[171] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Zitnick. “Microsoft COCO: Common Objects in Context”. In: vol. 8693. 2014. ISBN: 978-3-319-10601-4.

[172] Mark Sandler, Andrew G. Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. “MobileNetV2: Inverted Residuals and Linear Bottlenecks”. In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2018), pp. 4510–4520.

[173] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. “U-net: Convolutional networks for biomedical image segmentation”. In: *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*. Springer. 2015, pp. 234–241.

[174] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. “Tensorflow: A system for large-scale machine learning”. In: *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*. 2016, pp. 265–283.

[175] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019, pp. 8024–8035.

[176] Max-Plank Institute. *MPII Human Pose Dataset*. http://human-pose.mpi-inf.mpg.de/. Accessed on February 24, 2025. 2023.

[177] Diederik Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: *International Conference on Learning Representations* (2014).

[178] ONNX. *Open Neural Network Exchange*. https://onnx.ai/. 2023.

[179] Gyeongsik Moon, Ju Chang, and Kyoung Mu Lee. "Camera Distance-aware Top-down Approach for 3D Multi-person Pose Estimation from a Single RGB Image". In: (July 2019).

[180] Raghuraman Krishnamoorthi. "Quantizing deep convolutional networks for efficient inference: A whitepaper". In: *arXiv preprint arXiv:1806.08342* (2018).

[181] Cheng Zhang, Fan Yang, Gang Li, Qiang Zhai, Yi Jiang, and Dong Xuan. "MV-Sports: a motion and vision sensor integration-based sports analysis system". In: *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*. IEEE. 2018, pp. 1070–1078.

[182] Manju Rana and Vikas Mittal. "Wearable sensors for real-time kinematics analysis in sports: A review". In: *IEEE Sensors Journal* 21.2 (2020), pp. 1187–1207.

[183] Long Sha, Patrick Lucey, Yisong Yue, Xinyu Wei, Jennifer Hobbs, Charlie Rohlf, and Sridha Sridharan. "Interactive sports analytics: An intelligent interface for utilizing trajectories for interactive sports play retrieval and analytics". In: *ACM Transactions on Computer-Human Interaction (TOCHI)* 25.2 (2018), pp. 1–32.

[184] Graham Thomas, Rikke Gade, Thomas B Moeslund, Peter Carr, and Adrian Hilton. "Computer vision for sports: Current applications and research topics". In: *Computer Vision and Image Understanding* 159 (2017), pp. 3–18.

[185] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. "Going deeper with convolutions". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 1–9.

[186] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. "Learning spatiotemporal features with 3d convolutional networks". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 4489–4497.

[187] Fabio Carrara, Petr Elias, Jan Sedmidubsky, and Pavel Zezula. "LSTM-based real-time action detection and prediction in human motion streams". In: *Multimedia Tools and Applications* 78.19 (2019), pp. 27309–27331.

[188] Tianwei Lin, Xu Zhao, and Zhaoxuan Fan. "Temporal action localization with two-stream segment-based RNN". In: *2017 IEEE International Conference on Image Processing (ICIP)*. IEEE. 2017, pp. 3400–3404.

[189] JAIN Mihir, LI Zhenyang, Efstratios Gavves, and Cornelis Gerardus Maria Snoek. *Action localization in sequential data with attention proposals from a recurrent network*. US Patent App. 15/250,755. Sept. 2017.

[190] Serena Yeung, Olga Russakovsky, Ning Jin, Mykhaylo Andriluka, Greg Mori, and Li Fei-Fei. "Every moment counts: Dense detailed labeling of actions in complex videos". In: *International Journal of Computer Vision* 126.2 (2018), pp. 375–389.

[191] Roeland De Geest and Tinne Tuytelaars. “Modeling temporal structure with lstm for online action detection”. In: *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE. 2018, pp. 1549–1557.

[192] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. “On the difficulty of training recurrent neural networks”. In: *International conference on machine learning*. PMLR. 2013, pp. 1310–1318.

[193] Khurram Soomro, Haroon Idrees, and Mubarak Shah. “Action localization in videos through context walk”. In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 3280–3288.

[194] Sijie Yan, Yuanjun Xiong, and Dahua Lin. “Spatial temporal graph convolutional networks for skeleton-based action recognition”. In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 32. 1. 2018.

[195] Maheen Rashid, Hedvig Kjellstrom, and Yong Jae Lee. “Action graphs: Weakly-supervised action localization with graph convolution networks”. In: *Proceedings of the IEEE Winter Conference on Applications of Computer Vision*. 2020, pp. 615–624.

[196] Pallabi Ghosh, Yi Yao, Larry Davis, and Ajay Divakaran. “Stacked spatio-temporal graph convolutional networks for action segmentation”. In: *Proceedings of the IEEE Winter Conference on Applications of Computer Vision*. 2020, pp. 576–585.

[197] Mihir Jain, Jan Van Gemert, Hervé Jégou, Patrick Bouthemy, and Cees GM Snoek. “Action localization with tubelets from motion”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2014, pp. 740–747.

[198] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. “Rich feature hierarchies for accurate object detection and semantic segmentation”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2014, pp. 580–587.

[199] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. “Faster R-CNN: towards real-time object detection with region proposal networks”. In: *IEEE transactions on pattern analysis and machine intelligence* 39.6 (2016), pp. 1137–1149.

[200] Yu-Wei Chao, Sudheendra Vijayanarasimhan, Bryan Seybold, David A Ross, Jia Deng, and Rahul Sukthankar. “Rethinking the Faster R-CNN architecture for temporal Action Localization”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 1130–1139.

[201] Rui Hou, Chen Chen, and Mubarak Shah. “Tube convolutional neural network (t-cnn) for action detection in videos”. In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 5822–5831.

[202] Colin Lea, Michael D Flynn, Rene Vidal, Austin Reiter, and Gregory D Hager. “Temporal convolutional networks for action segmentation and detection”. In: *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 156–165.

[203] Wlodzislaw Duch, Janusz Kacprzyk, Erkki Oja, and Slawomir Zadrozny. *Artificial Neural Networks: Formal Models and Their Applications–ICANN 2005: 15th International Conference, Warsaw, Poland, September 11-15, 2005, Proceedings*. 2005.

[204] Rui Dai, Srijan Das, Luca Minciullo, Lorenzo Garattoni, Gianpiero Francesca, and Francois Bremond. “PDAN: Pyramid Dilated Attention Network for Action Detection”. In: *Proceedings of the IEEE Winter Conference on Applications of Computer Vision*. 2021, pp. 2970–2979.

[205] Yazan Abu Farha and Jurgen Gall. “Ms-tcn: Multi-stage temporal convolutional network for action segmentation”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 3575–3584.

[206] Kanav Vats, Mehrnaz Fani, Pascale Walters, David A Clausi, and John Zelek. “Event detection in coarsely annotated sports videos via parallel multi-receptive field 1d convolutions”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 2020, pp. 882–883.

[207] Shaojie Bai, J Zico Kolter, and Vladlen Koltun. “An empirical evaluation of generic convolutional and recurrent networks for sequence modeling”. In: *CoRR, abs/1803.01271* (2018).

[208] Muhammed Kocabas, Nikos Athanasiou, and Michael J. Black. “VIBE: Video Inference for Human Body Pose and Shape Estimation”. In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2020.

[209] Paritosh Parmar and Brendan Tran Morris. “What and How Well You Performed? A Multitask Learning Approach to Action Quality Assessment”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2019.

[210] Paritosh Parmar and Brendan Tran Morris. “Learning to Score Olympic Events”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*. July 2017.

[211] Paritosh Parmar and Brendan Morris. “Action quality assessment across multiple actions”. In: *2019 IEEE winter conference on applications of computer vision (WACV)*. IEEE. 2019, pp. 1468–1476.

[212] Qing Lei, Hongbo Zhang, and Jixiang Du. “Temporal attention learning for action quality assessment in sports video”. In: *Signal, Image and Video Processing* (2021), pp. 1–9.

[213] Emily E Cust, Alice J Sweeting, Kevin Ball, and Sam Robertson. “Machine and deep learning for sport-specific movement recognition: A systematic review of model development and performance”. In: *Journal of sports sciences* 37.5 (2019), pp. 568–600.

[214] Ashley Hall, Brandon Victor, Zhen He, Matthias Langer, Marc Elipot, Aiden Nibali, and Stuart Morgan. “The detection, tracking, and temporal action localisation of swimmers for automated analysis”. In: *Neural Computing and Applications* (2020), pp. 1–19.

[215] Vladimir M Zatsiorsky and Vladimir M Zaciorskij. *Kinetics of human motion*. 2002.

[216] VM Zatsiorsky. “Methods of determining mass-inertial characteristics of human body segments”. In: *Contemporasy Problems of Biomechnics* (1990).

[217] Paolo De Leva. “Adjustments to Zatsiorsky-Seluyanov’s segment inertia parameters”. In: *Journal of biomechanics* 29.9 (1996), pp. 1223–1230.

[218] Bertold Fricke and Thomas Köthe. *Wasserspringen: Einblicke in die Sporttechnik und ihre Vermittlung*. Vol. 12. Meyer & Meyer Verlag, 2009.

[219] Rudolph Emil Kalman. “A new approach to linear filtering and prediction problems”. In: (1960).

[220] Herbert E Rauch, F Tung, and Charlotte T Striebel. “Maximum likelihood estimates of linear dynamic systems”. In: *AIAA journal* 3.8 (1965), pp. 1445–1450.

[221] *SAE Levels of Driving Automation Refined for Clarity and International Audience*. https://www.sae.org/blog/sae-j3016-update. Accessed on February 24, 2025. 2023.

[222] SAE International. *SAE J3016 Update*. https://www.sae.org/standards/content/j3016_202104/. Accessed on February 24, 2025. 2023.

[223] Synopsys. *Autonomous Driving Levels Explained*. https://www.synopsys.com/automotive/autonomous-driving-levels.html. Accessed on February 24, 2025. 2023.

[224] Rambus. *Driving Automation Levels*. https://www.rambus.com/blogs/driving-automation-levels. Accessed on February 24, 2025. 2023.

[225] Gear Patrol. *A Guide to SAE Self-Driving Levels*. https://www.gearpatrol.com/cars/a42642120/sae-self-driving-levels. Accessed on February 24, 2025. 2023.

[226] Tesla. *Autopilot and Full Self-Driving Capability*. https://www.tesla.com/support/autopilot. Accessed: Dec. 10, 2023. 2023.

[227] Christian Häne, Lionel Heng, Gim Hee Lee, Friedrich Fraundorfer, Paul Furgale, Torsten Sattler, and Marc Pollefeys. "3D visual perception for self-driving cars using a multi-camera system: Calibration, mapping, localization, and obstacle detection". In: *Image and Vision Computing* 68 (2017), pp. 14–27.

[228] Fu Xiong, Boshen Zhang, Yang Xiao, Zhiguo Cao, Taidong Yu, Joey Tianyi Zhou, and Junsong Yuan. "A2J: Anchor-to-Joint Regression Network for 3D Articulated Pose Estimation from a Single Depth Image". In: *CoRR* abs/1908.09999 (2019).

[229] Gyeongsik Moon, Ju Yong Chang, and Kyoung Mu Lee. "V2v-posenet: Voxel-to-voxel prediction network for accurate 3d hand and human pose estimation from a single depth map". In: *Proceedings of the IEEE conference on computer vision and pattern Recognition*. 2018, pp. 5079–5088.

[230] Keze Wang, Shengfu Zhai, Hui Cheng, Xiaodan Liang, and Liang Lin. "Human Pose Estimation from Depth Images via Inference Embedded Multi-Task Learning". In: *Proceedings of the 24th ACM International Conference on Multimedia*. MM '16. Amsterdam, The Netherlands: Association for Computing Machinery, 2016, pp. 1227–1236. ISBN: 9781450336031.

[231] Jigyasa Singh Katrolia, Ahmed ElSherif, Hartmut Feld, Bruno Mirbach, Jason R. Rambach, and Didier Stricker. "TICaM: A Time-of-flight In-car Cabin Monitoring Dataset". In: *32nd British Machine Vision Conference 2021, BMVC 2021, Online, November 22-25, 2021*. BMVA Press, 2021, p. 277.

[232] Steve Dias, Oliver Wasenmuller, HansPeter Beise, Thomas Stifter, and Didier Stricker. "SVIRO: Synthetic Vehicle Interior Rear Seat Occupancy Dataset and Benchmark". In: *IEEE Winter Conference on Applications of Computer Vision (WACV)*. 2020.

[233] Paul A Green, Heejin Jeong, and Te-Ping Kang. "Using an OpenDS driving simulator for car following: A first attempt". In: *Adjunct Proceedings of the 6th International Conference on Automotive User Interfaces and Interactive Vehicular Applications*. 2014, pp. 1–6.

[234] *Dep Meshworks*. https://depusa.com/. Accessed on December 10, 2023. 2023.

[235] MakeHuman Community. *MakeHuman*. http://www.makehumancommunity.org/. Accessed on December 10, 2023.

[236] Hum3D. *Hum3D*. https://hum3d.com/. Accessed on December 10, 2023.

[237] Nima Ghorbani and Michael J. Black. "SOMA: Solving Optical Marker-Based MoCap Automatically". In: *Proc. International Conference on Computer Vision (ICCV)*. Oct. 2021, pp. 11117–11126.

[238] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. "Learning representations by back-propagating errors". In: *nature* 323.6088 (1986), pp. 533–536.

[239] Dor Bank, Noam Koenigstein, and Raja Giryes. *Autoencoders*. 2021.

[240] Pierre Baldi. "Autoencoders, unsupervised learning, and deep architectures". In: *Proceedings of ICML workshop on unsupervised and transfer learning*. JMLR Workshop and Conference Proceedings. 2012, pp. 37–49.

[241] Viviana Crescitelli, Atsutake Kosuge, and Takashi Oshima. "POISON: Human Pose Estimation in Insufficient Lighting Conditions Using Sensor Fusion". In: *IEEE Transactions on Instrumentation and Measurement* 70 (2021), pp. 1–8.

[242] Lorenzo Porzi, Adrian Penate-Sanchez, Elisa Ricci, and Francesc Moreno-Noguer. “Depth-aware convolutional neural networks for accurate 3D pose estimation in RGB-D images”. In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2017, pp. 5777–5783.

[243] Andrea D’Eusanio, Stefano Pini, Guido Borghi, Roberto Vezzani, and Rita Cucchiara. “RefiNet: 3D Human Pose Refinement with Depth Maps”. In: *2020 25th International Conference on Pattern Recognition (ICPR)*. 2021, pp. 2320–2327.

[244] Viviana Crescitelli, Atsutake Kosuge, and Takashi Oshima. “POISON: Human Pose Estimation in Insufficient Lighting Conditions Using Sensor Fusion”. In: vol. 70. 2021, pp. 1–8.

[245] Tao Yu, Jianhui Zhao, Zerong Zheng, Kaiwen Guo, Qionghai Dai, Hao Li, Gerard Pons-Moll, and Yebin Liu. “DoubleFusion: Real-Time Capture of Human Performances with Inner Body Shapes from a Single Depth Sensor”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 42.10 (2020), pp. 2523–2539.

[246] Lasse Hansen, Marlin Siebert, Jasper Diesel, and Mattias P Heinrich. “Fusing information from multiple 2D depth cameras for 3D human pose estimation in the operating room”. In: *International journal of computer assisted radiology and surgery* 14.11 (2019), pp. 1871–1879.

[247] Pascal Schneider, Yuriy Anisimov, Raisul Islam, Bruno Mirbach, Jason Rambach, Didier Stricker, and Frédéric Grandidier. “Timo—a dataset for indoor building monitoring with a time-of-flight camera”. In: *Sensors* 22.11 (2022), p. 3992.

[248] Matteo Fabbri, Fabio Lanzi, Simone Calderara, Stefano Alletto, and Rita Cucchiara. “Compressed Volumetric Heatmaps for Multi-Person 3D Pose Estimation.” In: *Computer Vision and Pattern Recognition*. 2020.

[249] István Sárándi, Timm Linder, Kai Oliver Arras, and Bastian Leibe. “Metrabs: metric-scale truncation-robust heatmaps for absolute 3d human pose estimation”. In: *IEEE Transactions on Biometrics, Behavior, and Identity Science* 3.1 (2020), pp. 16–30.

[250] Wenjie Luo, Bin Yang, and Raquel Urtasun. “Fast and furious: Real time end-to-end 3d detection, tracking and motion forecasting with a single convolutional net.” In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 3569–3577.

[251] Aaron S Jackson, Chris Manafas, and Georgios Tzimiropoulos. “3d human body reconstruction from a single image via volumetric regression”. In: *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*. 2018, pp. 0–0.

[252] Gul Varol, Duygu Ceylan, Bryan Russell, Jimei Yang, Ersin Yumer, Ivan Laptev, and Cordelia Schmid. “Bodynet: Volumetric inference of 3D human body shapes”. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 20–36.

[253] Wei Tang, Pei Yu, and Ying Wu. “Deeply learned compositional models for human pose estimation.” In: *In European Conference on Computer Vision (ECCV)*. 2018, pp. 3569–3577.

[254] Haiping Wu, Bin Xiao, Noel Codella, Mengchen Liu, Xiyang Dai, Lu Yuan, and Lei Zhang. “Cvt: Introducing convolutions to vision transformers”. In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2021, pp. 22–31.

[255] Angela Dai and Matthias Nießner. “3dmv: Joint 3d-multiview prediction for 3d semantic scene segmentation.” In: *European Conference on Computer Vision*. 2018.

[256] Charles Ruizhongtai Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. “PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation”. In: *CoRR* abs/1612.00593 (2016).

[257] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J. Guibas. “PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space”. In: *CoRR* abs/1706.02413 (2017).

[258] Wenxuan Wu, Zhongang Qi, and Li Fuxin. “Pointconv: Deep convolutional networks on 3d point clouds”. In: *Proceedings of the IEEE/CVF Conference on computer vision and pattern recognition*. 2019, pp. 9621–9630.

[259] Yangyan Li, Rui Bu, Mingchao Sun, Wei Wu, Xinhan Di, and Baoquan Chen. “Pointcnn: Convolution on x-transformed points”. In: *Advances in neural information processing systems* 31 (2018).

[260] Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. “nuscenes: A multimodal dataset for autonomous driving”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 11621–11631.

[261] Jens Behley, Martin Garbade, Andres Milioto, Jan Quenzel, Sven Behnke, Cyrill Stachniss, and Jurgen Gall. “Semantickitti: A dataset for semantic scene understanding of lidar sequences”. In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2019, pp. 9297–9307.

[262] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. “Vision meets robotics: The kitti dataset”. In: *The International Journal of Robotics Research* 32.11 (2013), pp. 1231–1237.

[263] Shuguang Li, Bei Liu, Yang Zhao, Ke Zheng, and Hong Cheng. “An object detection method enhanced by sparse point cloud for low illumination in autonomous driving”. In: *2022 IEEE 25th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE. 2022, pp. 293–297.

[264] Masakazu Ohno, Riki Ukyo, Tatsuya Amano, Hamada Rizk, and Hirozumi Yamaguchi. “Privacy-preserving pedestrian tracking with path image inpainting and 3D point cloud features”. In: *Pervasive and Mobile Computing* 100 (2024), p. 101914.

[265] Avinash Prabu, Renran Tian, Lingxi Li, Jialiang Le, Srinivasan Sundararajan, and Saeed Barbat. “Peek into the Future Camera-based Occupant Sensing in Configurable Cabins for Autonomous Vehicles”. In: *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*. IEEE. 2021, pp. 3321–3327.

[266] Michael Fürst, Shriya TP Gupta, René Schuster, Oliver Wasenmüller, and Didier Stricker. “HPERL: 3d human pose estimation from RGB and lidar”. In: *2020 25th International Conference on Pattern Recognition (ICPR)*. IEEE. 2021, pp. 7321–7327.

[267] Zhijie Yao, Yazhou Liu, Zexuan Ji, Quansen Sun, Pongsak Lasang, and Shengmei Shen. “3D Driver Pose Estimation Based on Joint 2d-3d Network”. In: *2019 IEEE International Conference on Image Processing (ICIP)*. 2019, pp. 2546–2550.

[268] Qingyong Hu, Bo Yang, Linhai Xie, Stefano Rosa, Yulan Guo, Zhihua Wang, Niki Trigoni, and Andrew Markham. “RandLA-Net: Efficient Semantic Segmentation of Large-Scale Point Clouds”. In: *CoRR* abs/1911.11236 (2019).

[269] Fabian Groh, Patrick Wieschollek, and Hendrik Lensch. “Flex-convolution (million-scale point-cloud learning beyond grid-worlds)”. In: *arXiv preprint arXiv:1803.07289* (2018).

[270] Oren Dovrat, Itai Lang, and Shai Avidan. “Learning to Sample”. In: *CoRR* abs/1812.01659 (2018).

[271] Abubakar Abid, Muhammad Fatih Balin, and James Zou. “Concrete autoencoders for differentiable feature selection and reconstruction”. In: *arXiv preprint arXiv:1901.09346* (2019).

[272] Jiancheng Yang, Qiang Zhang, Bingbing Ni, Linguo Li, Jinxian Liu, Mengdie Zhou, and Qi Tian. “Modeling point clouds with self-attention and gumbel subset sampling”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 3323–3332.

[273] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. “Show, attend and tell: Neural image caption generation with visual attention”. In: *International conference on machine learning*. PMLR. 2015, pp. 2048–2057.

[274] Yuval Eldar, Michael Lindenbaum, Moshe Porat, and Yehoshua Y Zeevi. “The farthest point strategy for progressive image sampling”. In: *IEEE transactions on image processing* 6.9 (1997), pp. 1305–1315.

[275] Albert Haque, Boya Peng, Zelun Luo, Alexandre Alahi, Serena Yeung, and Li Fei-Fei. “Viewpoint Invariant 3D Human Pose Estimation with Recurrent Error Feedback”. In: *CoRR* abs/1603.07076 (2016).

[276] Microsoft. *Azure Kinect body tracking SDK*. https://learn.microsoft.com/en-us/azure/kinect-dk. Accessed on February 24, 2025. 2023.

[277] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. “Rethinking Atrous Convolution for Semantic Image Segmentation”. In: *CoRR* abs/1706.05587 (2017).

[278] Fisher Yu and Vladlen Koltun. *Multi-Scale Context Aggregation by Dilated Convolutions*. 2015.

[279] Adam Paszke, Abhishek Chaurasia, Sangpil Kim, and Eugenio Culurciello. “ENet: A Deep Neural Network Architecture for Real-Time Semantic Segmentation”. In: *CoRR* abs/1606.02147 (2016).

[280] Bruno Artacho and Andreas E. Savakis. “Waterfall Atrous Spatial Pooling Architecture for Efficient Semantic Segmentation”. In: *CoRR* abs/1912.03183 (2019).

[281] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L. Yuille. “DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs”. In: *CoRR* abs/1606.00915 (2016).

[282] Gen Li and Yuantao Gu. *Approximation of Gram-Schmidt Orthogonalization by Data Matrix*. 2017.

[283] Young-Woon Cha, True Price, Zhen Wei, Xinran Lu, Nicholas Rewkowski, Rohan Chabra, Zihe Qin, Hyounghun Kim, Zhaoqi Su, Yebin Liu, Adrian Ilie, Andrei State, Zhenlin Xu, Jan-Michael Frahm, and Henry Fuchs. “Towards Fully Mobile 3D Face, Body, and Environment Capture Using Only Head-worn Cameras”. In: *IEEE Transactions on Visualization and Computer Graphics* 24.11 (2018), pp. 2993–3004.

[284] Weipeng Xu, Avishek Chatterjee, Michael Zollhöfer, Helge Rhodin, Pascal Fua, Hans-Peter Seidel, and Christian Theobalt. “Mo2Cap2: Real-time Mobile 3D Motion Capture with a Cap-mounted Fisheye Camera”. In: *CoRR* abs/1803.05959 (2018).

[285] Yahui Zhang, Shaodi You, and Theo Gevers. “Automatic Calibration of the Fisheye Camera for Egocentric 3D Human Pose Estimation from a Single Image”. In: *2021 IEEE Winter Conference on Applications of Computer Vision (WACV)*. 2021, pp. 1771–1780.

[286] Dong-Hyun Hwang, Kohei Aso, Ye Yuan, Kris Kitani, and Hideki Koike. “MonoEye: Multimodal Human Motion Capture System Using A Single Ultra-Wide Fisheye Camera”. In: Oct. 2020.

[287] Alireza Fathi, Ali Farhadi, and James M. Rehg. “Understanding egocentric activities”. In: *2011 International Conference on Computer Vision*. 2011, pp. 407–414.

[288] Minghuang Ma, Haoqi Fan, and Kris M. Kitani. “Going Deeper into First-Person Activity Recognition”. In: *CoRR* abs/1605.03688 (2016).

[289] Congqi Cao, Yifan Zhang, Yi Wu, Hanqing Lu, and Jian Cheng. “Egocentric Gesture Recognition Using Recurrent 3D Convolutional Neural Networks with Spatiotemporal Transformer Modules”. In: *2017 IEEE International Conference on Computer Vision (ICCV)*. 2017, pp. 3783–3791.

[290] Grégory Rogez, James Steven Supancic III, and Deva Ramanan. “Egocentric Pose Recognition in Four Lines of Code”. In: *CoRR* abs/1412.0060 (2014).

[291] Haruka Yonemoto, Kazuhiko Murasaki, Tatsuya Osawa, Kyoko Sudo, Jun Shimamura, and Yukinobu Taniguchi. “Egocentric articulated pose tracking for action recognition”. In: *2015 14th IAPR International Conference on Machine Vision Applications (MVA)* (2015), pp. 98–101.

[292] Hao Jiang and Kristen Grauman. “Seeing Invisible Poses: Estimating 3D Body Pose from Egocentric Video”. In: *CoRR* abs/1603.07763 (2016).

[293] Helge Rhodin, Christian Richardt, Dan Casas, Eldar Insafutdinov, Mohammad Shafiei, Hans-Peter Seidel, Bernt Schiele, and Christian Theobalt. “EgoCap: Egocentric Marker-less Motion Capture with Two Fisheye Cameras”. In: *CoRR* abs/1609.07306 (2016).

[294] Saeid Vosoughi and Maria A. Amer. “Deep 3D Human Pose Estimation Under Partial Body Presence”. In: *2018 25th IEEE International Conference on Image Processing (ICIP)*. 2018, pp. 569–573.

[295] Bruno Artacho and Andreas Savakis. “Waterfall Atrous Spatial Pooling Architecture for Efficient Semantic Segmentation”. In: *Sensors* 19.24 (2019). ISSN: 1424-8220.

[296] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L. Yuille. “DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs”. In: *CoRR* abs/1606.00915 (2016).

[297] Xavier Glorot and Yoshua Bengio. “Understanding the difficulty of training deep feedforward neural networks”. In: *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*. Ed. by Yee Whye Teh and Mike Titterington. Vol. 9. Proceedings of Machine Learning Research. Chia Laguna Resort, Sardinia, Italy: PMLR, 13–15 May 2010, pp. 249–256.

[298] Weipeng Xu, Avishek Chatterjee, Michael Zollhoefer, Helge Rhodin, Pascal Fua, Hans-Peter Seidel, and Christian Theobalt. “Mo^{2}Cap^2 : Real-time Mobile 3D Motion Capture with a Cap-mounted Fisheye Camera”. In: *IEEE Transactions on Visualization and Computer Graphics* (2019), pp. 1–1. ISSN: 1077-2626.

[299] Jian Wang, Lingjie Liu, Weipeng Xu, Kripasindhu Sarkar, Diogo Luvizon, and Christian Theobalt. *Estimating Egocentric 3D Human Pose in the Wild with External Weak Supervision*. Jan. 2022.

[300] Dan Levi, Liran Gispan, Niv Giladi, and Ethan Fetaya. “Evaluating and calibrating uncertainty prediction in regression tasks”. In: *arXiv preprint arXiv:1905.11659* (2019).

[301] Dimity Miller, Niko Sünderhauf, Haoyang Zhang, David Hall, and Feras Dayoub. “Benchmarking Sampling-based Probabilistic Object Detectors.” In: *CVPR Workshops*. Vol. 3. 2019.

[302] Lorenzo Bertoni, Sven Kreiss, and Alexandre Alahi. “Monoloco: Monocular 3d pedestrian localization and uncertainty estimation”. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2019, pp. 6861–6871.

[303] Apratim Bhattacharyya, Mario Fritz, and Bernt Schiele. “Long-term on-board prediction of people in traffic scenes under uncertainty”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 4194–4202.

[304] Lukas Neumann, Andrew Zisserman, and Andrea Vedaldi. “Relaxed softmax: Efficient confidence auto-calibration for safe pedestrian detection”. In: (2018).

[305] Abhinav Kumar, Tim K Marks, Wenxuan Mou, Ye Wang, Michael Jones, Anoop Cherian, Toshiaki Koike-Akino, Xiaoming Liu, and Chen Feng. “LUVLi Face Alignment: Estimating Landmarks’ Location, Uncertainty, and Visibility Likelihood”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 8236–8246.

[306] Nitesh B Gundavarapu, Divyansh Srivastava, Rahul Mitra, Abhishek Sharma, and Arjun Jain. “Structured Aleatoric Uncertainty in Human Pose Estimation.” In: *CVPR Workshops*. Vol. 2. 2019.

[307] Davis Rempe, Tolga Birdal, Aaron Hertzmann, Jimei Yang, Srinath Sridhar, and Leonidas J. Guibas. “HuMoR: 3D Human Motion Model for Robust Pose Estimation”. In: *CoRR* abs/2105.04668 (2021).

[308] Pramod Murthy, Hammad T Butt, Sandesh Hiremath, Alireza Khoshhal, and Didier Stricker. “Learning 3D joint constraints from vision-based motion capture datasets”. In: *IPSJ Transactions on Computer Vision and Applications* 11.1 (2019), pp. 1–9.

[309] Pramod Murthy, Bertram Taetz, Arpit Lekhra, and Didier Stricker. “DIVENET: Dive Action Localization and Physical Parameter Extraction for High Performance Training”. In: *IEEE Access*. 2023.

[310] Pramod Murthy, Bertram Taetz, and Didier Stricker. “Low-latency detection of diving sports movements in training videos”. In: *20th Spring School "Technologies in Competitive Sports", Leipzig, Germany*. IAT. 2022.

[311] Pramod Murthy, Onorina Kovalenko, Ahmed Elhayek, C Couto Gava, and Didier Stricker. “3D human pose tracking inside car using single rgb spherical camera”. In: *ACM Chapters Computer Science in Cars Symposium (CSCS)*. 2017.

[312] Denis Tome, Thiemo Alldieck, Patrick Peluse, Gerard Pons-Moll, Lourdes Agapito, Hernan Badino, and Fernando De la Torre. “Selfpose: 3d egocentric pose estimation from a headset mounted camera”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2020).

[313] Ilia Petrov, Vlad Shakhuro, and Anton Konushin. “Deep probabilistic human pose estimation”. In: *IET Computer Vision* 12.5 (2018), pp. 578–585.

[314] Omar Del-Tejo-Catalá, Jose-Luis Guardiola, Javier Pérez, David Millán Escrivá, Alberto J Perez, and Juan-Carlos Perez-Cortes. “Probabilistic Pose Estimation from Multiple Hypotheses”. In: *IEEE Access* (2023).

[315] Vandad Davoodnia, Saeed Ghorbani, Marc-André Carbonneau, Alexandre Messier, and Ali Etemad. “UPose3D: Uncertainty-Aware 3D Human Pose Estimation with Cross-View and Temporal Cues”. In: *arXiv preprint arXiv:2404.14634* (2024).

[316] Sai Kumar Dwivedi, Cordelia Schmid, Hongwei Yi, Michael J. Black, and Dimitrios Tzionas. “POCO: 3D Pose and Shape Estimation using Confidence”. In: *International Conference on 3D Vision (3DV)*. Mar. 2024.

[317] Nicolas Duczek, Matthias Kerzel, and Stefan Wermter. “Continual learning from synthetic data for a humanoid exercise robot”. In: *arXiv preprint arXiv:2102.10034* (2021).

[318] Hiromu Taketsugu and Norimichi Ukita. “Active Transfer Learning for Efficient Video-Specific Human Pose Estimation”. In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. 2024, pp. 1880–1890.

[319] Steve Dias Da Cruz, Bertram Taetz, Thomas Stifter, and Didier Stricker. “Autoencoder attractors for uncertainty estimation”. In: *2022 26th International Conference on Pattern Recognition (ICPR)*. IEEE. 2022, pp. 2553–2560.

[320] Thomas Lucas, Fabien Baradel, Philippe Weinzaepfel, and Grégory Rogez. “Posegpt: Quantization-based 3d human motion generation and forecasting”. In: *European Conference on Computer Vision*. Springer. 2022, pp. 417–435.

[321] Yao Feng, Jing Lin, Sai Kumar Dwivedi, Yu Sun, Priyanka Patel, and Michael J. Black. “ChatPose: Chatting about 3D Human Pose”. In: *CVPR*. 2024.

[322] Alejandro Newell, Kaiyu Yang, and Jia Deng. “Stacked Hourglass Networks for Human Pose Estimation”. In: *CoRR* abs/1603.06937 (2016).

[323] Zerui Chen, Yan Huang, Hongyuan Yu, Bin Xue, Ke Han, Yiru Guo, and Liang Wang. “Towards Part-Aware Monocular 3D Human Pose Estimation: An Architecture Search Approach”. In: *ECCV*. 2020.

[324] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. “Deep Residual Learning for Image Recognition”. In: *CoRR* abs/1512.03385 (2015).

[325] Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, and Quoc V. Le. “MnasNet: Platform-Aware Neural Architecture Search for Mobile”. In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2019), pp. 2815–2823.

[326] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zach DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. “Automatic differentiation in PyTorch”. In: 2017.

[327] Muhammed Kocabas, Nikos Athanasiou, and Michael J Black. “VIBE: Video inference for human body pose and shape estimation”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 5253–5263.

[328] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. “Learning phrase representations using RNN encoder-decoder for statistical machine translation”. In: *arXiv preprint arXiv:1406.1078* (2014).

[329] Naureen Mahmood, Nima Ghorbani, Nikolaus F Troje, Gerard Pons-Moll, and Michael J Black. “AMASS: Archive of motion capture as surface shapes”. In: *arXiv preprint arXiv:1904.03278* (2019).

[330] Christos Baziotis, Nikos Athanasiou, Alexandra Chronopoulou, Athanasia Kolovou, Georgios Paraskevopoulos, Nikolaos Ellinas, Shrikanth Narayanan, and Alexandros Potamianos. “Ntua-slp at semeval-2018 task 1: Predicting affective content in tweets with deep attentive rnns and transfer learning”. In: *arXiv preprint arXiv:1804.06658* (2018).

[331] Richard Sinkhorn. “A relationship between arbitrary positive matrices and doubly stochastic matrices”. In: *The annals of mathematical statistics* 35.2 (1964), pp. 876–879.

[332] Paul-Edouard Sarlin, Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. “SuperGlue: Learning Feature Matching With Graph Neural Networks”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2020.

[333] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. “Attention Is All You Need”. In: *CoRR* abs/1706.03762 (2017).

[334] Sam Johnson and Mark Everingham. “Clustered Pose and Nonlinear Appearance Models for Human Pose Estimation.” In: *bmvc*. Vol. 2. 4. 2010, p. 5.

[335] Weiyu Zhang, Menglong Zhu, and Konstantinos G Derpanis. “From actemes to action: A strongly-supervised representation for detailed action understanding”. In: *Proceedings of the IEEE international conference on computer vision*. 2013, pp. 2248–2255.

[336] Benjamin Sapp and Ben Taskar. “MODEC: Multimodal Decomposable Models for Human Pose Estimation”. In: *In Proc. CVPR*. 2013.

[337] Zhe Cao, Gines Hidalgo, Tomas Simon, Shih-En Wei, and Yaser Sheikh. “OpenPose: realtime multi-person 2D pose estimation using Part Affinity Fields”. In: *arXiv preprint arXiv:1812.08008*. 2018.

[338] Timo von Marcard, Roberto Henschel, Michael Black, Bodo Rosenhahn, and Gerard Pons-Moll. “Recovering Accurate 3D Human Pose in The Wild Using IMUs and a Moving Camera”. In: *European Conference on Computer Vision (ECCV)*. 2018.

[339] Albert Haque, Boya Peng, Zelun Luo, Alexandre Alahi, Serena Yeung, and Li Fei-Fei. “Towards Viewpoint Invariant 3D Human Pose Estimation”. In: *European Conference on Computer Vision*. Oct. 2016.

[340] Keze Wang, Shengfu Zhai, Hui Cheng, Xiaodan Liang, and Liang Lin. “Human pose estimation from depth images via inference embedded multi-task learning”. In: *Proceedings of the 24th ACM international conference on Multimedia*. 2016, pp. 1227–1236.

[341] Amir Shahroudy, Jun Liu, Tian-Tsong Ng, and Gang Wang. “Ntu rgb+ d: A large scale dataset for 3d human activity analysis”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 1010–1019.

[342] Stefano Pini, Andrea D’Eusanio, Guido Borghi, Roberto Vezzani, and Rita Cucchiara. “Baracca: a multimodal dataset for anthropometric measurements in automotive”. In: *2020 IEEE International Joint Conference on Biometrics (IJCB)*. IEEE. 2020, pp. 1–7.

[343] Manuel Martin, Alina Roitberg, Monica Haurilet, Matthias Horne, Simon Reiß, Michael Voit, and Rainer Stiefelhagen. “DriveAct: A Multi-modal Dataset for Fine-grained Driver Behavior Recognition in Autonomous Vehicles”. In: (Oct. 2019).

[344] Matteo Fabbri, Fabio Lanzi, Simone Calderara, Andrea Palazzi, Roberto Vezzani, and Rita Cucchiara. “Learning to detect and track visible and occluded body joints in a virtual world”. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 430–446.

[345] Gul Varol, Javier Romero, Xavier Martin, Naureen Mahmood, Michael J Black, Ivan Laptev, and Cordelia Schmid. “Learning from synthetic humans”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 109–117.

[346] Hiroyasu Akada, Jian Wang, Soshi Shimada, Masaki Takahashi, Christian Theobalt, and Vladislav Golyanik. “UnrealEgo: A New Dataset for Robust Egocentric 3D Human Motion Capture”. In: *European Conference on Computer Vision (ECCV)*. 2022.

[347] Helge Rhodin, Christian Richardt, Dan Casas, Eldar Insafutdinov, Mohammad Shafiei, Hans-Peter Seidel, Bernt Schiele, and Christian Theobalt. “Egocap: egocentric marker-less motion capture with two fisheye cameras”. In: *ACM Transactions on Graphics (TOG)* 35.6 (2016), pp. 1–11.

[348] Dongxu Zhao, Zhen Wei, Jisan Mahmud, and Jan-Michael Frahm. “Egoglass: Egocentric-view human pose estimation from an eyeglass frame”. In: *2021 International Conference on 3D Vision (3DV)*. IEEE. 2021, pp. 32–41.

[349] Jian Wang, Lingjie Liu, Weipeng Xu, Kripasindhu Sarkar, Diogo Luvizon, and Christian Theobalt. “Estimating Egocentric 3D Human Pose in the Wild with External Weak Supervision”. In: *CVPR* (2022).

[350] Yuxuan Liu, Jianxin Yang, Xiao Gu, Yijun Chen, Yao Guo, and Guang-Zhong Yang. “EgoFish3D: Egocentric 3D Pose Estimation From a Fisheye Camera via Self-Supervised Learning”. In: *IEEE Transactions on Multimedia* 25 (2023), pp. 8880–8891.

[351] Hildegard Kuehne, Hueihan Jhuang, Estíbaliz Garrote, Tomaso Poggio, and Thomas Serre. “HMDB: a large video database for human motion recognition”. In: *2011 International conference on computer vision*. IEEE. 2011, pp. 2556–2563.

[352] Will Kay, Joao Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, et al. “The kinetics human action video dataset”. In: *arXiv preprint arXiv:1705.06950* (2017).

[353] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. “UCF101: A dataset of 101 human actions classes from videos in the wild”. In: *arXiv preprint arXiv:1212.0402* (2012).

[354] Yi Zhou, Connelly Barnes, Jingwan Lu, Jimei Yang, and Hao Li. “On the Continuity of Rotation Representations in Neural Networks”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. CVPR. 2019, pp. 5745–5753.

List of Figures

1.1 Ill-posed problem of 3D Human Pose Estimation. The figure shows multiple 3D pose hypothesis to have same 2D pose reprojection [51] 4

1.2 Various challenges for human pose estimation in monocular images and videos. (Images: [52]) 5

1.3 The scope of thesis is to focus on single-person pose using monocular view images (left). The thesis parts (right) showing the different application scenarios and the partial observation type in each scenario. The horizontal bars show the possible range of camera distance in meters for each part. 6

2.1 The figure illustrates the top-down approach for pose estimation. The Region of Interest(ROI) is detected using a person detector. The cropped person images is processed to detect joint keypoints. The keypoints are combined to output the skeletal 2D pose. (Image source: [52]) 16

2.2 The figure illustrates bottom-up approach to pose estimation. The input image is processed to detect all the joint keypoints in an image. The keypoints are later processed to form a person's skeleton. (Image source: [52]) 16

2.3 The figure shows four different 2D pose representations. The 2D pose representations can be represented in keypoint coordinates, heatmap regression, part affinity fields and skeleton graphs (from left to right). 19

2.4 Kinematic models in a tree structure of human body representation to measure joint rotation in 2D (left) and 3D rotation(right) with relative rotations [79]. . . . 20

2.5 An example of 3D kinematic chain representing left arm. 20

2.6 Commonly used human body models:(a) skeleton-based (b) contour-based (c) volume-based. [80] 22

2.7 Illustration of generated body shapes using SMPL body models [82]. 23

2.8 The different phases describing the development of SMPL model [82]. 24

3.1 The joint angles from motion capture datasets are transformed into (s_1, s_2, τ) swing-twist representation. Then look-up maps are prepared to train a discriminator on joint constraints to classify valid and invalid poses. 32

3.2 Swing-Twist joint angle rotation decomposition 34

3.3 The knee joint rotation anomalies in Akhter et al. dataset[2] 36

3.4 The right hip joint's range of swing rotation 37

3.5 Generated synthetic dataset for right hip joint with valid (green) and invalid (red) samples . . . 38
3.6 The valid left shoulder joint rotation of (i) Trained discriminators output (green) (ii) Akhter et al. dataset (blue) [109] and (iii) Synthetic dataset (magenta) . . . 39
3.7 Valid discriminator network output of left shoulder joint in (s_1, s_2, τ) space . . . 40
3.8 Interactive validation tool showing invalid left arm poses in red color and valid left arm poses in green color . . . 40
3.9 Interactive Dataset Analyzer tool showing 3D pose and all the look up maps learnt for each joint for the Human3.6M dataset [102] . . . 41
3.10 The Figure (a) shows the limitation of Quaternion (vector part) space of Right Hip (Right-Up-Leg) used by Herda et al. [114] shows duality of valid joint rotation. Anti-podal points +q & -q represent same rotation. Figure (b) illustrates artifacts in Euler Angle space of Right Hip (Right-Up-Leg) shows non-contiguous clusters of Valid Joint Rotation. Also boundary jumps at 180 degrees (for Z and X) and 90 degrees for Y rotation are visible . . . 42

4.1 Illustration of Deep 3D human pose regression using swing-twist decomposition for a right shoulder joint. . . . 45
4.2 HeatmapReg: Illustration of stacked hourglass network modules regress 16 swing heatmaps and twist joint orientations to construct the complete 3D human pose. . . . 47
4.3 VectorReg: Illustration of stacked hourglass network modules estimate 16 swing-twist joint orientations to construct the complete 3D human pose. . . . 49
4.4 The learned swing-twist space joint constraints [136] representing valid swing occupancy map (bottom) and twist range map [τ_{min},τ_{max}] (top) for the left shoulder joint orientation. The red points represent the upper bound (τ_{max}), and blue points represent the lower bound (τ_{min}) of the twist angle for a given valid swing (S_1, S_2) value. . . . 52
4.5 Vector regression estimates (depicted in red) for swing-twist angles (S_1, S_2, τ) of the left shoulder joint during a 2-minute "Directions" activity of Subject S9 from the Human3.6M Dataset [131]. The ground truth values are represented in blue. . . . 56
4.6 The top row displays RGB frames of a motion sequence, and the bottom row shows the corresponding 3D human poses. Ground truth 3D poses are shown in blue, while predicted poses are depicted in red. The first two columns on the left demonstrate successful pose predictions, while the last two columns on the right exhibit predictions over challenging scenarios with occlusions. . . . 56
4.7 The learned swing-twist space joint constraints [136] representing valid swing occupancy map (bottom) and twist range map [τ_{min},τ_{max}] (top) for the left shoulder joint orientation. The red points represent the upper bound (τ_{max}), and blue points represent the lower bound (τ_{min}) of the twist angle for a given valid swing (S_1, S_2) value. . . . 57

4.8 The baseline network structure similar to [146] . . . 58

5.1 Illustration of the problem statement: Extracting Diving Pose Metrics using Action Localization and Diving Pose extraction from RGB images capturing Diving sports motion. . . . 68
5.2 Exemplar images from the DSV diving dataset depicting a person jumping over the springboard. . . . 70
5.3 The plot shows the range of the diving time for different diving heights in the DSV datasets. . . . 70
5.4 Representative images from the IAT dataset showcasing frames from Olympic diving performance broadcasting videos . . . 71
5.5 The architecture of DiveNet pose regression model. The input image(HxW) is fed through the ResNet backbone and Waterfall Atrous Spatial Pyramid(WASP) module to obtain 256 feature channels at reduced resolution by a factor of 8. The decoder module generates 17 heatmaps, one per 16 joints and one for COM, followed by a local max operation which produces 16 keypoints and 1 COM (as shown in output image with a red circle) . . . 71
5.6 The PCK-h plot compares the accuracy of different models at different PCK-h threshold. . . . 73
5.7 The figures demonstrate the 2D posture estimate results of various models on the DSV Diving set. Divenet shows superior pose regression results on challenging diving images. . . . 74
5.8 The figures demonstrate the predictions of various models on exceedingly challenging scenarios in the DSV and IAT Diving sets. The last column shows the current limitations of DiveNet's predictions on challenging scenarios. . . . 75
5.9 Our representation of BlazePose model architecture [169] . . . 77
5.10 MobileHumanPose model results for the pre-trained and fine-tuned models. . . 80
5.11 Effect of quantization and compression on the 2D human pose output. . . . 81
5.12 Effect of quantization and compression on the 3D human pose estimation output. . . . 81

6.1 Illustration of diving action localization in an untrimmed video. . . . 84
6.2 Action Localization using class labels (NoDive, Start, Dive, End) for video frames. The dotted vertical blue line and green line represent the start frame and end frame of the diving action respectively. . . . 87
6.3 Images from the recorded DSV diving dataset. Figure (a), (b), (c) and (d) shows examplar frames of a diving video representing the classes NoDive, Start, Dive, End, respectively. . . . 87
6.4 Diving action localization - TCN block architecture . . . 88
6.5 The ground truth and predicted probability signals for the 4 class model (No Dive, Start, Dive, End). Bold lines represent the ground truth probability signal, and colored areas represent the predicted probabilities. . . . 90

6.6 Histogram of the latency measure of the estimated start and end boundary markers. The y-axis represents the number of sequences and x-axis represents the lag in frames. The negative values on x-axis represents prediction prior to the reference frame and positive values represents prediction after the reference frame. 90
6.7 The confusion matrix shows the model accuracy over test set. The model uses a HMR backbone network with 17 frames as input. 91
6.8 Model accuracy over various individual classes using different class weighting configurations. 92
6.9 Ablation study to analyze the required context in (a) and effectiveness of TCN with respect to GRU networks (b). 93

7.1 Illustration of physical parameter extraction framework from diving action video. 98
7.2 Illustration of the COM computation. First, we compute the midpoints of bone segments representing bone centroids (a). Next, we compute the COM by a weighted sum of centroids (b). The COM trajectory in the image space (c). . 99
7.3 Diving motion peak height estimation 99
7.4 Diving motion COM trajectory estimation using homography. 104
7.5 Measurement of body axis in the images. Figure (a) shows the body axis (green arrow) computed using least-square method over estimated all the 2D pose joint keypoints. The body axis using only torso joints joints(magenta). Figure(b) shows ellipsoid approximation of the body represented in orange color. The body axis is measured using the major axis(red) in an ellipse . . . 108
7.6 The diving trajectory of estimated center of mass in each dive and body axis(using ellipsoid method) while ending the dive. 109
7.7 Body Angle Filtering. 110
7.8 The plots show the COM error distribution of Unipose-FT (top) and DiveNet (bottom) model. The median COM error (left-orange line) of Unipose-FT model is 36.08 pixels and 95 percentile error threshold (left-red line) lies at 60.76 pixels. The median COM error of DiveNet (right-orange line) is significantly lower, at 5.87 pixels and the 95 percentile threshold (right-red line) lies at 16.64 pixels. 111
7.9 The plots show 2D joint angle error distribution of Hip Angles and Knee Angles for Unipose-FT (left) and DiveNet (right). The median angle error (left-orange line) of Unipose-FT model is 8.28 degrees and 95 percentile error threshold (left-red line) lies at 35.435 degrees. The median angle error of DiveNet (right-orange line) is marginally higher at 8.52 degrees and the 95 percentile threshold (right-red line) lies at 38.96 degrees 112
7.10 Homography based projections on Orthogonal Physical view. 112
7.11 Illustration showing demonstrator tool to extract different pose parameters in recorded training session diving videos. 113
7.12 BlazePose model integrated on android phone. 114

7.13 MobileHumanPose model inference results on DSV (top row) and IAT dataset(bottom row). 115

8.1 Own representation of driving automation levels defined by ASE [221] 122
8.2 Example images of TICaM dataset consists of the real depth images, real RGB image, real infrared image, synthetic depth image, synthetic RGB image, synthetic infrared image [231]. 125
8.3 Example images of SVIRO dataset consists of the synthetic depth image, synthetic RGB image, synthetic infrared image [232]. 126
8.4 The figure shows the sample images in DREICAP-3D dataset. There are four images consisting of IR and depth with fish eye view (raw) and perspective view (undistorted) images. 127
8.5 Different activities in the DREICAP-3D dataset 128
8.6 Illustration of motive application showing a snapshot of attached markers for a sample motion sequence recording of DREICAP-3D dataset. 129
8.7 The plots show reconstruction of motion capture data tracked over motion sequence in the dataset. Plot (a) show the raw jitter data captured from markers. Plot (b) shows the smoothed raw data after jitter clean up. Plot(c) shows the final reconstructed missing data for short gaps using Motive application. . . . 130
8.8 Illustration of the process of assigning labels(color) to mocap point clouds(grey) trajectory [237] by removing ghost points(red) 131
8.9 The process of generating smplx annotations for raw mocap point cloud using the SOMA [237] method 132
8.10 The figure (a),(b) are front view and back view of markers layout in Optitrack recording dataset respectively. The figure (c) and (d) corresponds to the front view and back view synthetic training 133
8.11 The DREICAP-3D dataset sample images(top row) and their corresponding smplx pose annotations(bottom row). 134

9.1 The fusion model which takes infrared image and depth image to estimate the 3D driver pose. 137
9.2 The figures shows our own representation of the network architecture [250]. Volumetric Heatmap Autoencoder (VHA) aims to learn the compressed representation of input volumetric heatmaps that preserve the information content (top). The Code Predictor Network (CPN) aims to predict the codes obtained by the VHA(bottom) 139
9.3 Our representation of the architecture presented in W. Luo, et al [250]. The Volumetric Heatmap Autoencoder (VHA) aims to learn the compressed representation of input volumetric heatmaps that preserve the information content in the latent code $e(H)$. 140
9.4 Code Predictor Network taking an IR image as input and predicting the codes.140

9.5 Code Predictor Network - Fusion which takes features from IR and Depth images to predicts the codes. 141
9.6 Convolutional Vision Transformer Encoder Module [254]. 142
9.7 Pose Refiner Network structure comprising three fully connected layers, with *N* representing the number of joints. Figure adapted from M. Fabbri et al. [248]. 142
9.8 The figure shows representation of a 3D joint volumetric heatmap, which shows the joints 2D heatmap uncertainties along the depth channels. 144
9.9 Post processing step. 144
9.10 The figure represents the output of our Code Predictor Model 9.2.2 on some of the example images of TiCaM dataset. 145
9.11 Illustration of 3D pose outputs obtained through CPN-Fusion 9.2.2 applied to selected images from the SVIRO dataset. 147
9.12 The figure represents the 2D/3D output from various Code Predictor Networks(CPN) on test sample taken from the DREICAP-3D dataset. The first column represents the 2D output. The second and third column represents the 3D output using Mlab and Pyplot respectively. 149
9.13 Figure shows the comparison between the various Code Predictor Network on raw images with and without using Pose Refiner Network. F1 indicates that a predicted joint is considered “true positive” if the distance from the corresponding ground truth joint is less than a given threshold. 151
9.14 Qualitative results of Pose Refiner model on the DREICAP-3D test dataset. . 152

10.1 The overview of proposed method 155
10.2 The figure shows the PointConvNet’s network structure with local feature aggregation (LFA) module 157
10.3 Line chart of each joint’s mAP for random sampling and farthest point sampling on the ITOP dataset 160
10.4 All images are point clouds sampled with 2048 points (a) is the point cloud sampled by random sampling, (b) is the point cloud sampled by FPS 161
10.5 Bar chart shows mAP for each joint using random sampling(RS) and with local feature aggregation (RS-LFA) on the DREICAP-3D test dataset; the accuracy plot of person04(top) and person06(bottom) participant test sequences shows accuracy over different joints. 162
10.6 The mAP of each joint for groundtruth generated by SOMA and Kinect 3D pose ground truth data in DREICAP-3D testset dataset(person04) 163
10.7 Comparison plots of the groundtruth generated using two different methods: SOMA(red) and Kinect(green) 163
10.8 The plot shows the 3D PCK evaluation results at different thresholds for SOMA method and Kinect method generated 3D pose ground truth data 164

11.1 The general hardware configuration for EgoCentric Pose Estimation is shown in figure(a) [71]. The figure (b) illustrates how the EgoCentric image has a distorted view of the fisheye camera [71]. The figure (c) displays the subject's 3D joint position. 172
11.2 The XR-Ego Network Architecture as presented by [71]. The lower branch is dedicated to reconstructing 2D heatmaps, while the upper branch is responsible for regressing 3D joint positions. 174
11.3 The figure shows a triple branch feature-based network, EgoPoseNet. In addition to ResNet as the backbone, we integrating the WASP module [295], we process the input RGB image with dimensions $H \times W$ to generate 256 feature channels at a reduced resolution by a factor of eight. The first branch of EgoPoseNet produces $H \times W \times K$ 2D heatmaps, the second branch predicts $K \times 3$ 3D joint positions, and the third branch regresses a 6D representation of joint rotations with dimensions $K \times 6$. 176
11.4 The Figure shows 2D heatmap uncertainty in model predictions. Fig(a) and Fig (b) show the comparsion of 2D heatmap regressed by xr-Ego and EgoPoseNet model respectively for partly visible wrist joints. 180
11.5 The figures illustrate qualitative results showcasing 2D and 3D pose outputs on the XR-EgoPose dataset 185
11.6 Figure shows uncertainty in model predictions using the Monte Carlo Dropout method. The x-axis in all the figures shows the frame number. The y-axis shows the mean and variance of 3D joint predictions in red and ground truth in green in mm 186
11.7 Figure shows model predictions angle error. The x-axis in all the figure shows the frame number. The y-axis depicts angle error in red. 187
11.8 The Figure shows 2D heatmap uncertainty in model predictions. Fig(a) and Fig (b) show the frames with 2D heatmap overlayed. The Frame 67 shows large error on right shoulder due partial visibility of left arm. Similary, the model prediction has lowest error on Frame 79 due to full visible of right arm including the wrist joints. 187

A.1 Stacked Hourglass network for pose estimation consisting of multiple stacked hourglass modules allowing for repeated bottom-up, top-down inference [322].199
A.2 A single "hourglass" module is illustrated. As shown in this Figure, each box in the diagram corresponds to a residual module. Throughout the entire hourglass, there are the same amount of features [322]. 200
A.3 MobileHumanPose model architecture [170] 201
A.4 Comparison with various backbone networks. * denotes the corresponding network pre-trained on ImageNet. Metric: MPJPE (mm) [170] 201
A.5 Architecture of SPIN [144]. 202
A.6 Architecture of VIBE. [327] 203

A.7 The entire pipeline of SOMA model [237]. From generating synthetic training data to labeling the point cloud and computing the corresponding SMPL-X model . . . 204
A.8 Architecture of SOMA model(left), self-attention block(middle) and Conv1D-block(right) [237] . . . 205
A.9 Illustration of the Optimal Transport layer processing [332] . . . 206
A.10 Architecture of the scaled dot product attention(left) and the multi-head attention(right) [333] . . . 206
A.11 The hierarchical feature learning architecture of PointNet++ [257] . . . 207
A.12 The Waterfall Atrous Spatial Pooling Architecture as presented by [295]. . . . 210

C.1 Definition of Continuous Representation and its relation with neural network. [354] . . . 216
C.2 Visual representation of 2D rotation from original space in SO(2) to angular representation space. [354] . . . 217

List of Tables

3.1 Classification comparison results between learned discriminator (ours) and Akhter et al. [109] on generated left shoulder synthetic dataset 41
3.2 The accuracy of the discriminator networks for different joints on synthetic test dataset generated using look-up maps of Human3.6M dataset [102] 42

4.1 Activity-wise mean Euler angle error values in radians for models trained on different joint angle parameterizations. 52
4.2 Choice of 3D joint rotation representation: Mean 3D joint positions errors (MPJPE in mm) of each network trained separately on Euler Angles, Quaternions and Swing-Twist. Evaluation over Protocol #1 on Human 3.6M Dataset [131] across all activities. 54
4.3 The comparison of mean Swing-Twist angle error for HeatmapReg and VectorReg architectures with constraints(HeatmapReg-wc,VectorReg-wc) and without constraints in degrees. We measure the error over all activities preformed by test set Subjects [S9,S11] from Human 3.6M Dataset [131] 54
4.4 Joint-wise evaluation of 3D joint positions errors (MPJPE in mm) - Protocol #1, Protocol #2 and Protocol #3 54
4.5 Activity-wise mean Euler angle error values in radians for models trained on different joint angle parameterizations. 55
4.6 Comparison of mean Swing-Twist angle error for HeatmapReg and VectorReg architectures with constraints (HeatmapReg-C, VectorReg-C) and without constraints in radians. Errors are measured across all activities performed by test set Subjects [S9,S11] from Human 3.6M Dataset [131]. 57
4.7 Evaluation of baseline lifting of 2D GT annotations to 3D pose. All architectures are trained for 100 epochs. MPJPE and PAMPJPE are in mm. Angle error is expressed in degree. 58
4.8 The activity-wise comparision of 3D mean per joint positions errors with procrustes alignment (PA-MPJPE in mm) for all the baseline methods trained on Human3.6M [131] (S1, S5, S6 and S8 and tested for S9 and S11) Training is done for all the models using same hyperparameters and 100 epochs. . . . 59

5.1 The PCK@0.25 accuracy metric showing the accuracy of pose regression models on DSV and IAT test dataset. 72
5.2 PCKH metric applied to the DSV diving dataset. 79

5.3 PCK-h metric applied to the IAT diving dataset. 79
5.4 3D evaluation of MobileHumanPose [170] and Moon et al. model [179] on MuPots-3D dataset. 80
5.5 The joints accuracy at PCK-h@0.5 for different compression models. 82

6.1 Quantitative results: Ablation study results of different type of feature extractor with different input length. For instance, HMR_5 is Human Mesh Recovery model with five frames as input. The metric is per frame classification accuracy, the higher the better. The top two best results are highlighted in bold. 93
6.2 Accuracy(Acc) and Latency (Lat) scores for models with different weighted configurations used in weighted cross entropy loss during the training. The latency values represent the percentage of frames that have latency between $[-1, 1]$ in the respective columns. 94

7.1 The table shows the COM physical parameter mean pixel errors over different height jumps performed in the test set of diving datasets 110
7.2 The Table shows a comparison of the mean joint angle errors in degrees over different height jumps in the diving test set. The lower the angle value, the better is the model performance. We observed less than 1 degree difference and no significant variation in Hip angle and Knee angle error between the two models. 111
7.3 The Table shows the sensitivity in peak height estimation of the diving trajectory measured in meters over different height jumps performed in the test set of DSV diving datasets. 111

8.1 The table describes the detailed comparison of each in-car cabin datasets. . 128
8.2 The specific activities of each session in DREICAP-3D dataset 129
8.3 Evaluation results of the trained model on DREICAP-3D dataset 133

9.1 This table shows the MPJPE error in meters for each joint individually on the validation set of the DREICAP-3D dataset in various scenarios of our Code Predictor Network. 148
9.2 The table compares various VHA models trained with different voxelization factor on DREICAP-3D test set by F1 score. F1 score considers a predicted joint to be true positive if the distance from the corresponding ground truth joint is less than 1 voxel space. 150

10.1 The Table provides a comprehensive overview of the model's performance across different sampling methods and ground truth generation techniques. The rows show mAP of each person in the test-set for random sampling (RS), RS-LFA(SOMA), and RS-LFA(Kinect) on the DREICAP-3D dataset. The last row presents the mean mAP values. 161

10.2 The Table shows the mAP of each person in the test set for 3D KeyPoints using RS(SOMA) and 3D Joint Rotations(SOMA) over DREICAP-3D dataset 164

11.1 Performance of EgoPoseNet network on xR-EgoPose test dataset [71] and Mo2Cap2 test dataset [298]. Our method outperforms the state-of-the-art methods on xR-EgoPose dataset and provide competitive results on Mo2Cap2 test set. 181
11.2 The Table presents EgoPoseNet 2D pose evaluation results on xR-EgoPose test dataset [71] using Mean Pixel Error (MPE) metric. The Table presents a comparison of results for XR-EgoNet (our implementation), EgoPoseNet single branch, EgoPoseNet dual branch, and EgoPoseNet triple branch. . . 181
11.3 The Table shows EgoPoseNet performance on xR-EgoPose test dataset [71] for 3D pose using MPJPE error. The Table also compares the result for XR-EgoNet(our) implementation, EgoPoseNet dual branch, and EgoPoseNet triple branch. 182
11.4 Performance of our egocentric pose estimation network on xR-EgoPose test dataset [71] for 2D pose. The angle error measured in degrees for various skeleton joints are shown for two different models. The Table compares the result for EgoPoseNet(row 1) and EgoPoseNet-AL with additional angle losses(row2). 182

A.1 Heatmap Encoder network architecture of EgoposeNet. The encoder receives $15 \times 48 \times 48$ 2D heatmaps from ResNet-101. It uses three convolution layers, each followed by batch normalization, leaky ReLU with a negative slope of 0.2, and a dropout layer with a 0.5 probability. The output after convolution 1 is $64 \times 24 \times 24$, convolution 2 is $128 \times 12 \times 12$, and convolution 3 is $512 \times 6 \times 6$. Afterward, a flattened layer is applied followed by a fully connected layer to generate a latent z of size 20. 208
A.2 Heatmap Decoder branch network architecture of EgoposeNet. The decoder gets $256 \times 23 \times 23$ feature map from the WASP module and $256 \times 92 \times 92$ low-level features from layer 1 of ResNet-101. MaxPool1 and Bilinear Interpolation is performed on low-level features followed by a concatenation of low-level features and WASP features and three convolution layers. The final output is $16 \times 46 \times 46$ 2D heatmaps. 208
A.3 Pose Decoder branch network architecture of EgoposeNet. The decoder gets $256 \times 23 \times 23$ feature map from the WASP module and $256 \times 92 \times 92$ low-level features from layer 1 of ResNet-101. MaxPool1 and Bilinear Interpolation is performed on low-level features followed by a concatenation of low-level features and WASP features and three convolution layers. Further MaxPool2 layer is applied followed by flattened layer to generate 11638 output. A fully connected layer downsamples 11638 to the final output of 48 which signifies 3D joint positions of 16 joints. 209

B.1 The table provides overview of different human pose and activity datasets used in the literature. 211

Curriculum Vitae

Personal Information



Name	Pramod Narasimha Murthy

Education

2016-2024	**Dr.-Ing. (Doctor of Engineering)**, Computer Science (*Augmented Vision*), Rheinland-Pfälzische Technische Universität Kaiserslautern-Landau (RPTU), Germany
2011-2016	**Master of Science**, Computer Science (*Intelligent Systems*), Technische Universität Kaiserslautern, Germany
2005-2006	**Post Graduate Diploma** in *Wireless and Mobile Computing*, Centre for Development of Advanced Computing (C-DAC), Pune, India
2001-2005	**Bachelor of Engineering** (*Computer Science & Engineering*), Swami Ramanand Teerth Marathwada University, Nanded, India

Experience

2016–2024	**Researcher**, Augmented Vision Department, German Research Center for Artificial Intelligence (DFKI), Kaiserslautern, Germany
2013–2015	**Research Assistant**, Chair at Real Time Systems, Technische Universität Kaiserslautern, Germany
2013–2015	**Research Assistant**, Flow and Material Simulation Department, Fraunhofer Institute of Industrial Mathematics(ITWM), Kaiserslautern, Germany
2010–2011	**R&D Engineer**, Multimodal Inputs & Gestures, Nokia R&D, Bangalore, India
2008–2010	**Senior Software Engineer,** Eterno Infotech (now Ver Se Innovation) Pvt. Ltd., Bangalore, India
2006–2007	**Member of Technical Staff,** Persistent Systems Ltd, Pune, India

Languages

English, Deutsch (B1), Tamil, Kannada, Marathi, Hindi

List of Publications

[1] Pramod Murthy, Bertram Taetz, and Didier Stricker. “Low-latency detection of diving sports movements in training videos”. In: *20th Spring School "Technologies in Competitive Sports", Leipzig, Germany*. IAT. 2022.

[2] Pramod Murthy, Bertram Taetz, Arpit Lekhra, and Didier Stricker. “DIVENET: Dive Action Localization and Physical Parameter Extraction for High Performance Training”. In: *IEEE Access*. 2023.

[3] Pramod Murthy, Onorina Kovalenko, Ahmed Elhayek, C Couto Gava, and Didier Stricker. “3D human pose tracking inside car using single rgb spherical camera”. In: *ACM Chapters Computer Science in Cars Symposium (CSCS)*. 2017.

[4] Pramod Murthy, Hammad T Butt, Sandesh Hiremath, Alireza Khoshhal, and Didier Stricker. “Learning 3D joint constraints from vision-based motion capture datasets”. In: *IPSJ Transactions on Computer Vision and Applications* 11.1 (2019), pp. 1–9.

[5] Hammad Tanveer Butt, Manthan Pancholi, Mathias Musahl, **Murthy, Pramod**, Maria Alejandra Sanchez, and Didier Stricker. “Inertial motion capture using adaptive sensor fusion and joint angle drift correction”. In: *2019 22th International Conference on Information Fusion (FUSION)*. IEEE, 2019, pp. 1–8.

[6] Hammad Tanveer Butt, Bertram Taetz, Mathias Musahl, Maria A Sanchez, **Murthy, Pramod**, and Didier Stricker. “Magnetometer robust deep human pose regression with uncertainty prediction using sparse body worn magnetic inertial measurement units”. In: *IEEE Access* 9 (2021), pp. 36657–36673.

[7] Hammad Tanveer Butt, Mathias Musahl, Maria A Sanchez, **Murthy, Pramod**, Kumar Ankit, and Didier Stricker. “Simultaneous End User Calibration of Multiple Magnetic Inertial Measurement Units With Associated Uncertainty”. In: *IEEE Access* 9 (2021), pp. 26468–26483.

[8] Sheela Raju Kurupathi, **Murthy, Pramod**, and Didier Stricker. “Generation of Human Images with Clothing using Advanced Conditional Generative Adversarial Networks.” In: *DeLTA* 3 (2020).

[9] Ahmed Elhayek, Onorina Kovalenko, **Murthy, Pramod**, Jameel Malik, and Didier Stricker. “Fully automatic multi-person human motion capture for vr applications”. In: *International Conference on Virtual Reality and Augmented Reality*. Springer. 2018, pp. 28–47.

[10] Jilliam M Díaz Barros, Jason Rambach, Pramod Murthy, and Didier Stricker. “In-Domain Inversion for Improved 3D Face Alignment on Asymmetrical Expressions”. In: *IEEE 18th International Conference on Automatic Face and Gesture Recognition (FG)*.

Supervised Student Works

[1] Sheela Raju Kurupathi. “Generative Model for Human Pose and Clothing using GAN”. Master’s thesis. RPTU Kaiserslautern-Landau, Aug. 2019.

[2] Siddharth Ingle. “3D Human Pose Estimation in the Wild”. Master’s thesis. RPTU Kaiserslautern-Landau, Sep 2019.

[3] Youssef Farag. “Human Pose Estimation on Embedded Devices”. Media Engineering and Technology Faculty. Bachelor’s thesis. German University in Cairo, Sep. 2019.

[4] Ankit Kumar. “3D Human Pose Estimation with Uncertainties”. Master’s thesis. Informatik, RPTU Kaiserslautern-Landau, Dec. 2020.

[5] Asmita Mittal. “Probabilistic Modeling of Human Motion Dynamics”. Master’s thesis. Informatik, RPTU Kaiserslautern-Landau, Apr. 2021.

[6] Shriya Kak. “Weakly Supervised 3D Face Reconstruction in the wild”. Master’s thesis (Co-supervised). Informatik, RPTU Kaiserslautern-Landau, Aug. 2021.

[7] Vinay Balasubramanian. “Facial Landmark Detection by Reconstruction”. Master’s thesis, (Co-supervised). Informatik, RPTU Kaiserslautern-Landau, Feb. 2022.

[8] Muhammad Umer Javaid. “Probabilistic 3D Egocentric Pose Estimation”. Master’s thesis. Informatik, RPTU Kaiserslautern-Landau, Aug. 2022.

[9] Ashish Shukla. “Efficient 3D Human Pose Estimation on Mobile Devices”. Fakultät EIM, Institut für Elektrotechnik und Informationstechnik. Master’s thesis. Universität Paderborn, Sep. 2022.

[10] Xue Yang. “3D Human Pose Estimation using Pointclouds”. Master’s thesis. Informatik, RPTU Kaiserslautern-Landau, Oct. 2022.

[11] Shubham Gupta. “3D Driver Pose Estimation from IR and Depth Images”. Master’s thesis. Informatik, RPTU Kaiserslautern-Landau, Nov. 2022.

[12] Arpit Lekhra. “Towards 3D EgoCentric Human Pose Estimation”. Master’s thesis. Informatik, RPTU Kaiserslautern-Landau, Mar. 2023.

Index

6D Rotation, 215

Action Proposal Networks, 86
Angle error, 220
Axis angle, 214

Blazepose, 77

Code Predictor Network, 140
Convolution Vision Transformer, 141

Deep Ensembles, 29
Deep Neural Networks
    EgoPoseNet, 209
    PointNet++, 207
    Stacked Hourglass, 199
    WASP, 209
Divenet
    Action Localzation, 87
    Body Angular Rotation, 108
    COM Trajectory, 98
    Diving Plane Homography, 103
    Peak Height, 99
    Pose2D, 71
    Pose3D, 200
DREICAP-3D, 126
DSV Dataset, 69

EgoPoseNet, 175
Euler angles, 213

FCN, 38

Graph Neural Networks, 85

Heatmap representations, 19

IAT Dataset, 70
ITOP, 159

Keypoint representation, 18
Kinematic, 20

Look-up Maps, 36
LSTM, 85

MobileHumanPose, 200
Monte Carlo Dropouts, 28
Monte-Carlo Dropouts, 183
MPE, 219
MPJPE, 220

PAMPJPE, 220
PCK-h, 219
Pose Refiner Network, 142

Recurrent Neural Networks, 85
Rotation 6D, 216
Rotation Matrix, 213

sampling
    farthest point, 156
    random, 155
SCAPE, 24
Skeleton Graphs, 19
SMPL, 23
SOMA, 131, 204
SPIN, 202
Sportscap dataset, 70
SVIRO, 125
Swing-Twist Decomposition, 34

TCN, 86
TICaM, 124

Unit Quaternions, 214

VIBE, 203
Volumetric Heatmap Autoencoder, 138

xR-Ego Network, 174