Dissertation

Human Pose Estimation in Images and Videos for Sports Analytics

2D Keypoint and 3D Mesh Estimation for Challenging Scenarios and Extreme Poses

Katja Ludwig

Faculty of Applied Computer Science
University of Augsburg

Advisor:	Prof. Dr. rer. nat. Rainer Lienhart
Reviewers:	Prof. Dr. rer. nat. Rainer Lienhart
	Prof. Dr. rer. nat. Elisabeth André
	Niccolò Bisagno, PhD, Assistant Professor, University of Trento, Italy
Thesis defense:	June 2nd, 2025
Examiners:	Prof. Dr. rer. nat. Rainer Lienhart
	Prof. Dr. rer. nat. Elisabeth André
	Niccolò Bisagno, PhD, Assistant Professor, University of Trento, Italy
	Prof. Dr. phil. Annemarie Friedrich

Abstract

Since the emergence of professional sports competitions, athletes have continually sought to enhance their skills to achieve success. In recent decades, the widespread availability of cameras and smartphones has made video and image analysis an integral part of sports training, helping athletes to assess their performance and analyze potential for improvement. Since the rapid pace of athletic movements often makes it nearly impossible for the human eye to capture every detail in real time, even for coaches which excel at identifying errors and areas for improvement, video or image based analyses can help. Video recordings allow for detailed analysis through slow motion and repeated replays, enabling comprehensive feedback to athletes. However, such manual analyses are both time-intensive and laborious. This thesis explores the application of computer vision techniques to automate motion analysis in the demanding context of sports. The focus of this thesis is on individual sports, where tracking the athlete's posture is critical for movement evaluation. Specifically, we investigate the tasks of 2D Human Pose Estimation (2D HPE) and 3D Human Mesh Estimation (HME) within the unique challenges posed by individual sports disciplines, including rapid motions and extreme body poses.

In the first part of this thesis, we focus on 2D HPE with imperfect data. Since niche sports often lack financial resources, the recordings are often of low quality due to outdated camera equipment. Further, the amount of annotations is mostly limited, due to the same reasons and time constraints of the coaches. Therefore, we propose two methods to tackle these problems. For ski jumpers, we create an approach that combines a 2D HPE model with heuristics and a robust optimization method to automatically estimate important flight parameters from low-quality videos. Moreover, we propose two competitive methods to train a 2D HPE model in a semi-supervised manner with just 50 labeled images for triple and long jump athletes.

The second part of this thesis proposes a new method to estimate arbitrary keypoints on the human body and sports equipment. Such keypoints can be used to improve the robustness of analyses relying on single keypoints and enabling new types of analyses which require more knowledge of the body pose than just a simple skeleton. We leverage the capabilities of Transformer networks and are able to train a model that can estimate an arbitrary amount of freely selectable points on the human surface. We introduce a new metric to measure the ability of the model to estimate the keypoints at the correct distance from the border of the corresponding body part and prove that our model performs well on this task. With these new capabilities, new types of human motion analyses can be performed, e.g. involving the body outline of athletes like in running analyses.

However, the keypoints are still limited to two dimensions, which is sufficient for many analyses, but not for all. Therefore, in the third part of this thesis, we propose a method

to estimate 3D human meshes of athletes which ensures consistent body shapes for each athlete. Investigations of existing HME models and even datasets show that body shapes of the same person are inconsistent in almost all cases, meaning that the basic body shape like bone lengths of forearms, thighs, etc., changes. Therefore, we create a new model that can estimate a consistent body shape from anthropometric measurements. Based on this model, we propose a novel HME method that outperforms existing HME models in the sports domain by a large margin.

In summary, this thesis deals with the estimation of human poses and meshes in the challenging context of individual sports. We propose solutions for 2D HPE with imperfect data as it is common in niche sports and introduce new methods for estimating arbitrary keypoints on the body and equipment of athletes, enabling new types of analyses. Finally, we propose a novel HME method that ensures consistent body shapes for each athlete, outperforming existing methods in the sports domain. The methods proposed in this thesis can be used to automate motion analysis in individual sports, providing coaches and athletes with valuable insights to improve performance.

Acknowledgments

First and foremost, I would like to express my gratitude to my advisor, Prof. Rainer Lienhart, for his guidance, support, and encouragement throughout my journey as a PhD student. I deeply appreciate that he gave me the opportunity to pursue my research under his mentorship and for his valuable advice shaping my research path, refining my publications, and navigating the many challenges of academia. His insights and expertise played a key role in the completion of this thesis. I also want to thank Prof. Elisabeth André and Niccolo Bisagnò for their valuable time and effort spend to review this thesis.

Furthermore, I want to acknowledge my colleagues Julian Lorenz, Daniel Kienzle, Robin Schön, and Mrunmai Phatak who accompanied me during my journey and have become my friends during that time. I am grateful for the time we spent together in and outside the lab, the countless discussions we had about each of our research topics, the constructive feedback and new ideas they provided, their time and support in proofreading my papers and this thesis, and the support we gave each other. They made my time at the chair enjoyable and rewarding, even during the more challenging moments. I would also like to extend my gratitude to my former colleagues Dr. Moritz Einfalt, Sebastian Scherer, Dr. Philipp Harzig and Dr. Stephan Brehm who introduced me to the fascinating fields of computer vision and deep learning and helped me to get started with my research and the daily business as a PhD student. They also provided me valuable feedback and support during the first years of my research, and they are still open for discussions and help whenever I need it.

The main focus of my thesis is the application of computer vision models to the field of sports. I want to thank my project partners for their collaboration and acquiring and sharing their data, including the Institute of Applied Training Science Leipzig and the Olympic Training Center Hessen. Their support and cooperation were critical in enabling this research.

Finally, I want to thank my family and close friends, especially my parents, for their continuous support, encouragement, and understanding throughout my journey. They have always been there for me, even during the hard times, and helped me to overcome challenges and stay motivated such that I could complete this thesis. I am grateful for their presence in my life.

Contents

Glossary

AP	Average precision. A generic metric that is used for performance evaluation. 17, 82
CNN	Convolutional neural network. 22, 147
DARK	Distribution-aware coordinate regression for keypoints. A method to extract the sub-pixel precise location of joints from heatmaps with Taylor series expansion. 64, 81, 90
DNNs	Deep artificial neural networks. 1
EMA	Exponential moving average. 7, 36, 63, 90
HME	3D Human Mesh Estimation. 2, 4, 107, 117, 148
HPE	Human Pose Estimation, either in 2D (2D HPE) or 3D (3D HPE). 1, 15, 22, 31, 45, 55, 71, 107, 117, 147
HRNet	High-Resolution Network. A deep learning model that uses high-resolution representations throughout the whole network. 32, 52, 58
IK	Inverse Kinematics. A computational method used to determine the joint configurations required to achieve a desired position and orientation of a given set of joints. 9, 118, 119, 134, 149
LLM	Large language model. 150
Mask R-CNN	Mask Region-based Convolutional Neural Network. A region-based deep learning model with a branch for predicting segmentation masks per region of interest. 22
Mean-PCA	(Mean) Percentage of Correct Angles. A metric to evaluate the accuracy of 2D angles between body parts analogous to PCK. 26
MLP	Multi-layer perceptron. 58, 79, 99
MPJPE	Mean per-joint position error. A standard metric to compare estimated and annotated 3D human poses. 112, 118
MSE	Mean squared error. 35, 130, 162
MTE	Mean thickness error. A metric to evaluate the location of keypoints relative to the border and the center of the corresponding body part. 49, 81, 91, 161

MVE	Mean vertex error. A metric to evaluate the location of vertices of the 3D human mesh. 112, 131
NN	Neural network. 128
OKS	Object keypoint similarity. A metric to compare pose estimates to annotated poses. 16, 48, 68, 80
PA-MPJPE	Procrustes aligned mean per-joint position error. A variant of the MPJPE metric. 112
PCA	Principal component analysis. A statistical method to reduce the dimensionality of data. 15, 19, 125
PCK	Percentage of correct keypoints. A metric to evaluate the fraction of keypoints that are located with sufficient spatial precision in an image. 7, 16, 31, 37, 48, 64, 74, 80, 91, 99, 148, 161
PCT	Percentage of correct thickness. A metric to evaluate the location of keypoints relative to the border and the center of the corresponding body part analogous to PCK. 49, 81, 91, 99, 161
RANSAC	Random Sample Consensus. A method to robustly estimate the parameters of a mathematical model from a set of observed data. 15, 18, 23, 27, 147
ReLU	Rectified linear unit. An activation function used in deep learning models. 99
ResNet	Residual neural network. A deep learning model that uses residual blocks to improve training and performance. 22
SMPL	Skinned multi-person linear model. A model and compact parametrization for 3D mesh representations of human bodies. 109, 120
SMPL-X	SMPL expressive. An extension of the SMPL model that includes facial expressions and hand poses and a more fine-grained mesh. 9, 109, 118, 148
SVR	Support vector regression. A classical machine learning method that finds a function within a specified margin of tolerance to approximate the relationship in the given data. 9, 128
UU	Uplift and upsample. A 2D-to-3D uplifting model. 134
ViT	VisionTransformer. A transformer-based model for computer vision tasks.. 7, 45, 57, 75, 86

1 Introduction

Passion and interest for sports is a common trait among people worldwide. Sports are not only a form of entertainment, but also a way to stay healthy and fit. They can be a source of inspiration and motivation, and can also be a way to bring people together, regardless of their background, culture, or beliefs. Being a fan of a sports team, a specific athlete, or the athletes of the own country unites people from different social classes and generations. Public interest makes sports attractive to sponsors, such that professional athletes of popular sports are able to make a living as long as their performance is good enough to attract sponsors and fans.

The rise of Deep Neural Networks (DNNs), especially in computer vision applications, has changed a lot in sports. On the one hand, DNNs are used to improve the experience of viewers and fans. Only a small amount of fans is able to watch sports highlights live in the stadiums, but many more can watch the games and competitions on TV or online. Large effort is put into the production of sports broadcasts, such that the viewers can follow the game as if they were in the stadium. DNNs can be used to enhance the experience of these remote viewers. Applications of computer vision include automatically detecting and tracking players to display their running paths, providing additional information about the athletes or the game, investigating critical situations and referee decisions, and to create highlights.

On the other hand, DNNs are also used by the athletes and coaches themselves. They are facing the constant pressure of improving their performance to keep up with the competition. Detailed analyses of their own performance and the performance of their competitors is crucial to identify weaknesses and strengths, and to develop strategies to improve. Using DNNs to analyze the performance in detail, and provide feedback about improvements and mistakes to the athletes and coaches, can make the difference between victory and defeat.

In individual sports, such analyses are mainly based on the athlete's movements. The movements of the athletes are captured by means of cameras, and the recorded video data is processed by DNNs to extract the necessary information. The most important information is the pose of the athlete, which describes the position of the joints and body parts of the athlete at a specific point in time. The exact selection of the joints depends on the sport and the specific requirements of the analysis. The task of estimating such a pose from an image or video is called Human Pose Estimation (HPE) in the computer vision community. Apart from image classification, object detection, and semantic segmentation, HPE is one of the main benchmarking tasks in computer vision. Therefore, it has gained a lot of attention in the research community in the last years and many models have been proposed to solve this task, resulting in better and better results on standard benchmark datasets.

With the rapid advances in deep learning and computer vision, the capabilities of HPE models have improved significantly. Starting with estimation of 2D poses of single humans in the pixel space of images, research moved on to solve more complex tasks including the simultaneous estimation of poses from multiple humans in the same image, the estimation of poses with fine-grained details like hands or faces, and the estimation of 3D poses from monocular 2D videos. Moreover, a similar task called human mesh estimation (HME) has been introduced, which estimates a 3D mesh of the human body from a 2D image. This task does not focus on main joints like HPE, but tries to capture the whole body surface.

However, most research is focusing on everyday poses and activities. In most cases, applying such models to sports data results in poor performance, because the poses in sports are often more challenging to estimate. At the same time, sports analyses require precise data to be useful for the athletes and coaches. Therefore, this thesis is dedicated to the development of models that are able to tackle the challenges of sports data and provide accurate pose estimates. Our goal is to leverage the recent advances in computer vision to improve automatic sports analyses and to provide athletes and coaches with the necessary tools to improve performance.

1.1 Problem Definition and Challenges

In this section, we want to give a brief overview of the main problems and challenges that we are addressing in this thesis.

2D Human Pose Estimation The goal of 2D Human Pose Estimation is to locate selected keypoints of the human body in images or videos. Such keypoints are typically visually distinct features like joints (e.g, elbow, knee, ankle) or body landmarks (e.g., nose, eyes, big toe tip, head top). The task is to estimate the 2D coordinates of this sparse set of keypoints in the image plane. Since we want to use 2D HPE for sport analyses in this thesis, we require 2D HPE models with highly accurate keypoint detections. Although huge advances have been made in the recent years, including powerful Convolutional Neural Networks (CNNs) and Transformer architectures, 2D HPE remains a challenging task on its own.

One challenge that 2D HPE faces is the spatial resolution of DNN models. Achieving highly precise keypoint detections requires high-resolution features throughout the model. This has a direct impact on the model architectures. Obtaining a large receptive field and high-resolution features at the same time is challenging. This problem can be simplified a bit by using crops of the humans in the input image instead of the whole image as an input to the HPE model, but this introduces the additional step of detecting the humans in the image first. Further, the HPE model needs to be run multiple times, once for each crop. Moreover, achieving a high spatial precision with only cropped inputs is still challenging.

Another challenge for 2D HPE is occlusion, especially self-occlusion. For example, if a human is viewed from the side, the keypoints on the far side of the body are partly

or fully occluded by the body itself. 2D HPE models need to be able to estimate all required keypoints, even if they are not visible in the image. This is a challenging task, because the model needs to understand the context of the image and the human body to estimate all keypoints correctly.

Sports Data Apart from the general challenges of 2D HPE, sports data introduces additional challenges. Human motion analyses have already been carried out for a long time in some sports disciplines. However, most of the time, these analyses are done manually by human experts, which is a time-consuming process. The annotations that result from this manual process can easily be used as labels for training HPE models, which is an advantage. However, some annotations are specifically made based on assumptions or internal knowledge for the analyses and do not match the requirements of the HPE models. An example is the annotation of ski jumpers during their flight phase, recorded from the side. Coaches usually only annotate the keypoints on the near side. However, if body parts or skis from both sides are visible, the annotators have decided to set the keypoint in the middle of both visible keypoints. This semantic is extremely difficult for HPE models to learn, which results in poor performance in these scenarios and the necessity to reannotate this data.

In contrast to scenarios where already existing manual motion analyses should be replaced by automatic HPE models, there are also sports disciplines where no motion analyses are carried out yet. Consequently, the available data is entirely unlabeled. Since annotating is both time-intensive and constrained by the limited availability of coaches, only a small amount of data can realistically be annotated. This leads to the creation of small datasets. In contrast, standard benchmark datasets for everyday activities are orders of magnitude larger, which is an essential requirement to train modern DNNs with millions of parameters. Training HPE models on these limited datasets results in poor generalization to new data, presenting a significant challenge for sports-related applications.

Another significant challenge in sports data is the quality of the images. Sports events or trainings are often captured under difficult conditions, including poor lighting, extreme weather, rapid movements, and frequent occlusions. Additionally, niche sports often face limited funding, resulting in the use of outdated or unsuitable cameras. These factors lead to noisy and low-quality images, making it challenging for HPE models to accurately estimate keypoints. Furthermore, variations in camera angles across different viewpoints exacerbate the difficulty, hindering the models' ability to generalize effectively to new perspectives.

Closely linked to image quality is the challenge of fast and diverse motions. Athletes in sports perform rapid and complex movements, which are often not present in everyday activities. These movements can lead to motion blur, which degrades the image quality. Additionally, the poses of athletes in sports are significantly more diverse than in everyday activities. For example, gymnasts perform maneuvers where they are upside down or extremely bent, high jumpers arch their backs and jump almost horizontal over the bar, and ski jumpers are stretched out during their flight in an unnatural angle. Such

poses require HPE models to learn an exceptionally broad spectrum of poses, complicating their ability to capture the underlying semantics of the human body. The first part of this thesis focuses on tackling the challenges of sports data and improving the performance of HPE models for the sports domain.

Detection of Freely Selected Keypoints The second part of this thesis is built around the problem of detecting freely selected keypoints on the human body. 2D HPE models are typically trained to detect a fixed set of keypoints. These keypoints need to be annotated in the input data and the network is trained to predict the positions of these keypoints. Typical sets of keypoints contain 10 to 40 keypoints that we call standard keypoints. Some applications like specific sport analyses require additional keypoints, e.g., on the boundary of body parts to analyze the movements in a more fine-grained manner. These keypoints are not part of the standard set of keypoints and are not annotated in the data. Annotating such a large number of keypoints is infeasible. Definitions like all points on the boundary of a body part (e.g., the outline of the legs) result in a continuous spectrum of keypoints. Although restricted to pixel coordinates, such definitions can result in huge amounts of keypoints, especially in large images. Therefore, annotating these keypoints is not feasible. An alternative would be to use the set of standard keypoints combined with a segmentation model that predicts segmentation masks for the body parts. However, datasets with fine-grained segmentation masks of body parts are rare. Therefore, models that are trained on the body part segmentation task have poor performance, especially for challenging images. Specific keypoints derived from such segmentation masks are not reliable and therefore not usable for downstream tasks.

Apart from deriving arbitrary keypoints from segmentation masks and standard keypoints, models can be trained to detect arbitrary keypoints directly. This strategy introduces other challenges. First, the number of keypoints is not fixed, which requires to design a model architecture that can handle an arbitrary number of keypoints. Most 2D HPE models are designed to predict a fixed number of keypoints and each of their outputs are designated to a specific keypoint. Second, labels for every possible keypoint need to be generated. Annotating them is not feasible as mentioned above, therefore another method is needed. Generating them based on segmentation masks is possible, but error-prone, since the segmentation masks are often of poor quality. This needs to be taken into account when training a model.

Human Mesh Estimation in Videos The third part of this thesis shifts the focus from 2D HPE to the third dimension, especially Human Mesh Estimation (HME). Detecting all keypoints on the human body in three dimensions results in the detection of the whole body surface, which is essentially the task of HME. HME is a more complex and challenging task than 2D HPE. It requires the estimation of a fine-grained mesh of the human body in 3D space, which consists of a vast amount of vertices. Most HME models simplify the task by disentangling the estimation of the body pose and the body shape, while modeling the human body by a predefined template mesh which can be adjusted

to the estimated pose and shape. This simplification makes the task feasible, since it reduces the output space dimensions by a large margin.

However, the task remains challenging. The main challenge is the lack of ground truth data. While 2D HPE can be easily annotated by clicking at the correct joint positions in images, 3D annotations are much more difficult to obtain. The most common method to obtain 3D keypoint annotations is to use motion capture systems, which are expensive and require a controlled environment. They are mainly applicable in lab environments, which results in poor generalization to in-the-wild data. Another option is to use already trained 3D HPE models to obtain first guesses for the 3D keypoints for images from multiple camera views and apply correction mechanisms. This strategy is applicable to in-the-wild data, but requires synchronized cameras and results in less accurate data. Such 3D keypoint annotations can be used to train 3D HPE models and weakly supervised HME models, but for a full HME model training, they contain not enough information. HME ground truth data needs to contain not only the keypoints, but the whole body surface. This data is even more difficult to obtain. Some small datasets exist which are created by obtaining 3D scans of humans and afterwards fitting the template mesh to the scans. However, these datasets are small and not diverse enough to train a model that generalizes well. Therefore, the (additional) usage of synthetic datasets with artificially generated humans is prevalent in the research community. Further, a combination of trained HME models, 3D and 2D HPE models, heuristics, different camera views, and manual annotations are used to create labels, but the quality of these labels is not ideal for training accurate models.

A shared challenge for both HME and 3D HPE models is the extraction of the third dimension from monocular 2D images. HME models often operate similarly to depth estimation models, using single images to infer the missing third dimension using visual features. In contrast, many 3D HPE models leverage the time domain by first extracting 2D poses from a video and then estimating a single 3D pose from a sequence of 2D poses. This approach effectively utilizes temporal information, akin to multiple camera views, by making use of consistent properties of the human body, e.g., immutable bone lengths. However, this method is not directly applicable to HME models, as the 2D pose alone is insufficient to reconstruct the body surface. Consequently, recent HME models still rely on single images, since incorporating long frame sequences is impractical due to the high computational cost. This limitation leads to inconsistencies between consecutive frames, presenting a significant challenge that this thesis aims to address.

1.2 Contributions

The contributions of this thesis can be summarized as follows.

Reliable Determination of Ski Jumpers' Flight Parameters Ski Jumping is a niche sport, which is the reason why even professional ski jumpers need to manage their training analyses with low financial resources. Therefore, coaches are still using old low-quality and low-resolution cameras to evaluate the performance of their athletes. They

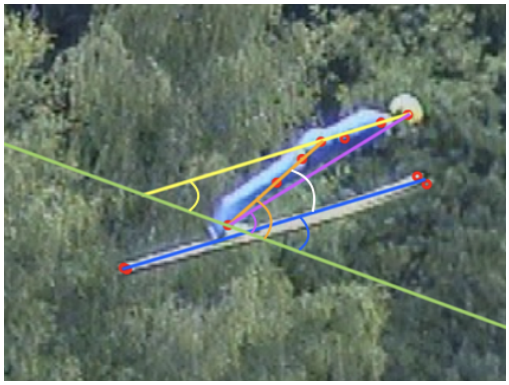


Figure 1.1: Relevant angles of a ski jumping pose: upper body angle (yellow), lower body angle (orange), total body angle (purple), average ski angle (blue) and angle difference between lower body and skis (white). The green line represents the tangent to the flight trajectory of the athlete.

use these videos to manually annotate some keypoints of the athletes and calculate the flight parameters of the ski jumpers based on these. Important flight parameters contain the angle of the lower body and the skis, the angle of the upper, lower and total body to the flight trajectory and the ski angle, as visualized in Figure 1.1. These parameters are crucial for the performance of the ski jumpers. We can show that executing a PCA on these parameters can roughly indicate the jump length [52]. Since manual annotation of keypoints in multiple frames and for multiple camera views is time-consuming, such analyses are only available for top-class athletes. We leverage the already annotated keypoints from manual analyses to train a 2D HPE model on ski jumpers. Existing 2D HPE models fail on this task since the appearance of ski jumpers is very different from everyday poses. Our trained model performs by far better, but still makes errors, often due to low image quality. The manual workflow of the coaches includes selecting the best frames for each camera view and calculating the flight parameters as the mean of the angles in each annotated frame. We incorporate a similar approach. First, we filter the model detections with various heuristics to remove obviously wrong poses. Next, we use the robust technique of RANSAC to calculate the flight parameters. We evaluate two variants of RANSAC. One variant is based on the detected keypoints and the other one based on the flight parameters. Our best method estimates 99.3% of the flight parameters correctly within an error margin of 5 degrees. A former approach tried to solve the same problem, but produced worse results that were not usable by the coaches for a reliable analysis. With our approach, this is now possible, and flight parameter analyses can be made available not only for top class athletes, but also for any athlete using a ski jumping hill equipped with cameras.

Improving 2D HPE Models on Limited Labels with Semi-Supervised Learning Unlike ski jumpers, coaches in other niche sports disciplines often cannot utilize motion analysis for their athletes due to time or financial constraints. In many sports, general-purpose

2D HPE models are unsuitable because of the significant domain gap between everyday datasets and specific sports contexts. To address this challenge, we explore strategies for achieving optimal detection performance with minimal annotated data, thereby reducing the effort required for manual annotation. Given that collecting image material is relatively straightforward, our focus lies on semi-supervised learning approaches, which, to the best of our knowledge, had not been extensively explored for HPE at the time of our research [58].

We first investigate a self-training-based method [96]. Initially, a baseline model is trained using labeled data. This model is then employed to generate pseudo labels for unlabeled images. To ensure the quality of these pseudo labels, we filter them by comparing model outputs across different augmentations, as this approach proves to be more reliable than relying solely on model confidence. After the training converges, we update the pseudo labels using the newly trained model and repeat the process for three to four iterations until convergence. Additionally, we experiment with mean teacher training [86], which employs an exponential moving average (EMA) model to produce pseudo labels, which is called the teacher model. The student model is trained on augmented images, while the teacher model generates labels on non-augmented images.

Both approaches are evaluated on a dataset containing images of triple and long jump athletes, using only 50 labeled images. The self-training approach demonstrates significant better performance, closing 60% of the gap between fully supervised training on all available images and fully supervised training with only 50 labeled images. Specifically, we achieve a Percentage of Correct Keypoints (PCK) at threshold 0.1 of 88.6%. This is a good result given the limited labeled data, and it is sufficiently robust for coaches to integrate the model into their motion analysis workflows.

Directly Detecting Keypoints between Standard Keypoints Using a fixed set of keypoints limits not only the possible analyses, but also the performance of several analyses. Regarding the flight parameters of the ski jumpers which are angles calculated from keypoints, using more than just three keypoints to estimate the angle would result in a more robust estimation. Therefore, we create a new type of 2D HPE model which is able to estimate not only the standard keypoints, but also the continuous straight line of points between two standard keypoints. A visualization can be found in Figure 1.2, where these intermediate keypoints are colored in white. Since all points are detected individually, the result is not as straight as the line. This has the advantage that errors in some keypoints can be compensated by other detections for the calculation of derived parameters like the flight angles.

Estimating continuous points is a challenging task and has not been studied before our work. Typically, the output dimension of HPE models is fixed, resulting also in a fixed number of output keypoints. We solve this problem by using a Transformer [89] architecture, which is able to handle input sequences of different lengths. Apart from visual tokens which are the typical input for Vision Transformers (ViT) [18], we add a token for each desired keypoint to the input [53]. In contrast to TokenPose [48] which uses a similar architecture, we do not learn an input token for each standard keypoint,

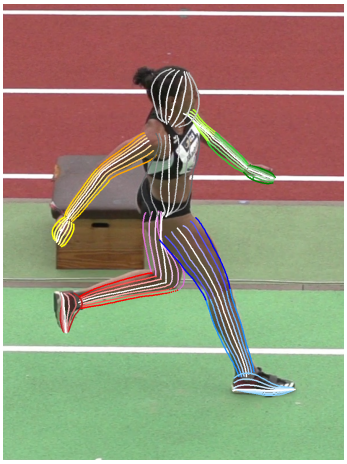


Figure 1.2: Example detection results for arbitrary keypoint detection, visualized with equally spaced lines to both sides of each body part of the athlete including the outer boundary in pure color and the central line in white with a color gradient from the boundary to the intermediate keypoints (white). Different body parts are visualized with different colors.

but a mapping from a human-readable keypoint query vector to the corresponding token. The keypoint query vector can be designed to detect any point on the line between two standard keypoints. Our model can deal with an arbitrary number of keypoint tokens. Experiments on sports datasets with ski jumpers and triple and long jump athletes as well as on the everyday activity dataset COCO show that our model is able to detect standard and intermediate keypoints with high accuracy.

Estimation of Arbitrary 2D Keypoints Intermediate keypoints help to improve some kinds of sports analyses, but are limited in their use. Therefore, we extend our method in various steps to detect any arbitrary keypoint on the human body and further on the skis of ski jumpers. We use the same architecture as for the intermediate keypoints, but change the keypoint query vectors such that they capture the whole surface of the human body and not only the intermediate line. An example is provided in Figure 1.2. We keep them human-readable, such that the necessary keypoints for any kind of analysis can be expressed with the keypoint query vectors.

Creating ground truth data for this task is challenging. Therefore, we focus on the limbs at first, since it is easier to create ground truth for them [54]. We use the standard keypoints together with segmentation masks to create ground truth labels. Next, we include the skis and incorporate strategies to deal with only partly correct segmentation masks. In this way, we are able to leverage low quality automatically generated segmentation masks for our task [55]. Lastly, we extend our method to detect arbitrary keypoints on the whole body and include head, hands, feet, and torso to our approach [57]. We experiment with different strategies to encode the keypoints. Moreover, we improve the detection on bent arms and legs by improving the label generation mechanism.

This is especially important for sports analyses, since athletes often perform poses with heavily bent limbs.

Furthermore, we find that adjustments to the architecture are necessary. The ViT architecture aggregates information between all tokens. This leads to keypoint queries having an influence of each other and the detected keypoints being dependent on the other keypoints that should be detected. Since this behavior is not desired, we introduce a mechanism to make the keypoint queries independent of each other. We restrict the attention mechanism to the visual tokens similar to cross-attention [9]. This results in a robust detection of the keypoints, meaning that the detected output for each keypoint is independent of the other keypoints.

Improving HME Performance with Temporal Cues and Consistent Body Shapes Being able to detect arbitrary keypoints on the human body in images enables new motion analyses, but is still limited to two dimensions. Therefore, we aim to lift the detection of arbitrary keypoints to the third dimension, which is equal to the detection of the whole body surface or the task of HME. At first, we evaluate the ground truth quality of available 3D HME and HPE datasets. We find that the body shape of humans in the datasets is not consistent in most datasets [56], meaning that immutable parameters like bone lengths of the same person differ from frame to frame in 3D HPE datasets and the body shape parameters are not identical for subsequent frames in HME datasets. This is a problem especially in the field of sports which requires precise estimations for the motion analysis. By analyzing the results of existing HME models, we find even larger inconsistencies in the estimated body shapes. Therefore, we propose an approach to achieve consistent body shapes and accurate meshes with marginal overhead.

Since professional athletes are measured anyway, we propose to leverage anthropometric measurements to generate a fixed body shape per person for the SMPL-X body model [69]. Since the body shape parameters are not human interpretable, we learn a small MLP and use Support Vector Regression (SVR) to generate the body shape parameters from the anthropometric measurements. The resulting models can be used to create a single set of body shape parameters per person, which we reuse for all frames showing this person. Using these body shape parameters instead of the estimated ones can already improve the results of existing HME models.

However, the results of HME models are not satisfying regarding the precision of the keypoints. 3D HPE models outperform them. A reason is that 3D HPE models operate on sequences of 2D poses, whereas HME models operate on single images. Therefore, we propose to leverage 3D HPE models for HME. We use the 3D HPE model Uplift and Upsample [21] to estimate 3D poses. Next, we use Inverse Kinematics (IK) to adjust a SMPL-X body model to the estimated 3D poses. We discard the IK result for the body shape and keep the body pose, which contains further rotation information that is missing in the 3D HPE poses. This body pose joined with the body shape parameters estimated from the anthropometric measurements results in a consistent body shape and accurate mesh. Evaluations on two public 3D sports datasets show that our approach outperforms existing HME models.

1.3 List of Publications

Most parts of this thesis have been published in the academic literature and have been presented at international conferences, despite the very last paper which is currently under review. The following list gives an overview of the publications that are part of this thesis.

Robust Estimation of Flight Parameters for Ski Jumpers. [52], Katja Ludwig, Moritz Einfalt and Rainer Lienhart, IEEE International Conference on Multimedia and Expo (ICME) Workshops 2020, London, UK, July 2020.

Self-Supervised Learning for Human Pose Estimation in Sports [58], Katja Ludwig, Sebastian Scherer, Moritz Einfalt and Rainer Lienhart, IEEE International Conference on Multimedia and Expo (ICME) Workshops 2021, Shenzhen, China, July 2021.

Detecting Arbitrary Intermediate Keypoints for Human Pose Estimation with Vision Transformers [53], Katja Ludwig, Philipp Harzig and Rainer Lienhart, Winter Conference on Applications of Computer Vision (WACV) Workshops 2022, Waikoloa, HI, January 2022.

Recognition of Freely Selected Keypoints on Human Limbs [54], Katja Ludwig, Daniel Kienzle and Rainer Lienhart, IEEE/CVF International Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), New Orleans, LA, June 2022.

Detecting Arbitrary Keypoints on Limbs and Skis with Sparse Partly Correct Segmentation Masks [55], Katja Ludwig, Daniel Kienzle, Julian Lorenz and Rainer Lienhart, Winter Conference on Applications of Computer Vision (WACV) Workshops 2023, Waikoloa, HI, January 2023.

All Keypoints You Need: Detecting Arbitrary Keypoints on the Body of Triple, High, and Long Jump Athletes [57], Katja Ludwig, Julian Lorenz, Robin Schön and Rainer Lienhart, IEEE/CVF International Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), Vancouver, BC, Canada, June 2023.

Leveraging Anthropometric Measurements to Improve Human Mesh Estimation and Ensure Consistent Body Shapes [56], Katja Ludwig, Julian Lorenz, Daniel Kienzle, Tuan Bui and Rainer Lienhart, IEEE/CVF International Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), Nashville, TN, June 2025.

1.4 Thesis Outline

This thesis consists of three parts. The main contributions of all three parts are already listed in sequential order in Section 1.2. In this section, we therefore give only a brief overview of the general outline.

The first part of this thesis focuses on standard 2D HPE in sports scenarios with imperfect data. In Chapter 2, we introduce the foundations for our approaches. We explain the task of 2D HPE and the corresponding metrics formally in Section 2.1. We further introduce the common benchmark dataset COCO and the sports datasets of ski jumpers and triple and long jump athletes that we use. Next, we explain RANSAC, a general robust method to estimate parameters of an underlying model. Moreover, we shortly describe a dimensionality reduction technique called Principal Component Analysis in Section 2.4 and finish with a general description of learning paradigms. In Chapter 3, we present our approach to robustly estimate the flight parameters of ski jumpers. We introduce the problem and review related work in Sections 3.1 and 3.2. We describe our model architecture and present evaluation results. The latter contains results regarding the image-wise keypoint detection and the flight parameters per camera view. We further include an approach to roughly predict the jumping distance based on the flight parameters and summarize all our findings. In Chapter 4, we present our approach to improve 2D HPE models trained with limited labels with semi-supervised learning. After a short introduction and description of related work, we present our semi-supervised learning approaches in Section 4.3. At first, we describe the self-training approach based on pseudo labels and how we select the pseudo labels for our training. Next, we introduce the mean teacher training approach. In Section 4.4, we describe our experimental setup and present the results regarding both semi-supervised approaches. We finish with a short summary of our findings.

Part II focuses on the detection of keypoints beyond the standard set of keypoints. In Chapter 5, we describe the foundations for this part, starting with formal task definitions and followed by explanations for metrics, the used datasets and the architecture of TokenPose, which our approaches are based on. The section on datasets introduces two datasets that we curated and published. In Chapter 6, we present our approach to detect arbitrary intermediate keypoints. We introduce the task, related work and our method. In Section 6.3.1, we investigate the learned keypoint tokens from TokenPose and analyze why combinations of them are not sufficient to detect intermediate keypoints. Therefore, we introduce our own architecture in Section 6.3.2. We present our experiments in Section 6.4 with results for the IAT-Skijump v2, the BiSp-Jump v2, and the COCO dataset and conclude with a short summary. Chapter 7 describes our path to detect arbitrary keypoints on the human body. We introduce the task and related work at first. In Section 7.3, we present our approach to detect arbitrary keypoints on the limbs. We describe the label generation mechanism and two ways to represent desired keypoints as an input for the model. Next, we introduce the model architecture to solve this task and the results on two datasets. In Section 7.4, we extend our approach to detect arbitrary keypoints on the skis of ski jumpers. We start off with a description of the adapted ground truth and keypoint query generation mechanisms. In Section

7.4.3, we explain how we adapt the attention mechanism to be able to make independent keypoint estimations. We further present strategies to deal with situations with only few segmentation masks, training jointly on arbitrary, intermediate, and standard keypoints. We present results for all these approaches in Section 7.4.5. Section 7.5 describes the final improvements to detect arbitrary keypoints on the whole body. We explain the adaptations to the ground truth generation process and different types to encode the chosen keypoints. We improve the evaluation scheme and present the final results in Section 7.5.4. The chapter is concluded with a short summary.

The third part of this thesis is dedicated to 3D HPE and HME. In Chapter 8, we introduce the necessary foundations. We start with a formal definition of 3D HPE and HME. Further, we provide a short explanation of the SMPL-X body model and the metrics used to evaluate the models. We further introduce the datasets used in this part. In Chapter 9, we present our approach to improve HME models with consistent body shapes. We introduce the problem and related work at first. Next, we analyze errors in 3D human ground truth for multiple datasets. In Section 9.4, we present our approach to leverage anthropometric measurements to generate consistent body shapes. We describe the data collection process and the model architectures that we use. In Section 9.5, we present approaches to use these body shapes for improved HME results. First, we outline methods to enhance existing HME models. We then present our novel approach, which integrates a 3D HPE model, inverse kinematics, and our custom body shape model. Additionally, we demonstrate how our approach enables fine-tuning of existing HME models for improved performance. We present the results of our approach in Section 9.5.2.3 and end with an overview of all results and a short summary.

We conclude this thesis in Chapter 10 with a summary of the key findings and a discussion of potential future work.

Part I

2D Human Pose Estimation with Imperfect Data in Sports

2 Foundations

2D Human Pose Estimation (2D HPE) is a common task in computer vision, alongside image classification and object detection. The goal of 2D HPE is to accurately detect humans and identify their specific poses in images or videos. This task has numerous applications, including human-computer interaction, healthcare, fashion, gaming, and sports. However, it is inherently challenging due to the need for precise detection of multiple keypoints on the human body. These challenges are exacerbated by imperfect data, such as occlusions, low resolution, or noise. Occlusions are particularly problematic in HPE because the human body often occludes parts of itself. Low resolution can stem from poor image quality or the small size of the human figure in the image. Noise is commonly introduced by fast movements resulting in motion blur, camera quality issues, or low lighting conditions.

In sports, these challenges are even more intensified. Fast movements are frequent, and the range of poses is highly diverse. Sports datasets often feature poses that are significantly different from those in everyday life, leading to poor generalization of HPE models trained on standard datasets. Consequently, new annotations are required to train models capable of accurately detecting sports-specific poses. However, creating these annotations is costly, resulting in smaller datasets and further complicating the task, which is another kind of imperfect data.

In this chapter, we will introduce the basics of 2D HPE. We formally define the task and the PCK metric that we use for evaluation. We will also introduce the datasets that we use in this part: COCO, IAT-Skijump and BiSp-Jump. Further, we introduce RANSAC, a technique for outlier detection that we use for dealing with imperfect detections. Next, we briefly explain Principal Component Analysis (PCA), a common dimensionality reduction technique. Moreover, we explain the differences between supervised, semi-supervised and unsupervised training. These techniques are useful when dealing with small datasets and are applied in the following.

2.1 2D Human Pose Estimation

As already described, 2D HPE aims to detect a human in an image and estimate the locations of specific keypoints on the human body. These keypoints are typically joints, such as the elbows, knees, and wrists, or other distinctive keypoints such as eyes, fingertips, ears, etc. The number of keypoints can vary depending on the dataset and the specific task. Based on these keypoints, a stick-figure representation of the human pose can be created.

2.1.1 Task Definition

A human pose is represented by the 2D coordinates of the desired keypoints on the human body. Hence, the output of a 2D HPE model are the x- and y-coordinates of each desired keypoint in pixel coordinates. Formally, if we want to estimate k keypoints, we need a model $M_{\text{2D-HPE}}$ that operates on input x such that

$$M_{\text{2D-HPE}}(x) = p^{\text{2D}} \in \mathbb{R}^{k\times 2}. \tag{2.1}$$

The input x is an image in most cases. It is most common to split videos into single frames and pass each frame as an image to the HPE model. Let h, w be the height and width of the image I, respectively. Then, $p^{\text{2D}}_{i,0}$ represents the x-coordinate of each keypoint $i \in \{1, ..., k\}$, while $p^{\text{2D}}_{i,1}$ represents the y-coordinate, whereby $p^{\text{2D}}_{i,0} \in [0, w)$ and $p^{\text{2D}}_{i,1} \in [0, h)$ for all keypoints.

2.1.2 Evaluation Metrics

There are several metrics to evaluate the performance of a 2D HPE model. The most common metrics are the Percentage of Correct Keypoints (PCK) and the Object Keypoint Similarity (OKS). The used metric depends on the dataset and the specific task. In this chapter, we focus on the PCK metric, but we also introduce OKS, since it is the standard metric for the COCO dataset which we also use in this thesis.

2.1.2.1 Percentage of Correct Keypoints

For evaluation, the Percentage of Correct Keypoints (PCK) is one of the most common metrics. PCK considers a keypoint as correct at a certain threshold t if the distance of the detected keypoint to the ground truth keypoint is less or equal than t times the distance between a reference length, typically the distance between left shoulder and right hip. Let i_1, i_2 be the indices for the keypoints spanning the reference length. Let g^{2D} be the ground truth keypoints and C be the set of correctly predicted keypoints. Then, a detection p^{2D}_j of the j-th keypoint is seen as correct at threshold t, meaning $p^{\text{2D}}_j \in C$, if

$$||p^{\text{2D}}_j - g^{\text{2D}}_j||_2 \leq t \cdot ||g^{\text{2D}}_{i_1} - g^{\text{2D}}_{i_2}||_2 \iff p^{\text{2D}}_j \in C. \tag{2.2}$$

The recall at a certain PCK threshold tells us the percentage of the keypoints that is considered correct at that threshold. Let N be the number of annotated visible keypoints in the dataset, then

$$\text{PCK@}t = \frac{|C|}{N} \tag{2.3}$$

Common thresholds are 0.1 and 0.2.

2.1.2.2 Object Keypoint Similarity

The Object Keypoint Similarity (OKS) is the primary metric for COCO evaluations. Let $\delta(\cdot)$ be a function converting boolean values to the integers, meaning its result is 1

for input *True* and 0 for input *False*. Then, OKS is calculated as

$$OKS = \frac{\sum_i exp(-d_i^2/2s^2k_i^2)\delta(v_i > 0)}{\sum_i \delta(v_i > 0)},$$

whereby d_i is the euclidean distance between corresponding ground truth and detected keypoint, v_i is the ground truth visibility flag indicating if a keypoint is visible in the image or not, s is the object scale (the square root of the person segment area, e.g., the bounding box enclosing the person or the segmentation mask of the person) and k_i a per keypoint specific constants. These constants correspond to the required precision for a keypoint to be considered correct. For example, the precision for the eyes needs to be higher than for the hips, since the eyes are smaller. The k_i values are calculated based on the standard deviation of the keypoint annotations from multiple annotators of the same image (with respect to the scale).

The metric used for evaluations is the mean average precision (AP) based on the OKS. This means that the average precision is calculated for OKS thresholds of 0.5 to 0.9 with interval steps of 0.05. The final AP is then the mean of all AP values for all thresholds.

2.2 Datasets

There is a multitude of datasets available for 2D HPE, most of them focusing on everyday life scenarios. However, sports-specific datasets are rare, since labels that require humans to annotate are expensive. In this section, we introduce the datasets that we use in this part of the thesis.

2.2.1 COCO

The Microsoft Common Objects in Context [50] (COCO or MS-COCO) dataset, is one of the most popular datasets for 2D HPE. It contains images of everyday life scenarios and is annotated with 17 keypoints. It contains over 200,000 images with 250,000 labeled person instances. For training, we use the train2017 split consisting of 57,000 images, our results are reported on the val2017 split. The annotated keypoints are nose, l./r. eye, l./r. ear, l./r. shoulder, l./r. elbow, l./r. wrist, l./r. hip, l./r. knee, l./r. ankle. Some keypoints might not be visible in the images, due to occlusions or persons only being partially in the field of view. Example images can be found in Figure 2.1.

Figure 2.1: Example images from the COCO dataset. [50]

2.2.2 IAT-Skijump

The IAT-Skijump dataset was collected and provided by the Institute for Applied Training Science (IAT) in Leipzig. The training dataset contains 10,070 annotated images from 290 jumps and the test set 3,388 images from 101 different jumps. The videos were recorded at different ski jumping hills, during multiple events and with different athletes, so their statures and dressings vary. The footage also covers a wide variety of weather and light conditions, e.g. snow, rain, fog, summer, winter, day, and night. Only few images from every video are annotated, usually 2–4 frames per camera view, whereby each video consists of 4–8 camera views. Annotated keypoints are both ski tips and tails, head, shoulder, elbow, hand, hip, knee and ankle. The annotations of the joints are only available of one side of the body (the one facing the camera). The dataset contains images of the flight phase as well as images of the athlete during inrun, where the skis are not visible and not annotated. Example images are displayed in Figure 2.2.

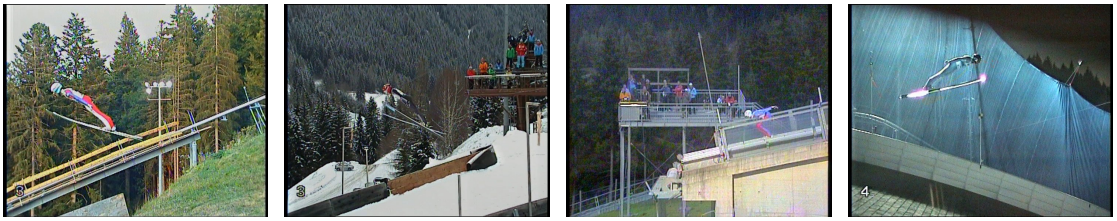


Figure 2.2: Example images from the IAT-Skijump dataset.

2.2.3 BiSp-Jump

The BiSp-Jump dataset contains video recordings of long and triple jump athletes, collected by the Olympic Training Center Hessen. It consists of 4,522 labeled images from 186 video sequences, whereby 3,154 images from 122 videos are used for training, 306 images from 18 videos for validation and the remaining 1,062 images from 46 videos as the test set. The recordings were taken during competitions or training and show various athletes and sports sites, containing indoor and outdoor sites. The recordings have a constant frame rate of 200Hz and a length between 670 and 1900 frames. All videos are annotated with 20 keypoints (r./l. eye, r./l. ear, nose, neck, r./l. shoulder, r./l. elbow, r./l.wrist, mid-hip, r./l. hip, r./l. knee, r./l. ankle, r./l. big toe, r./l. small toe, r./l. heel). We provide example images in Figure 2.3.

2.3 RANSAC

RANSAC (Random Sample Consensus) [17] is a technique for fitting a mathematical model to a dataset that contains outliers. It can also be seen as an outlier detection method. The algorithm works by iteratively selecting a random subset of the data and fitting the model to this subset. The model is then used to classify the remaining data points as inliers or outliers based on predefined rules. The process is repeated for a fixed

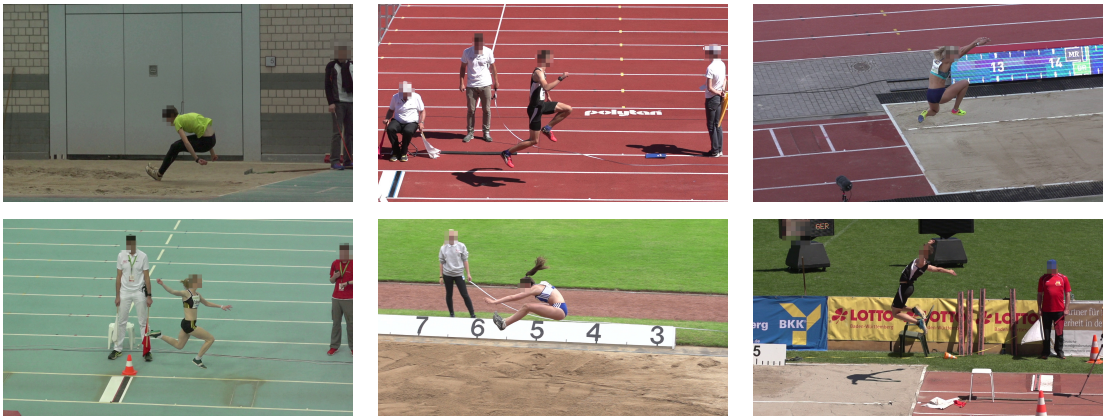


Figure 2.3: Example images from the BiSp-Jump dataset.

number of iterations, and the model with the highest number of inliers is selected. The final model is then created by fitting the model to the set of all inliers. We use RANSAC to filter out wrongly detected keypoints in a time series of poses.

2.4 Principal Component Analysis

Principal Component Analysis (PCA) [70] is a linear dimensionality reduction technique that transforms data into a new coordinate system with fewer dimensions. Formally, PCA reduces a set of N vectors $v_i \in \mathbb{R}^n$ to vectors $\hat{v}_i \in \mathbb{R}^m$ where $m \ll n$ and $i = 1, ..., N$. The reduction mechanism at first transforms the data into a new coordinate system such that the first few dimensions, known as principal components, capture the maximum variance of the data. These dimensions of the vectors are kept, while the others are discarded. The principal components represent the directions of the greatest variance, enabling the identification of the most significant features while reducing redundant information. We use PCA to perform a rough jumping distance estimation based on the flight parameters of ski jumpers.

2.5 Learning Paradigms

Typically, machine learning models are trained using labeled data. During training, the model processes some samples, and its output is compared to the ground truth labels. The difference between the model's predictions and the ground truth is then used to adjust the model's parameters. This approach is known as *supervised learning* because the labels guide and supervise the entire training process.

However, in many cases, labeled data is either scarce or expensive to obtain. To address this, techniques that require less labeled data are employed. When only a small portion of the data is labeled, the training paradigm is referred to as *semi-supervised learning*. In this approach, the model learns from both labeled and unlabeled data.

One common method in semi-supervised learning is *pseudo-labeling*, where the model generates its own labels to guide its training.

Finally, in *unsupervised learning*, no labels are available. Without labels, the scope of tasks the model can learn is more limited, and this approach is often used for pretraining. By leveraging large amounts of unlabeled data, the model is trained to learn meaningful representations of the data, which can then serve as a foundation for subsequent tasks.

3 Robust Estimation of Flight Parameters for Ski Jumpers

Body posture analysis is a widely used technique in professional sports to evaluate performance and provide recommendations for improvement. In popular sports like soccer, substantial financial resources are available to develop and deploy advanced computer vision systems. However, in niche sports such as ski jumping, financial constraints often limit access to high-quality technical equipment and sophisticated analysis tools.

In this chapter, we present an automated system for analyzing flight parameters in ski jumping. Due to limited financial resources, ski jumpers still rely on low-resolution cameras, which they position along the ski jumping hill. They manually annotate the keypoints of the athlete and the skis in the video footage, before calculating the necessary performance parameters based on these annotations. Our system automates this process. However, the task is challenging due to the poor video quality and resulting noisy frames in lots of the footage, as shown in Figure 2.2. The approach discussed in this chapter is based on the following publication, with some text passages directly taken from it:

Robust Estimation of Flight Parameters for Ski Jumpers. [52], Katja Ludwig, Moritz Einfalt and Rainer Lienhart, IEEE International Conference on Multimedia and Expo (ICME) Workshops 2020, London, UK, July 2020.

In this chapter, we present a model that robustly estimates important flight parameters for ski jumpers during their flight phase based on several camera views from the side along the ski jumpers' typical flight trajectories. A convolutional neural network for Human Pose Estimation, but also trained to detect skis, serves as a base model. It identifies 98.0% of the relevant flight parameters correctly within an angle threshold of 5°, improving by 11.6% over previous work. To deal with estimation errors due to the low image quality, a post-processing step is employed. A pose checker first removes all wrong poses by using comparisons of distances and relative positions of the detected keypoints. A second step executes two RANSAC variants. One robustly estimates the average pose and another one the average pose angles. This model lifts the detection performance to 99.3% of the relevant flight parameters within a threshold of 5°.

3.1 Introduction

Ski jumping is an Olympic discipline in which the success of athletes highly depends on the body posture during the jump. A ski jump can be divided into four phases. In the first phase, athletes slide down the in-run and gain speed. While approaching the

take-off table, ski jumpers lift their body and take off with the help of their speed and their own leap (phase 2). During launch and the following flight phase (phase 3) it is important for the athletes to position their body perfectly in order to increase lift, which is necessary to achieve a long flight distance. In the fourth phase, the athletes land on the ground. The landing point determines the final jumping distance.

Athletes work hard to achieve the perfect body posture at take-off and during the flight phase in every jump. Therefore, some ski jumping hills are lined with cameras along the flight trajectories of the ski jumpers. Coaches evaluate the recorded jumps by selecting frames manually, annotating relevant keypoints by hand and calculating the flight parameters using these hand-annotated keypoints. The system proposed in this chapter fully automates this process. Given the videos from all cameras along the hill that belong to a single jump as input, our model (1) detects keypoints of the athlete as well as ski tips and ski tails in each video frame of each camera if present, (2) executes a robust estimation in each camera view based on the single-frame results and (3) outputs the flight parameters for each camera. The relevant flight parameters for the coaches are shown in Figure 1.1. Based on these parameters and a Principal Components Analysis, it is possible to predict if a jump has a long, medium or short distance.

3.2 Related Work

Human Pose Estimation (HPE) is an active research field in computer vision. For years, most methods have used deep convolutional neural networks (CNNs) for that task and methods like CFA [84] and Res-Steps-Net [6] achieved the best scores on Human Pose Estimation benchmarks like MPII Human Pose [1] and COCO [50]. Regarding the architecture, HPE approaches can be categorized in single-stage [32, 12] and multi-stage [84, 6, 45, 93] methods. The basis of single-stage approaches are mainly networks that perform well on image classification tasks, like ResNet [33] or VGG [81]. Mask R-CNN [32], which we use as a base model, firstly determines regions of interest and then executes single-person pose estimation on these regions. Multi-stage approaches [84, 6, 45, 93] try to refine the pose estimates in every stage.

Computer vision has become quite popular in analyzing athletes of different sport disciplines. Fastovets et al. [23] propose a user-assisted method for estimating and tracking athlete poses from monocular TV sports footage. Their model is evaluated on hurdles and triple jump videos. Zecha et al. [100] use a multi-step architecture to estimate the poses of ski jumpers. With a convolutional sequence to sequence model, they predict the jump forces of ski jumpers directly from the pose estimates. With the usage of a dilated convolutional network, Einfalt et al. [20] automatically detect events like ground contact in pose sequences of triple jump recordings. Wei et al. [94] predict the location of the basketball from a monocular view, even if it's occluded, based on the trajectories of the players. For a performance analysis of swimmers, Einfalt et al. [22] use a convolutional neural network with frame sequences of the swimmer and the swimming style as inputs. The knowledge of the swimming style and the usage of a pose refinement over time improves the results per frame.

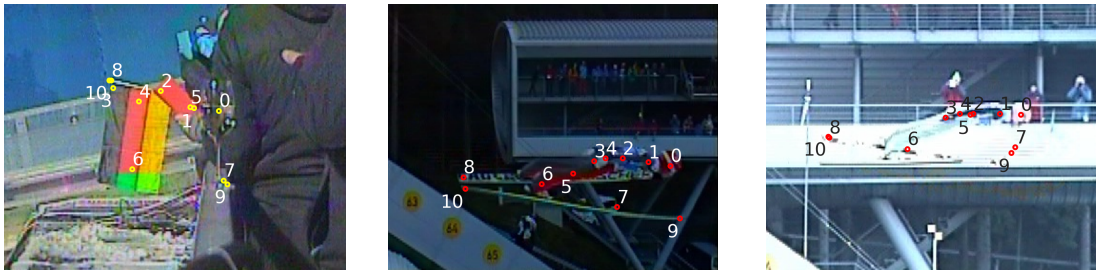
For many computer vision applications, robust estimation is an important step as results are often computed from noisy data with some outliers. A popular strategy for robust estimation is Random Sample Consensus (RANSAC) [27], which uses some samples from the whole data set to compute the model parameters and then calculates how many data points from the whole data set are in conformity with this model. After some iterations, the model with the highest number of conforming data points is chosen. Chum et al. [17] improve this method by adding local optimization after choosing the best model.

In a previous system developed by Zecha et al. [100], the keypoint detection is split into separate steps. At first, the position of the ski jumper is located within the frame using MobileNet [34]. Next, at this location a convolutional pose machine [93] detects the athlete's joints. Third, a Hough transformation is used to identify the skis. For each camera view, the mean pose is calculated afterwards, and the flight parameters are computed based on this mean pose. A careful evaluation by the coaches and performance diagnosticians shows that this multi-step model generates mostly reasonable results regarding the single-frame results, but the usage of a mean pose often impairs the final result due to outliers. This happens especially often for the ski detections, as the Hough transformation produces more than sporadically false results.

3.3 Method

Our goal is to improve the quality and robustness compared to the previous model [100]. Therefore, we have developed a new model that performs all detections in one single step. It is based on Mask R-CNN [32], but uses a branch to detect keypoints instead of generating segmentation masks. We also let the model learn to estimate the locations of the ski tips and ski tails, which is more reliable than the previously used Hough transformation. This is possible since we improved the ski tip and tail annotations. Before, only the central point between left and right ski tip as well as left and right ski tail was annotated. It was not possible to train a model with this semantic, therefore the Hough Transformation was used. Now, the left and right ski tips and tails are annotated separately, which allows the model to learn the ski keypoints, too.

During the annotation process, the flight parameters were derived as the mean of the angles of all annotated poses per camera view, usually 2–4. However, there are many more frames per camera showing the complete athlete. Thus, the new model can use all images from each camera view. On average, it detects a ski jumper in 14 images per camera. However, the estimations contain some errors, which are mostly due to bad image quality. Therefore, we employ a post-processing step involving the pose detections. The first part is a plausibility check to identify gross mistakes. These checks are based on the keypoints of the pose itself: The system checks (1) if the length of both skis is nearly equal, (2) if the length of the body (the distance head to hip plus hip to ankle) is not shorter than half of and not longer than the ski length, (3) if the head is above the ski tips, (4) if the hand is far enough from the ski tips or tails and the head, (5) if the length of the lower leg and thigh are nearly the same, (6) if the size of the upper body



(a) A pose is detected where no athlete is visible.

(b) The length of the skis does not match.

(c) Ankle and knee are at wrong positions.

Figure 3.1: Invalid poses identified by pose checking. The detected keypoints are visualized by red circles and marked with numbers, whereby number 0 marks the head, 1 the shoulder, 2 the elbow, 3 the hand, 4 the hip, 5 the knee, 6 the ankle, 7/9 the right/left ski tip, 8/10 the right/left ski tail.

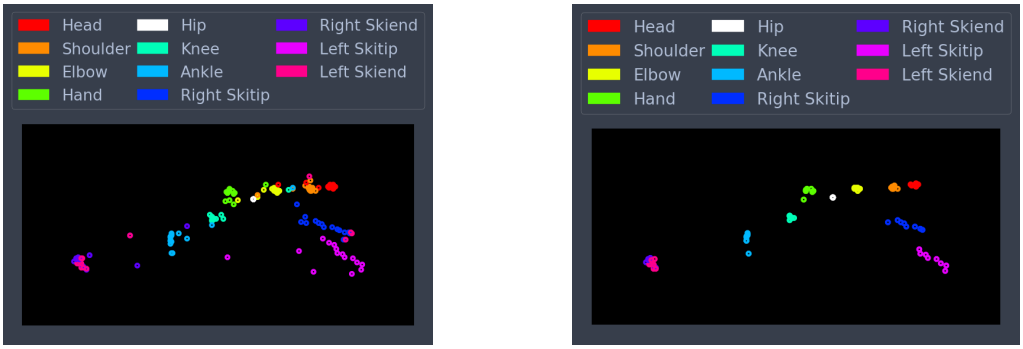
is similar to the size of the lower body, (7) if all keypoints are not too close to the image boundaries and (8) if all joints (except for the ankle) are on one side of the skis. Poses that do not pass these checks are removed. Examples for invalid poses can be found in Figure 3.1. Furthermore, this step sorts out poses where only a part of the athlete is shown in the picture. The model already detects the ski jumper, but the poses are not precise enough to contribute to the final result. Therefore, they are discarded. The pose checking process removes around 42.8% of all poses, so that on average 8 images remain per camera. Figure 3.2 visualizes this effect.

The second post-processing step takes all plausible poses and uses a robust estimation to output the final flight parameters. We use this technique as the pose of the athlete barely changes within one camera view. In the annotation process, the mean is used, but using the mean is too sensitive to detection outliers. Hence, we use locally optimized RANSAC [17] in two variants for that purpose. The first variant calculates the relevant angles from the keypoint locations and applies RANSAC to each set of angles. In a second variant, the poses are normalized by translating the hip of an athlete to the origin of the coordinate system. RANSAC is applied to the normalized keypoints, which results in a robust mean pose. The flight parameters per camera are in turn derived from this mean pose. We include the implementation details for RANSAC usage in Section 3.4.2.

Summarizing, the model consists of three main steps: (1) detecting keypoints of ski jumpers with a fine-tuned HPE version of Mask-RCNN in all frames of one camera view, (2) checking the detected poses and removing the invalid ones and (3) robustly estimating the flight parameters with RANSAC.

3.4 Evaluation

For our evaluations, we use the IAT-Skijump dataset as described in Section 2.2.2 and apply two different evaluation protocols.



(a) All 15 poses of one camera view.

(b) Remaining 8 poses after plausibility check.

Figure 3.2: Effect of pose filtering: On the left side, all poses of one camera view are displayed, centered at the hip joint. The right side shows the remaining poses after filtering.

3.4.1 Image-Wise Results

The first protocol is the PCK as described in Section 2.1.2.1. It evaluates the pose estimation results image-wise on the video images with ground truth annotations. We will refer to this technique as the evaluation on annotated images. We only apply it if mentioned explicitly. The recall curves for varying PCK thresholds are visualized in Figure 3.3 with solid lines. The recall at a PCK threshold t measures the percentage of the keypoints that are considered correct at threshold t.

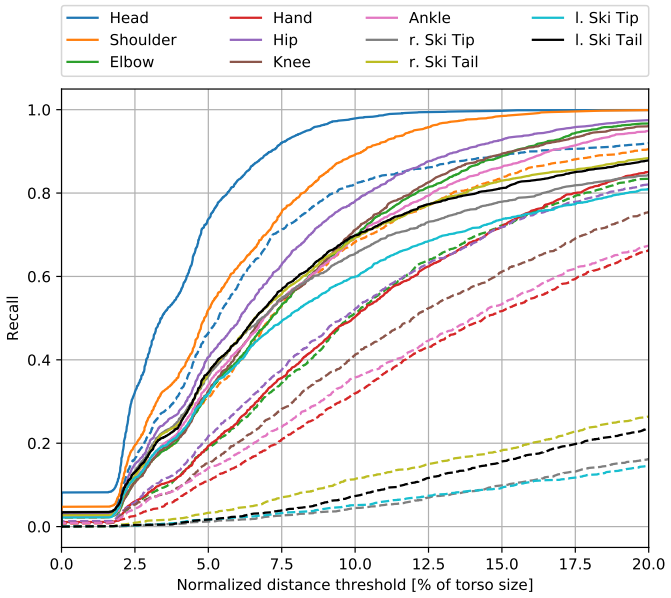


Figure 3.3: Recall curves for varying PCK thresholds on the test set. The results of the proposed model are displayed with solid lines, the results from the previous system [100] with dashed lines.

After pose checking, all detected ski jumpers are valid detections, indicating that our system achieves 100% precision. Hence, we do not use precision as a metric. The main focus is the compliance of the detected keypoints with the ground truth. Therefore, the distance between a detected keypoint and its corresponding ground truth position is the metric of interest.

The second protocol compares the estimated flight parameters per camera view with the mean of the angles from the annotated poses of one camera view. Our evaluation focuses on this protocol, since it complies with the evaluation by coaches. We therefore define the metric Percentage of Correct Angles (*Mean-PCA*). We call it Mean-PCA to distinguish it from the Principal Component Analysis, which is commonly abbreviated with PCA and which we use in another part of the system (see Section 3.4.3). Analogous to PCK, Mean-PCA considers an angle as correct at a threshold t if the difference between the angle computed from the detected joints and the angle calculated from the ground truth joints is below or equal t. The recall at a Mean-PCA threshold t measures the percentage of the angles that are considered correct at threshold t. Table 3.1 shows the recall values at Mean-PCA thresholds of 5° and 3° for all five relevant angles for the ski jump coaches (upper body angle, lower body angle, total body angle, ski angle and difference between lower body and ski angle).

Regarding the previous system [100] with the multi-step pose estimation, the current model achieves greater accuracy for all keypoints. The PCK values for the previous model are visualized in Figure 3.3 with dashed lines. Huge differences are encountered in the detection of the skis. The recall of the previous model at a PCK threshold of 20% is only 14.9% for the left ski tip and 16.4% for the right ski tip (Figure 3.3, dashed lines), while the current model achieves 81.2% and 84.3%, respectively (Figure 3.3, solid lines). The reason for this huge difference is that the Hough transformation used for the ski detection is far less precise. The previous system could not include it in the neural network training as the annotations at that time included only the averaged position of left and right ski tips and tails.

Although the positions of the ski tips and tails are not accurately estimated with the Hough transformation, the results for the derived angles of the skis are fairly good: 88.4% of the detected ski angles are within ±5°. In general, as Table 3.1 shows, the new model improves the recall values at a Mean-PCA threshold of 5° for all five relevant angles at least by 4.0% and at most by 22.9%, and by 13.0% to 30.0% at a Mean-PCA threshold of 3°.

System	t	Lower Body	Upper Body	Total Body	Ski	Diff. L.B./S.	Avg
Zecha et al. [100]	3°	64.0	75.6	84.4	80.9	51.3	71.3
Ours	3°	**84.6**	**92.3**	**97.4**	**97.4**	**81.3**	**90.6**
Zecha et al. [100]	5°	84.3	91.0	95.8	88.4	72.0	86.4
Ours	5°	**97.4**	**99.0**	**99.8**	**98.6**	**94.9**	**98.0**

Table 3.1: Recall values in % at Mean-PCA thresholds of 5° and 3° on annotated test set images.

3.4.2 Robust Flight Parameter Evaluation per Camera View

The first variant of robustly estimating the flight parameters starts with the calculation of the five angles of interest for every pose. As a second step, for each camera view, a constant model is estimated with RANSAC for each angle. We use 100 iterations with a sample size of 4. Hence, for each type of angle, 4 computed angle values are randomly chosen, and the average is calculated. Then, the number of inliers for this model among all calculated angle values is computed. Other angles are defined as inliers if they deviate at most by 4° from the average angle of the samples. For the model with the most inliers, the final result is calculated as the average angle of all inliers. The results for the Mean-PCA metric are shown in Figure 3.4. The results for all flight parameters, except the ski angle, are improved. Ski keypoints are the keypoints with the lowest PCK (see Figure 3.3) and have therefore the highest estimation errors.

For this method which we will refer to as RANSAC on angles, the angles are calculated in advance, meaning before RANSAC is executed. Therefore, some were calculated based on wrong poses, but ended up as inliers, if they fit the threshold. Hence, they might cause the final result to differ too much from the true value. Therefore, we also propose a second variant which avoids this problem.

The second approach translates the poses such that the detected hip joint lies in the origin of the coordinate system. All poses are now relative to the hip. RANSAC is executed on the translated joint coordinates with 100 iterations and 4 or 5 samples. Experiments on the validation set show that using 5 samples for the ski keypoints and 4 samples for the body joints achieves the best results. Hence, for each keypoint, 4 or 5 random samples are chosen and the average keypoint coordinates are calculated. The number of inliers among all values for this keypoint is computed afterwards. We define other keypoints as inliers if their distance to the average point of the samples is

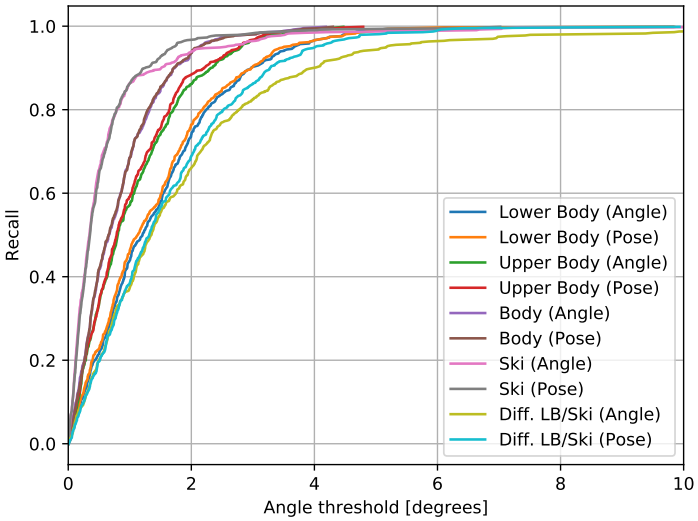


Figure 3.4: Recall curves for varying Mean-PCA thresholds on test set for RANSAC on angles and RANSAC on poses.

System	t	Lower Body	Upper Body	Total Body	Ski	Diff. L.B./S.	Avg
Image-Wise	3°	84.6	92.3	97.4	97.4	81.3	90.6
RANSAC Angles	3°	90.1	96.4	**97.9**	96.3	82.3	92.8
RANSAC Poses	3°	**90.3**	**96.7**	**97.9**	**98.2**	**86.1**	**94.0**
Image-Wise	5°	97.4	99.0	99.8	98.6	94.9	98.0
RANSAC Angles	5°	**98.9**	**100**	**100**	98.6	94.3	98.4
RANSAC Poses	5°	**98.9**	**100**	**100**	**99.2**	**97.9**	**99.3**

Table 3.2: Recall values in % at Mean-PCA thresholds of 3° and 5°: Results on annotated images, results with RANSAC on angles, and results with RANSAC on poses.

at most 35% of the currently estimated torso size (distance shoulder to hip keypoint). The result of this computation is a robustly estimated mean pose for each camera view. The angles of the flight parameters are now calculated by using the keypoints of the mean pose. Figure 3.4 shows the Mean-PCA results for this approach. RANSAC based on poses improves the results for all flight parameters (see Table 3.2). The recall at the Mean-PCA threshold of 5° is above 97% for all angles.

Both RANSAC methods achieve good results and improve the Mean-PCA values on the annotated images, except RANSAC on angles for the ski angle. Table 3.2 displays the recall values at Mean-PCA thresholds of 3° and 5°. RANSAC based on poses also improves the results for ski angles. The reason is that the calculation of an average pose is more precise for the skis, as only keypoints that are close together are included in the angle calculation. Table 3.2 shows that RANSAC on poses generates the best results for all angles and RANSAC on angles achieves the same result for the total body angle.

3.4.3 Jumping Distance Prediction Based on Flight Angles

The dataset provides 84 videos with the information of the jumping distance. All videos are from the same ski jumping hill and recorded with the same camera settings. We use a Principal Component Analysis (PCA) to map the five estimated flight parameters to their two principal components. Figure 3.5 depicts the flight parameters for two camera

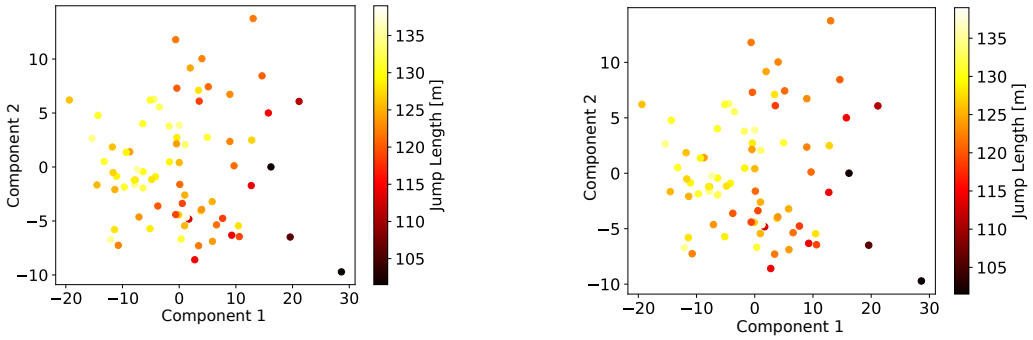


Figure 3.5: Results of Principal Component Analysis based on all flight parameters for two camera views.

views in this two-dimensional subspace with respect to the color-coded jumping distance. An exact prediction of the jumping distance is not feasible, as important information like the wind speed and direction is not available, but it is possible to see clusters of jumps with high distances in both visualizations (colored yellow in the left figure and yellow and orange in the right figure). Hence, if a new jump is recorded and the flight parameters extracted, it is possible to perform the same Principal Component Analysis as before and predict if the jump has a long, medium or short distance.

3.5 Summary

This chapter has introduced a technique for robustly estimating flight parameters for ski jumpers. It uses a Mask R-CNN [32] based model to detect the joints of the athlete and the ski tips and tails. As the coaches are interested in the flight parameters per camera, we could use a robust estimation based on all detections of one camera view to determine the values. This robust estimation includes a pose checker that removes wrong poses. It executes checks based on the distances and relative positions of the keypoints, e.g., it compares the body and the ski length or the length of upper and lower body. In a second step, RANSAC calculates the robust estimation of the poses considered valid by the pose checker. We examined two versions of RANSAC. The first operates already on calculated angles based on single poses, the second operates on single poses and calculates the final flight parameters based on the estimated mean pose.

Our evaluations have shown that our model performs notably better than the previous one by Zecha et al. [100] which used a multi-step pipeline to detect the body joints and a Hough transformation for the skis. Especially for the results of the ski angle and the difference between lower body and ski angle, the proposed model improves the recall at a Mean-PCA threshold of 5° by absolute +10% on the test set images. The percentage of correct angles could be improved even further with the usage of the robust estimation. The recall at a Mean-PCA threshold of 5° is over 97% for all flight parameters, using the best performing version of RANSAC, and over 86% at a Mean-PCA threshold of 3°. A PCA executed on the flight parameters is able to roughly predict the jumping distance and categorize a jump as long, medium or short.

4 Semi-Supervised Learning for Human Pose Estimation in Sports

As already described in the last chapter, financial constraints often limit access to high-quality technical equipment and sophisticated analysis tools, especially in niche sports. For the ski jumpers in Chapter 3, we could leverage the vast amount of hand annotated data that they produced due to their manual visual analysis. However, the hand annotation process is usually very costly. Hence, similar approaches to the one in Chapter 3 are not feasible for most sports disciplines. Therefore, we investigate semi-supervised learning approaches to reduce the amount of hand-annotated data needed to train an HPE model for a new sports domain. The less labeled data is needed, the more accessible the technology becomes for a wide range of sports disciplines. This chapter is mainly based on the following publication, with some text passages directly taken from it:

Self-Supervised Learning for Human Pose Estimation in Sports [58], Katja Ludwig, Sebastian Scherer, Moritz Einfalt and Rainer Lienhart, IEEE International Conference on Multimedia and Expo (ICME) Workshops 2021, Shenzhen, China, July 2021.

In this work, we propose two methods to fine-tune a 2D HPE system trained on general poses to a sports discipline specific HPE model using only a few labeled images. We show that 50 labeled poses and additional unlabeled videos are sufficient to achieve a Percentage of Correct Keypoints (PCK) of 88.6% at a threshold of 0.1 in the disciplines of triple and long jump, closing the gap between the supervised fine-tuning on the same 50 images and the fully supervised training on 60× more images by 60%. The first proposed method uses pseudo-labels as a semi-supervised training technique together with a filtering method of the pseudo-labels in an iterative manner. Furthermore, we show that a mean teacher approach, which is based on consistency between a teacher and a student model, can also improve the results to a large extent, although it performs slightly worse than the iterative pseudo-label approach.

4.1 Introduction

In many sports disciplines, Human Pose Estimation (HPE) is an important method for performance analysis and improvement of athletes. As described in the last chapter, ski jumpers use keypoint detection on their body and their skis to derive flight parameters such as ski angle, lower body angle and upper body angle to perfect their body posture during the flight phase and achieve long flight distances. This chapter uses the domain

of triple and long jump as an example sports domain, but the approach is also easily adaptable to any other discipline. An example for estimated poses during a triple jump is shown in Figure 4.1.

Annotating a large amount of 2D poses in many images by hand is enormously time-consuming, especially when lots of keypoints are necessary. In many sports disciplines, coaches do not have enough time to select images from training or competition videos and annotate them for the purpose of measuring the athletes' performance. Hence, HPE systems based on neural networks generate a large benefit as they automatically detect the needed keypoints in a fraction of time. However, to train a deep neural network capable of achieving an acceptable performance on images from the desired sports domain, lots of annotations are needed. Existing datasets mainly consist of images from everyday activities which do not cover the diversity of poses in most sports. Therefore, new annotations are needed in most cases. This chapter introduces two techniques that need only a few annotated images and some videos from the sports domain to train a neural network with superior performance based on semi-supervised training.

Figure 4.1: Human Pose Estimation for triple jump analysis.

4.2 Related Work

Human Pose Estimation is a field of active interest in computer vision. At the time of the publication of this work, the best scoring approaches on common benchmarks like COCO [50] or MPII Human Pose [1] are based on convolutional neural networks [35, 4]. The underlying backbone used in many models like [35, 4] is the *High Resolution Net* (HRNet) [92] and its variants for Human Pose Estimation, e.g. HigherHRNet [13]. This kind of backbone, which we also use in our experiments, maintains several branches of different resolutions, keeping the highest resolution from the beginning to the end of the network and providing data exchange between the different resolutions. Although Transformer [89] based networks are becoming the new state of the art in the last years, HRNet is still a very popular choice for HPE tasks, especially in bottom-up approaches. Moreover, Transformer networks tend to need more data for training than CNN based approaches.

Semi-supervised learning is a highly attractive research field in computer vision. Its goal is to enhance neural networks by exploiting additional unlabeled data for training such that the resulting model performs better than a fully supervised training using labeled data only. Most common approaches use consistency regularization or pseudo-labeling. A survey can be found in [88]. Consistency regularized methods are based on the idea that models should generate consistent predictions under different perturbations such as noise [87] or stronger augmentations [97]. In the domain of 2D HPE, Xie et al. [97] propose cutting out joints to simulate occlusion as a hard augmentation. Tarvainen et al. [87] further use a model ensemble as a target, because a model ensemble produces better predictions compared to a single model. Other approaches utilize pseudo-labels, which means that network predictions are used as annotations [40]. Xie et al. [96] prove that this method is effective for the ImageNet classification task. Furthermore, Srivastav et al. [83] show that the usage of pseudo-labels enhances their joint 2D/3D HPE pipeline for multi-person keypoint detection in operating rooms.

Our contributions can be summarized as follows. We show that semi-supervised learning raises the detection performance of 2D HPE systems fine-tuned for specific sports disciplines with a small labeled training dataset. Two different methods are proposed, whereby the first leverages pseudo-labels in an iterative process to increase network performance. We introduce a pseudo-label selection method that selects the most accurate predictions across various augmentations as pseudo-labels. Furthermore, we introduce mean teacher training for HPE, a single-step consistency based approach. We show that 50 labeled training images and 122 unlabeled videos are sufficient to generate superior detection results in the domain of long and triple jump.

4.3 Method

We use *HigherHRNet* [13] as a backbone network, which achieves good performance on HPE benchmarks. As the goal of this chapter is to achieve superior performance in keypoint detection of athletes from a given sports discipline with only a few labeled images from that sports domain, we start with pretrained weights from COCO [50]. As the keypoint definition of long and triple jump athletes is different from the COCO keypoint definition (details can be found in Section 2.2.3), we use only the backbone weights and not those from the network head.

4.3.1 Training with Pseudo-Labels

In this section, we introduce the first semi-supervised approach which leverages pseudo-labels in an iterative manner. In Section 4.3.2, we explain the second semi-supervised approach based on mean teacher training.

4.3.1.1 Training Procedure

At the beginning, a fully supervised training on a small, in-domain labeled dataset $D_{labeled}$ is executed. Early stopping is used to select the weights of the epoch with the

best score on the validation set. The next step is the generation of *pseudo-labels*, which means that the labels are not annotations by hand but created from the network itself. Based on the selected weights, pseudo-labels are generated for all unlabeled images of the training set, resulting in the pseudo-label dataset $D_{PL,1}$ for the first iteration. Details on the generation process are described in the next section. Afterwards, the first semi-supervised training iteration is started by training a newly initialized network on the generated pseudo-label dataset $D_{PL,1}$, starting from pretrained weights from COCO. Hence, the network sees only images different from the labeled training set $D_{labeled}$. Again, the best weights according to the validation score are selected. In the next step, fine-tuning based on the selected weights is executed with $D_{labeled}$. The best weights according to the validation results are determined and used to generate updated pseudo-labels $D_{PL,2}$. Then, the next semi-supervised training iterations are executed analogous to the first one. Figure 4.2 visualizes the training procedure. Xie et al. [96] show that this iterative process performs well on ImageNet image classification. We adapt the approach to Human Pose Estimation.

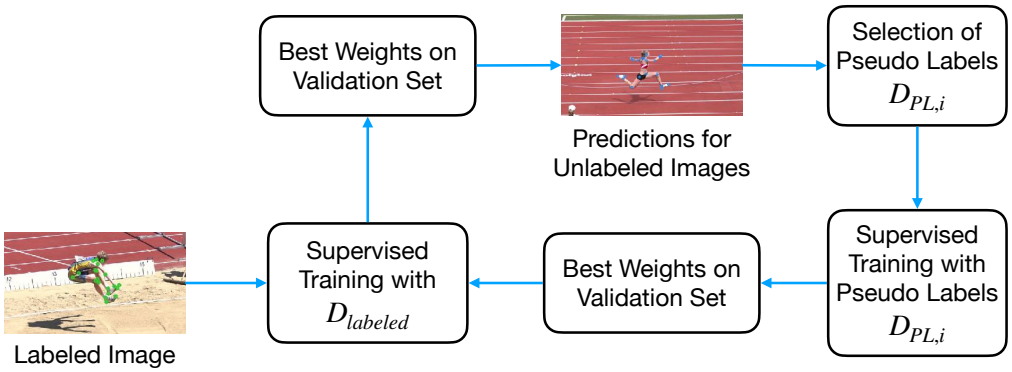


Figure 4.2: Training procedure for pseudo-label training.

4.3.1.2 Pseudo-Label Selection

In order to improve the network performance with pseudo-labels, the selection of the pseudo-labels is crucial. The most obvious possibility is to choose the labels based on the network confidence score, keeping all labels with a score larger than a certain threshold. In HPE, the confidence of the model is the maximum activation in the output heatmap. The problem of this method is the non-equal distribution of the pseudo-labels across the joints. Typically, certain keypoints are learned easier by the model, resulting in larger confidence scores compared to more challenging keypoints. Hence, the detection performance of joints with a lot of pseudo-labels increases, while the scores of joints with fewer labels stagnate. A solution to this imbalance problem is to take the labels with the best $p\%$ confidence scores of each joint, but we found that this method does not output the best labels based on the distance to the ground truth as there is no direct relation between the network confidence score and the distance between the predicted and the ground truth keypoint.

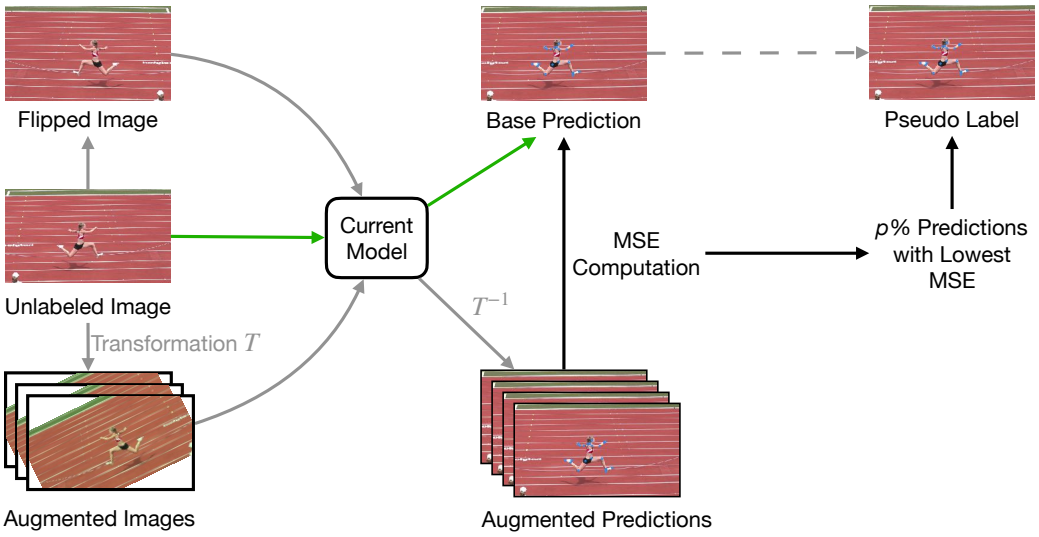


Figure 4.3: Pseudo-label selection method. Green arrows indicate that the base prediction is the prediction of the unlabeled image. The transformation T is randomly selected from the augmentations described in Section 4.4.1.

As a consequence, we use a different approach to select the best labels, not relying entirely on the network confidence scores. As a base prediction, we use the predictions generated with the raw, not augmented input image. We add prediction results of a horizontally flipped image and results from some randomly chosen augmentations (described in Section 4.4.1) to our prediction set. Predictions with a very low confidence score (below threshold t_{score}) are discarded. The mean squared error (MSE) in pixels between the base prediction and the augmented predictions is calculated. If the predictions are reliable, the locations of the predicted keypoints in the base image and the augmented images should not differ, leading to a small MSE value. Next, we select the predictions with the lowest $p\%$ MSE for the pseudo-label dataset per keypoint, resulting in an equal number of predictions for each keypoint. As pseudo-labels, the base predictions are used instead of the mean over all predictions, as single outliers could shift the mean enormously and the predictions on augmented images are less accurate as they are more difficult for the model. Figure 4.3 illustrates the pseudo-label selection steps.

4.3.2 Mean Teacher Training

Apart from using pseudo-labels, we consider another semi-supervised training method based on consistency between a teacher and a student model. Figure 4.4 visualizes the training steps. At first, the student and the teacher backbone are initialized with pretrained weights from COCO, while all other weights are initialized randomly. Both models are nearly identical with the single difference that we add dropout in all fusion layers of the HRNet backbone (see [92]) in the student model. During a warm-up period, only the supervised training steps are executed based on $D_{labeled}$, visualized in green in

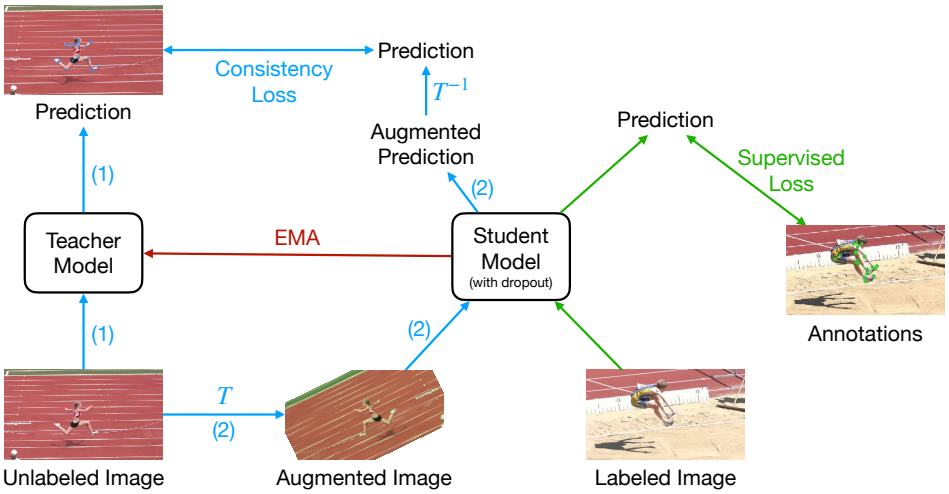


Figure 4.4: Mean teacher training. The supervised training part is visualized with green arrows, the consistency regularization part is illustrated in blue. The teacher weights are updated after each training step, symbolized by the red arrow.

Figure 4.4. After the warm-up period, the consistency loss is taken into account as well, which is explained in the following. Unlabeled images are fed into the teacher network to generate predictions, this process is marked with *(1)* in Figure 4.4. The same images are now augmented (see Section 4.4.1 for details) with a transformation T and given to the student network as an input (marked with *(2)*). The student predictions are transformed back to their corresponding position in the original image with the inverse transformation T^{-1}. The consistency loss is now calculated as the MSE of the predicted poses from the student and the teacher model. The loss is masked if the teacher confidence is too low (below threshold t_{score}). Throughout the complete training process, the teacher weights are updated after each training step to be the exponential moving average (EMA) weights of the student model, visualized in red in Figure 4.4. A training step consists of both supervised and consistency loss updates. For inference, the student model is used. Tarvainen et al. [87] show that this method leads to great improvements on common image classification benchmarks. We adapt this approach for the usage in HPE.

4.4 Evaluation

We evaluate our experiments on the BiSp-Jump dataset introduced in Section 2.2.3, which consists of images from triple and long jump athletes during training and competition scenarios.

4.4.1 Data Augmentation and Training Settings

During supervised training, fine-tuning and pseudo-label training, we use common augmentation methods: random horizontal flip, random rotation of up to 30°, random trans-

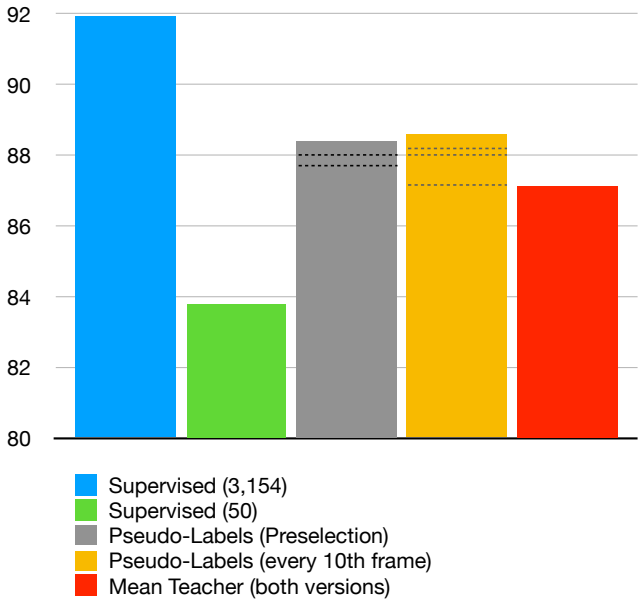


Figure 4.5: PCK@0.1 results for fully supervised and semi-supervised trainings on the test set. We display the results for different iterations (if applicable) with dashed lines in the bars. The results for the supervised training with all labels and with 50 labels are displayed in blue and green, respectively. We execute the experiments with pseudo-labels and with mean teacher either on every 10th frame of the videos, or on the frames that are labeled (but without using the label). We call the latter *preselection*, since the frames to be labeled are selected by the annotator. For mean teacher, both versions achieve the same result, therefore only one bar is visible.

lation of up to 40 pixels (using an input image size of 512 × 512), random scale in the range $[0.75, 1.5]$, and color jitter. These augmentations are also used during pseudo-label generation.

For our experiments, we use 50 randomly selected ground truth labels for our training dataset $D_{labeled}$. During pseudo-label creation, we set the score threshold $t_{score} = 0.1$, and we evaluate 10 different augmentations to compute the MSE. We train until convergence and start with best 60% pseudo-labels in the first iteration. As the quality of the pseudo-labels improves in each iteration, we increase the number of pseudo-labels that we use by 10–20%. In each iteration, we train for $10k$ steps with the pseudo-labels and execute the fine-tuning with the labeled images for 250 steps. For mean teacher training, we use the same score threshold t_{score} of 0.1 to mask the consistency loss. The EMA momentum is set to 0.999 and our warm-up period lasts for 1,500 steps. The dropout rate of the student model is set to 0.2.

4.4.2 Results

For evaluation, we use the Percentage of Correct Keypoints (PCK) metric as introduced in Section 2.1.2.1. During training, we use this metric at a threshold of 0.1 as the

performance measure on the validation set, which corresponds to a deviation of approx. 6 cm in this dataset. A fully supervised training with all available labels achieves a PCK@0.1 of 91.9% on the test set. The supervised training on 50 labels reaches a PCK@0.1 of 83.8%. We try to close the gap in between those results. Both boundaries are visualized in Figure 4.5.

4.4.2.1 Pseudo-Label Results

For the first experiment, we use the subset $D_{labeled}$ with 50 labeled images for training and the remaining 3,104 images that are also annotated as unlabeled images, but we do not use the labels. This has the benefit that the results are perfectly comparable to the fully supervised results as both networks have seen the exact same images during training. For comparability to the fully supervised training, we use the full validation set of 200 images in these experiments, but we verified that a validation set of 50 images leads to similar results, which ensures that a small amount of annotated images is also sufficient as the validation set. In practice, this setup requires the coaches, apart from annotating the images for $D_{labeled}$ and the validation set, to select meaningful images from the videos that should be used for training. In our first setup, we use images that were selected by coaches for annotation, but we discard the annotations. However, selecting an image (e.g. for the annotation process) is already some kind of information. An image selection process requires a lot less time than annotating all images, but more

	Images (Labels)	Run	PCK@0.1	PCK@0.2
1	3,154 (3,154)	supervised	91.9	96.5
2	50 (50)	supervised	83.8	90.0
3	3,154 (50)	iteration 1	87.7	93.6
		iteration 2	88.0	93.9
		iteration 3	88.4	94.4
4	17,656(50)	iteration 1	87.1	93.5
		iteration 2	88.0	94.2
		iteration 3	88.2	94.5
		iteration 4	88.6	94.7
5	3,154 (50)	MT	87.1	93.7
6	17,656 (50)	MT	87.1	93.5

Table 4.1: Recall values in % at PCK thresholds of 0.1 and 0.2 on annotated test set images. The first row shows the results for the fully supervised training and the second row the results for the supervised training on $D_{labeled}$ for comparison. Row 3 and 4 display the results for the pseudo-label semi-supervised training based on preselected images and on every 10th video frame, respectively. Row 5 and 6 display the mean teacher (MT) results for both variants.

Figure 4.6: Example images with predictions from the model trained with pseudo-labels. It can deal with occlusions, extreme poses, and new keypoints like toe tip or heel, although such cases are not included in COCO.

time than just providing videos without any frame selection. Therefore, we evaluate two setups: The first consists of the preselected images, meaning the images that were selected by the coaches for annotation, but without using the annotations. The second setup uses every 10th frame of the videos without preselection.

Table 4.1 shows the results for three iterations in the preselected setup in row 3. Figure 4.5 visualizes the results for all iterations in gray, using a dashed line to indicate the scores for the first two iterations. Moreover, Figure 4.6 shows some examples for model predictions after convergence.

We increase the percentage of pseudo-labels that we use in every iteration, since the model is improving from iteration to iteration. We start with 60% of the pseudo-labels in the first iteration and increase the percentage to 70% and 90% in iteration two and three, respectively. After three iterations, this training procedure converges.

The gap between the fully supervised training and the supervised training on 50 images is 8.1% at a PCK@0.1. After the first iteration, this gap can be narrowed to 4.2%. After the third iteration, the difference shrinks to even 3.5%. This is (relative) 40% of the original gap, with using less than 1.6% of the original labels. As we use the validation results at threshold 0.1, this threshold is also used as the main evaluation threshold. But regarding PCK threshold 0.2, the gap is being narrowed even further, from 6.5% to 2.1%, which is less than a third.

In the second experiment, we take the same 50 labels as in the first experiment, but no preselection of video frames is used. From all 122 videos belonging to the training dataset, every 10th frame is extracted and added to the unlabeled dataset. This results

in 17,656 images. We use every 10th frame such that two images are clearly different from each other. This strategy does not require additional work of the coaches despite recording some videos and annotating 50 frames. Table 4.1 shows the results for this experiment in row 4. Figure 4.5 visualizes the results for all iterations in orange. We use an additional fourth iteration with all pseudo-labels here, as further training still improves the results on the validation set, which is not the case in the first experiment. The results after the last iteration of this experiment are similar to the results from the first experiment, but this experiment has slightly better performance at PCK thresholds 0.1 and 0.2. At PCK threshold 0.1, we could close the gap to the fully supervised training from 8.1% to 3.3% and at threshold 0.2 from 6.5 % to 1.8%. The table shows that the improvement from iteration to iteration is slower than in the first experiment, therefore the fourth iteration still gains some improvement.

4.4.2.2 Mean Teacher Results

Identical to the pseudo-label evaluation, we conduct two mean teacher experiments. One with preselected images and one with every 10th frame from the videos corresponding to the training set. Table 4.1 shows the results for both experiments in row 5. Figure 4.5 visualizes the results in red. Both experiments achieve the exactly same score at PCK@0.1, therefore only one bar is displayed in the diagram. The results are collected on all annotated test set images using the network weights from the step with the highest validation score (early stopping). The table shows that the mean teacher results are slightly worse than the pseudo-label results after the final iterations, but perform a lot better than the supervised training on 50 images. At a PCK threshold of 0.1, the gap to the fully supervised score could be narrowed from 8.1% to 4.8% and at PCK threshold 0.2 from 6.5% to 2.8%, regarding the results from the first experiment.

4.4.2.3 Results with more Labels

To evaluate the benefit of more labeled images, we execute the semi-supervised methods also with 100 and 250 labels. See Table 4.2 for the exact results with using the 3,154 images from the supervised training as the training dataset. For 250 labeled images we change the percentages of the pseudo-labels to 70%, 80% and 95%. Otherwise, the first iteration does not have an effect as the PCK values are already higher after the first supervised training. The table shows that the gap between the fully supervised result and the supervised training shrinks from 8.1% with 50 labels to 5.9% with 100 labels and 3.5% with 250 labels. With 50 labels, we could close (relative) 60% of the gap, with 100 labels, this rate shrinks to 50% and with 250 labels to 45%. Furthermore, we analyze the performance differences after the initial supervised training on the few labels. The PCK@0.1 achieved with 100 labels is 2.2% higher than with 50 labels, and it is 2.4% higher with 250 labels than with 100 labels. After the pseudo-label training, the differences are a lot smaller, namely 0.6% between 50 and 100 labels and 1.0% between 100 and 250 labels.

Labeled Data	3,154	250	100	50
supervised	91.9	88.4	86.0	83.8
iteration 1		89.5	87.7	87.7
iteration 2		89.8	88.3	88.0
iteration 3		90.0	89.0	88.4
mean teacher		89.8	88.0	87.1

Table 4.2: Recall values in % at PCK threshold 0.1 on annotated test set images for pseudo-label and mean teacher training with different numbers of labeled images (first row). The second row contains the results for the fully supervised training and the supervised training on $D_{labeled}$ for comparison.

For mean teacher training, similar results are observable. Hence, the gain is larger for fewer labels and the PCK values are closer together after both semi-supervised trainings. For all experiments, single-step mean teacher results are in the area of the results from the first or second pseudo-label iteration. Hence, the usage of semi-supervised training is more effective with less annotated images, but it improves the results in every case. Obviously, the highest absolute score is achieved with the most labels, so there is a trade-off between efficiency of the semi-supervised learning and the final absolute score. All these findings should be taken into account when coaches decide how many images they annotate per hand. With the proposed approaches, it is now possible to train effective HPE models with very low effort needed from the coaches.

4.5 Summary

This chapter has introduced two techniques for semi-supervised learning with a few labeled images in order to train a network for Human Pose Estimation in a new sports domain. One method uses a mean teacher approach like in Tarvainen et al. [87], with a simultaneous training on the labeled and the unlabeled images. The semi-supervised training part uses a consistency loss between an EMA teacher model and a student model with dropout layers and stronger augmentation. The other method generates pseudo-labels similar to Xie et al. [96] and uses a selected subset of them for the first training step and the labeled images for the fine-tuning step. This iterative process is continued until convergence.

The evaluation results prove the sufficiency of a training dataset containing 50 labeled images and some video sequences to train a deep neural network for a new sports domain such that it generates acceptable results. The PCK values at a threshold of 0.1 could be raised from 83.8% to 88.4%, which closes the gap between the fully supervised training on 60× more images and the supervised training on 50 images by more than 60%. These methods could open the usage of Human Pose Estimation performance measurements to a wide range of sports disciplines in the future. The expense to collect video material is very low, as it only requires a smartphone or small camera. Furthermore, annotating 50 images by hand is also done quickly.

Part II

Towards Estimating Any Possible 2D Keypoint on the Human Body

5 Foundations

In the first part of this thesis, we addressed solutions for 2D Human Pose Estimation in scenarios involving low-quality data or limited labeled images. In these cases, the 2D pose is represented as a stick-figure skeleton model, consisting of a small set of fixed joints. While this simplified representation allows for various types of analyses, it significantly reduces the complexity of the human body. Key details, such as body shape, contours, and especially the outline of muscles and limbs, are lost in the process. Especially in the field of sports, such information is important for more in-depth analyses.

In this part of the thesis, we aim to detect more keypoints than the typical stick-figure joints commonly used in 2D Human Pose Estimation (HPE) datasets. We will refer to the keypoints annotated in these datasets as *standard keypoints* throughout this thesis. Our goal is to move beyond the fixed number of keypoints typically detected by models, a task that presents significant challenges. Most 2D HPE models are designed to output a predefined set of keypoints, either encoded in heatmaps or directly as coordinates. This leads to a fixed output structure where each element corresponds to a specific keypoint. However, creating a model capable of detecting any keypoint on the human body surface requires a different approach, as generating an infinite number of heatmaps or coordinates is impractical.

To tackle this, we will leverage the recent Transformer architecture [89] and its adaptation for vision tasks, the Vision Transformer (ViT) [18]. As a first step, we will focus on detecting *intermediate keypoints* — points that lie on a straight line between two standard keypoints. We further analyze the capabilities and limitations of a Transformer-based HPE approach. In the second step, we will extend the model to detect arbitrary keypoints across the entire human surface, including the body outline.

In this chapter, we introduce the tasks of intermediate and arbitrary keypoint detection. For our models, we need segmentation masks of body parts during training, therefore we further specify the task of body part segmentation. Next, we describe an additional metric that evaluates the performance of arbitrary keypoint detection. Moreover, we introduce the datasets that we use throughout this part of the thesis.

5.1 Task Definitions

In Section 2.1, we introduced the task of 2D HPE and provided a formal definition. In this section, we extend this definition to the tasks of intermediate and arbitrary keypoint detection. Recall that a standard 2D HPE model $M_{\text{2D-HPE}}$ operates on single Images I and outputs k previously defined fixed keypoints $p^{2D} \in \mathbb{R}^{k\times 2}$. This set of defined keypoints, which is defined by the used dataset, is called *standard keypoints* throughout this thesis.

5.1.1 Intermediate Keypoint Detection

We define *intermediate keypoints* as any point that lies on a straight line between two standard keypoints that belongs to the skeleton of the dataset. Such straight lines between standard keypoints that match the skeleton will be called *skeleton lines* $l_s$in the following. Let $S \subset \{1, ..., k\}^2$ be the set of all pairs of indices from the standard keypoints that define the skeleton. Any intermediate keypoint can now be defined as $\alpha \cdot p_i + (1 - \alpha)p_j, (i, j) \in S, \alpha \in [0, 1]$. The task of intermediate keypoint detection is to predict the coordinates of these points. Since a model can not have an infinite amount of outputs, we need to tell the model which intermediate keypoint we want it to detect. Therefore, our model has additional inputs, the value for α and the indices (i, j). Hence, a model M_{inter} for intermediate keypoint detection can be defined as

$$M_{inter}(I, \alpha, i, j) = p_{inter} \in \mathbb{R}^2 \tag{5.1}$$

with p_{inter} being the predicted coordinates of the intermediate keypoint. In our case, the model we present in Chapter 6 is able to deal with multiple pairs of indices (i, j) and values for α at the same time, hence it also predicts multiple intermediate keypoints at once.

5.1.2 Body Part Segmentation

Since we will need the results of body part segmentation for the next task, arbitrary keypoint detection, we shortly define this task first. Body part segmentation is similar to semantic segmentation. It means that in an image of a single human person, every pixel that belongs to the human is assigned to a specific body part class. Hence, a body part segmentation mask for an image I of size $h \times w$ with c body part classes is a 2D array of the same size with values in $\{0, ..., c\}$, where the class number 0 represents the background and each value > 0 represents a body part class. We will use body part segmentation masks to train our models for arbitrary keypoint detection, which we will explain in more detail in Section 5.1.3. The body part classes head, torso, left and right upper arm, left and right forearm, left and right thigh, left and right lower leg, left and right foot, and left and right hand are used throughout this thesis. We will also deal

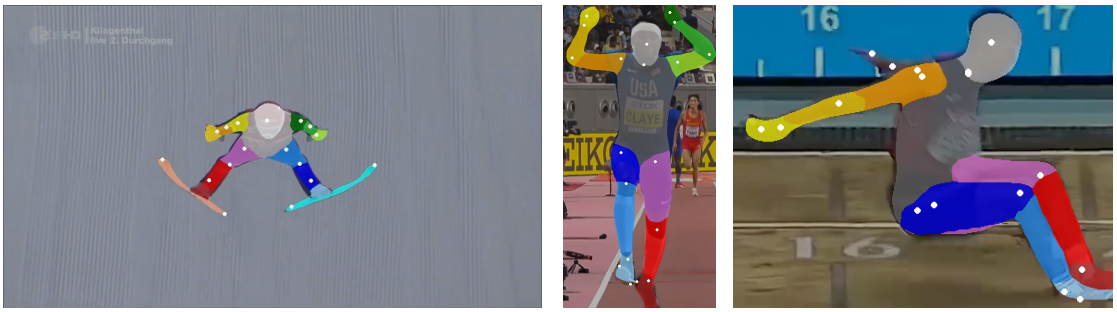


Figure 5.1: Examples of a body part segmentation masks of a ski jumper (left) and triple jump athletes (middle and right).

with ski jumpers in the following, where we have additional classes for the left and right ski. Examples for a body part segmentation masks of a ski jumper and triple and long jump athletes are shown in Figure 5.1.

5.1.3 Arbitrary Keypoint Detection

Arbitrary keypoint detection is similar to intermediate keypoint detection, but adds further complexity. In this thesis, we define an arbitrary keypoint dependent on the segmentation mask that corresponds to the body part that the keypoint is located in. For a chosen arbitrary keypoint p_{arb} lying on the body part which encloses the standard keypoints p_i and p_j, draw a line orthogonal to the line between p_i and p_j through p_{arb}. Figure 5.2 shows an example of that process and visualizes this line in blue color. Now, we call the intersection points of this line with the border of the segmentation mask belonging to this body part c_l and c_r (for left and right intersection point), and the intermediate keypoint that lies on this line p_{inter}. The arbitrary keypoint is now defined as $\beta \cdot c_{l/r} + (1 - \beta) \cdot p_{inter}$, depending if we want the arbitrary point on the left or right side of the skeleton line l_s (with respect to the image). We refer to the value β as the *thickness* of the arbitrary keypoint. A model M_{arb} for arbitrary keypoint detection can be defined as

$$M_{arb}(I, i, j, \alpha, \beta, l/r) = p_{arb} \in \mathbb{R}^2 \tag{5.2}$$

with p_{arb} being the predicted coordinates of the arbitrary keypoint. The model we present in Section 7 is able to deal with multiple requests for arbitrary keypoints at once and outputs the corresponding coordinates simultaneously.

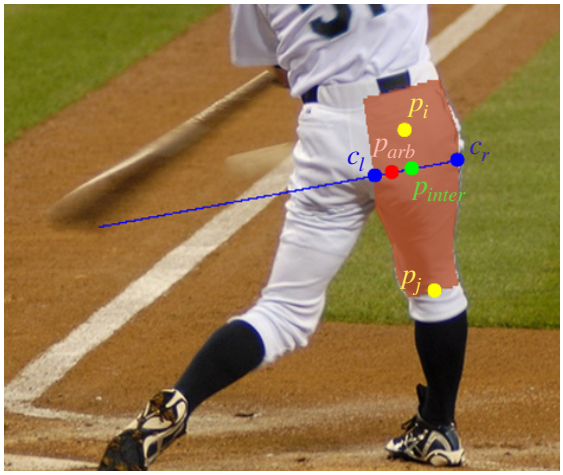


Figure 5.2: Example of a definition of an arbitrary keypoint (visualized in red) and the involved other secondary points. The standard keypoints p_i and p_j are visualized in yellow and the orthogonal line to the line through these points in blue. The intersection points with the segmentation mask c_l and c_r are shown in blue, and the intermediate keypoint p_{inter} in green.

5.2 Evaluation Metrics for Arbitrary Keypoint Detection

As the thickness especially of the limbs of a human is relatively small, the distance between arbitrary keypoints and corresponding intermediate keypoints is also relatively small. Regarding classical 2D HPE metrics like PCK and OKS (see Section 2.1.2), this leads to a high performance for models that do not predict the arbitrary keypoints p_{arb} correctly, but, e.g, the corresponding intermediate points p_{inter}. Although the model does not learn the semantic of the arbitrary keypoint, these metrics are not able to reveal this problem in their scores. Therefore, we propose to use a new metric considering the thickness to measure the success of identifying arbitrary keypoints correctly.

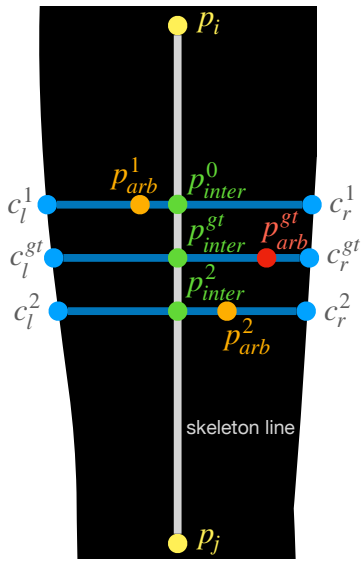


Figure 5.3: Semantic visualization of the calculation of the thickness error for two possible model predictions. The ground truth is displayed in red, the two predictions in orange. Prediction p^1_{arb} is placed on the opposite side of the gray skeleton line with respect to the ground truth point p^{gt}_{arb}, prediction p^2_{arb} is located on the same side.

Let p^{gt}_{arb} be the desired ground truth keypoint, p^{gt}_{inter} the corresponding intermediate keypoint, c^{gt}_l the intersection point on the other side of p^{gt}_{inter} and c^{gt}_r the intersection point on the same side, w.l.o.g., as visualized in Figure 5.3. The ground truth thickness t_0 is computed as

$$t_0 = \frac{||p^{gt}_{arb} - p^{gt}_{inter}||_2}{||c^{gt}_r - p^{gt}_{inter}||_2} \tag{5.3}$$

Assume the model predicts a point p^2_{arb} on the same side of the skeleton line as the ground truth point. Let p^2_{inter} be the intermediate keypoint corresponding to p^2_{arb} and c^2_l, c^2_r be the intersection points in the same way as before. Then, the predicted thickness t_2 is calculated as

$$t_2 = \frac{||p^2_{arb} - p^2_{inter}||_2}{||c^2_r - p^2_{inter}||_2} \tag{5.4}$$

The thickness error e_2 is now calculated as the absolute difference between the ground truth thickness and the predicted thickness:

$$e_2 = |t_0 - t_2|. \tag{5.5}$$

Furthermore, the model might predict a point p^1_{arb} on the opposite side of the skeleton line from the ground truth point. With p^1_{inter} being the intermediate keypoint corresponding to p^1_{arb} and c^1_l, c^1_r be the intersection points on the opposite and same side, respectively, the thickness error e_1 is calculated as

$$e_1 = \frac{||p^1_{arb} - p^1_{inter}||_2}{||c^1_l - p^1_{inter}||_2} + t_0 \tag{5.6}$$

Finally, if an intermediate keypoint p_{inter} can not be defined for a predicted point p^3_{arb}, e.g., because it does not lie in the body part segmentation, we set the thickness error e_3 to the maximum possible thickness error, which is $e_3 = 2$. With the described thickness errors, we can now measure how well a model learned a sense of the thickness.

As a first metric, we use the *Mean Thickness Error* (MTE), which is just the mean of all thickness errors e_i for all arbitrary keypoints p^i_{arb}. This metric calculates the mean deviation of the distance from an arbitrary keypoint to the skeleton line, while especially penalizing detections on the wrong side of the skeleton line. Furthermore, we introduce the *Percentage of Correct Thickness* (PCT). At a threshold t, it is defined as the fraction of thickness errors that are below t. Note that these metrics should not be used standalone as they do not take into account the absolute positions of the predictions. Only the relative position regarding the skeleton line and the body part boundaries are considered. They are only able to give a rough estimation, as the thickness error is always set to the maximum error if the keypoint does not lie on the correct body part. Therefore, in our experiments, we use the PCT in conjunction with the PCK.

5.3 Datasets

For arbitrary keypoint detection, as described in Section 5.1.3, we need body part segmentation masks. Apart from COCO (see Section 2.2.1), there are no common datasets that provide such masks, and especially no dataset in the field of sports. Therefore, we create our own datasets for this task, mostly not with hand-annotated masks, but with masks generated by a segmentation model.

5.3.1 COCO-DensePose

The original COCO [50] dataset contains over 200,000 images. We introduce it in Section 2.2.1. For our task, we need body part segmentation masks in order to generate arbitrary keypoints on the limbs. Therefore, we use the subset of COCO created for the DensePose [74] task. We use the train1 split containing 39,210 person segmentations as our training set, the val split with 2,243 person segmentations as our validation set and the train2 split with 7,297 as our test set. During the keypoint generation process, we found

that the segmentation masks contain a lot of wrong left-right annotations. We correct some of them with a heuristic and some manually, resulting in approx. 3,500 annotation corrections that we made publicly available [53]. The fixed and semantically well-defined keypoints in the COCO dataset are: l./r. eye, l./r. ear, l./r. shoulder, l./r. elbow, l./r. wrist, l./r. hip, l./r. knee, l./r. ankle.

5.3.2 IAT-Skijump v2

The IAT-Skijump v2 dataset is mostly identical to the IAT-Skijump dataset (see Section 2.2.2), but contains more videos and annotated images. The Institute for Applied Training Science (IAT) in Leipzig provided some more videos, especially of videos which led to bad detection results with the first model. The second version of this dataset contains now 11,381 annotated images from 354 jump videos. We separate 200 images for the validation set and keep 3,783 images from 121 videos for the test set. The remaining images are used for training. For example images, see Section 2.2.2.

5.3.3 BiSp-Jump v2

The BiSp-Jump v2 dataset is very similar to the BiSp-Jump dataset presented in Section 2.2.3. It consists of some more videos and annotated frames, resulting in 6,026 labeled images in total, whereby 4,101 images are used for training, 464 images for validation and 1,461 images for the test set. The dataset does not contain hand-annotated body part segmentation masks. Therefore, we use the DensePose [74] model with a ResNet101 [33] backbone and DeepLabV3 [10] as well as Panoptic FPN [39] heads from detectron2 [74] to generate them.

5.3.4 Skijump-Broadcast

We collect broadcast TV footage from 10 skijump competitions available on YouTube in order to provide a publicly available dataset for benchmarking arbitrary keypoint detection in ski jumping images. We call this dataset Skijump-Broadcast dataset. Each video of the dataset consists of 24 to 62 individual jumps with a total of 370 jumps. We annotate at maximum 8 frames per jump to have a broad diversity of jumps in the dataset. We select images during in-run and during the flight until the moment right before the landing. Over 80% of the images correspond to the flight phase. The dataset consists of images of various quality and lighting conditions, male and female athletes, and various perspectives of the ski jumpers. We annotate frames during the slow-motion replays as well, since their fidelity is often higher. We include the information if a frame was collected during a slow-motion replay in the dataset. Furthermore, the athlete's names provided in the TV broadcast were collected and added to the dataset. We split the dataset in a train, test and validation subset such that each athlete is only present in one subset. Our dataset consists of 2867 annotations: 2159 for training, 148 for validation, and 560 for testing. The annotated keypoints are head, left/right shoulder, left/right elbow, left/right wrist, left/right hip, left/right knee, left/right ankle, left/right ski tip, left/right ski tail.

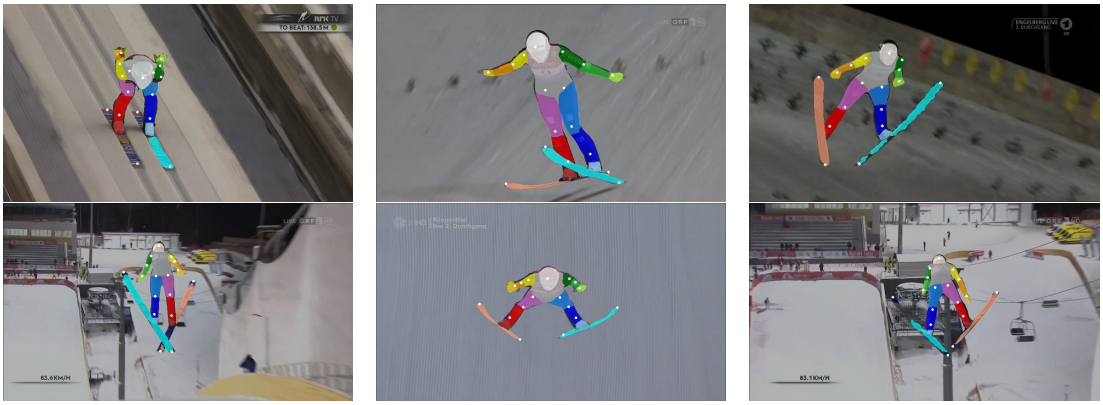


Figure 5.4: Example images from our Skijump-Broadcast dataset. The images are darkened and the segmentation masks (that are sometimes only partly correct or incomplete) are visualized with an overlay. Annotated keypoints are displayed with white circles.

We use the detectron2 [95] framework to generate segmentation masks for our dataset. In a first step, we use DensePose [74] to obtain segmentation masks of the body parts. Since images of ski jumpers are far from the domain of DensePose, most of the masks are completely or partly wrong. We select all masks that are mostly correct and discard the other ones, which results in 424 images. As detectron2 is also trained to segment skis, we feed the remaining images through an instance segmentation model in the second step. However, only a small proportion of skis is detected, and even less skis are detected correctly. A second look shows that some skis are detected, but wrongly classified as snowboards, surfboards, etc. Hence, we select and aggregate all masks that belong to skis by hand and split the ski masks in left and right ski. In many cases, only one ski is detected and/or only parts of a ski are contained in the mask. Some example images are displayed in Figure 5.4. Segmentation masks of the head, torso, left/right upper arm, left/right forearm, left/right hand, left/right thigh, left/right lower leg, left/right foot, and left/right ski are contained in the dataset: 326 segmentation masks in the train subset, 81 in the test subset and only 17 in the validation subset. Because these are too few masks for profound decisions, we coarsely label additional images with the body parts that are of interest for our research (limbs and skis), such that the validation set consists of 46 images.

5.3.5 Jump-Broadcast

We further create the Jump-Broadcast dataset to enable a public benchmark on arbitrary keypoint detection for triple, high, and long jump athletes. We have collected 26 videos of competitions from broadcast TV footage, summing up to 27 hours of video material. 9 videos cover triple jump competitions, 8 videos long jump competitions and the remaining 9 videos high jump competitions. A total of 193 different male and female athletes are present in the video footage. The sports site, lighting conditions and image quality vary throughout our dataset. Moreover, it contains a lot of extreme poses, espe-

cially during the jump phase. We select the frames by sampling approx. 5 equidistant frames from each jump (including in-run and jump phase) and each camera perspective. Slow-motion replays are mostly recorded with a different camera and therefore seen as a new camera perspective. We select 2403 images in total and annotate them with the following 20 keypoints: head, neck, left/right shoulder, left/right elbow, left/right wrist, left/right hand, left/right hip, left/right knee, left/right ankle, left/right heel, left/right toe tip. Furthermore, the annotations include information whether a frame corresponds to a slow-motion replay and the name of the athlete as is presented in the TV broadcast. We split the dataset in 1805 images for training, 576 images for testing and 122 images for validation in such a way that each athlete is only included in a single subset, even if they have participated in multiple competitions.

We use the DensePose [74] framework from detectron2 [95] to automatically generate segmentation masks for our dataset. Since some images are very blurry and the athletes perform extreme poses in comparison to everyday activities, some masks are completely or partly wrong. We sort out the worst segmentation masks by hand but keep masks that are partly correct. In the end, we keep 1797 segmentation masks, 1338 belonging to the training set, 97 to the validation set and 362 to the test set containing the head, torso, left/right upper arm, left/right forearm, left/right hand, left/right thigh, left/right lower leg and left/right foot. Figure 5.5 visualizes some images with generated segmentation masks.

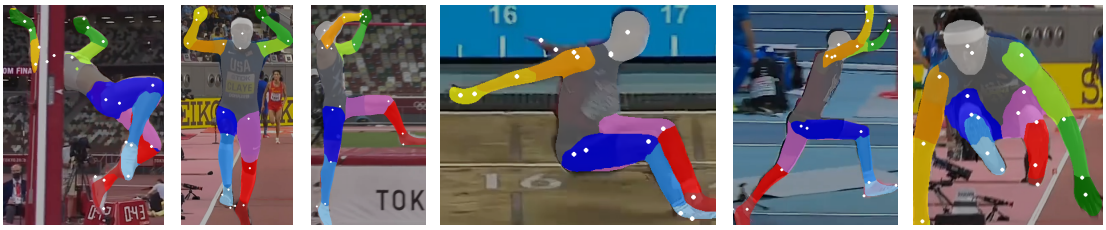


Figure 5.5: Cropped example images from the Jump-Broadcast dataset. Segmentation masks are displayed as an overlay. Annotated keypoints are visualized in white. These examples show the variety of poses in our dataset, including occlusions, front and side views, the in-run, and extreme poses during the jump phase.

5.4 TokenPose

TokenPose [48] is a combined convolutional and Transformer architecture that we use in this part of the thesis. It is one of the first 2D HPE architectures that incorporate Transformer [89] layers in their architecture. A visualization of it is provided in Figure 5.6.

Depending on the architecture variant, TokenPose at first extracts image features with either only the stem or the stem and the first three stages of an HRNet [92]. The resulting feature maps are split into patches and embedded to vectors via a linear projection. Experiments show that an HRNet feature extractor leads to superior results

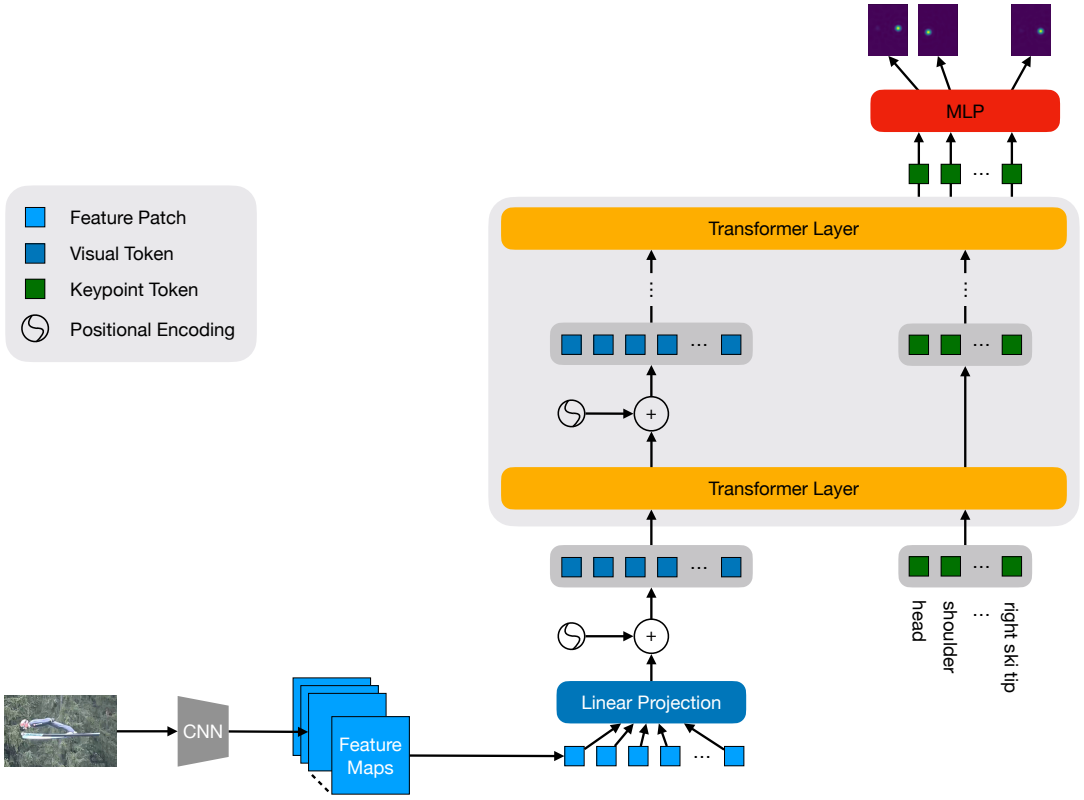


Figure 5.6: Overview of the TokenPose [48] architecture.

compared to the variant of TokenPose that omits the HRNet and directly embeds image patches into vectors.

A 2D sine positional encoding is added to all embedded image patch vectors, which we call visual tokens. The positional encoding is added to the visual tokens before the calculation of every Self-Attention (before the calculation of queries, keys and values). An ablation study shows that 2D sine works better for TokenPose than learnable positional encoding [48]. For every standard keypoint, TokenPose creates a learnable keypoint token. All these keypoint tokens are appended to the sequence of visual tokens. The joint sequence is then fed to multiple Transformer layers (6 to 24, depending on the variant). In every layer, the positional encoding is added to the visual tokens, but not to the keypoint tokens.

After the last Transformer layer, only the output of the keypoint tokens is used. An MLP with the same weights for every keypoint is learned to transform the keypoint tokens to small heatmaps. Formally, TokenPose outputs $k + n$ tokens $v_i \in \mathbb{R}^n, i = 1, ..., n + k$ with embedding dimension d, whereby the last n tokens are visual tokens and the first k tokens the keypoint tokens. The head then consists of a small MLP with shared weights θ that transforms the keypoint vectors to heatmaps of size $h \times w$.

$$\text{MPL}(\theta; v_i) = h_i \in \mathbb{R}^{h \cdot w} \xrightarrow{\text{reshape}} h'_i \in \mathbb{R}^{h \times w}, \qquad i = 1, ..., k. \tag{5.7}$$

The heatmaps are then used to calculate the keypoint coordinates. Experiments show that the largest TokenPose variant, TokenPose-L/D24 is able to outperform HRNet-W48 [92] on the COCO [50] dataset for almost all common metrics, although having a lot less parameters and computational cost. [48]

6 Intermediate Keypoint Detection with Vision Transformers

The stick-figure representation of humans in standard 2D HPE approaches is limited to a small set of fixed keypoints and the skeleton connecting these keypoints. This representation is sufficient to get a coarse notion of the current pose of a person in an image or a motion that a person performs in a video. However, for more in-depth analyses of motions and body postures, the stick-figure representation is not enough. Therefore, we take the first step towards detecting more keypoints on the human body and extend the detection capabilities of our model to any point on the skeleton, which we call *intermediate keypoints* as introduced in Section 5.1.1. Intermediate keypoints can further help to identify errors in the detection of the standard keypoints. If they do not form a straight line, it is likely that at least one of the standard keypoints on the edges of that skeleton line is detected wrongly. The work in this chapter is mainly based on the following publication, with some text passages directly taken from it:

Detecting Arbitrary Intermediate Keypoints for Human Pose Estimation with Vision Transformers [53], Katja Ludwig, Philipp Harzig and Rainer Lienhart, Winter Conference on Applications of Computer Vision (WACV) Workshops 2022, Waikoloa, HI, January 2022.

Training a model that can detect such intermediate keypoints poses some challenges. Most Human Pose Estimation datasets have a fixed set of keypoints. Hence, trained models are only capable of detecting exactly these standard keypoints. Adding new keypoints to the dataset requires a full retraining of the model. In contrast, we present a model based on the Vision Transformer architecture that can detect the standard keypoints and intermediate keypoints without any computational overhead during inference time. Furthermore, independently detected intermediate keypoints can improve analyses derived from the keypoints such as the calculation of body angles. Our approach is based on TokenPose [48] and replaces the fixed keypoint tokens with an embedding of human-readable keypoint vectors to keypoint tokens. For ski jumpers, who benefit from intermediate detections especially of their skis, this model achieves the same performance as TokenPose on the fixed keypoints and can detect any intermediate keypoint directly.

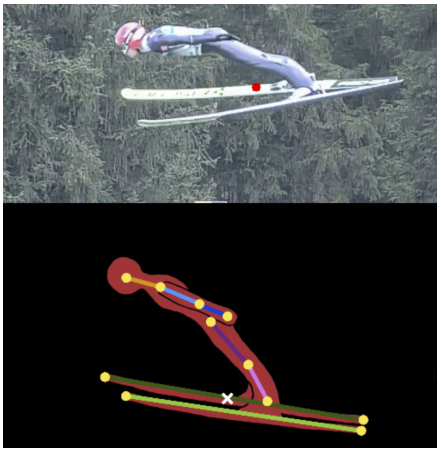


Figure 6.1: Intermediate keypoint detection. The white cross on the silhouette in the lower image shows the selection of the keypoint. In the upper video frame, the corresponding detected keypoint is displayed with a red circle. In this visualization application, the user can move the white cross in the silhouette to change the definition of the keypoint.

6.1 Introduction

As already mentioned in the previous chapters, video analysis based on the location of keypoints is a popular technique to evaluate and improve the performance of athletes in many individual sports disciplines. In the case of ski jumpers, as discussed in Chapter 3, keypoint locations are used to calculate body angles, which helps to evaluate their body posture during flight and achieve longer jumping distances. 2D HPE techniques can automate the detection of the keypoint locations, which makes the video analysis less time-consuming and available to more athletes. Ski jumpers are mainly interested in angles between body parts or skis, hence intermediate keypoints can help to get a more robust estimation of the angles. With more keypoints, a body part or ski angle is not solely based on the detection of the two end keypoints, but can be calculated as a mean of angles between arbitrary keypoints on the body part or ski. Other analyses that are based on intermediate keypoints or parameters derived from the keypoints can also be improved as the detections of intermediate keypoints can be used in conjunction with the interpolations of the skeleton line based on the standard keypoints.

In general, 2D HPE is a task of high interest in computer vision research. For a long time, architectures that use deep convolutional neural networks were most common because of their high performance in visual tasks. Typically, these models are trained on a dataset with a fixed keypoint definition and the goal is to achieve a localization of these standard keypoints as precisely as possible. The convolutional neural networks learn low and high level features in their backbones combined with a head network that learns to extract detection heatmaps for each defined keypoint based on the backbone features. These head networks are fixed to the defined standard keypoints. Adding new keypoints requires a full new training of at least the network head. Recently,

Transformer [89] networks have emerged from natural language processing tasks to vision tasks. In language applications, all words are at first embedded to vectors of fixed dimensions. Transformers can handle input sequences of various length, which is useful for sentences. Vision Transformer (ViT) [18] architectures split images in patches and embed these image patches to vectors like the words. In order to detect keypoints, TokenPose [48] appends additional learnable vectors to that sequence of embedded image patches. Each of these so-called tokens corresponds to a defined keypoint in the dataset. A more detailed explanation can be found in Section 5.4. In the standard TokenPose model, it is not possible to remove and append tokens as needed, because the network relies on the intermediate representations of all tokens to generate suitable predictions. Therefore, we extend and modify the TokenPose architecture so that we can train and evaluate an arbitrary number of intermediate points. We learn a linear transformation of vectors representing the desired keypoints to the embedding space instead of tokens representing the standard keypoints. Therefore, we can design the keypoint vector during inference time according to the keypoints that we like to detect. Figure 6.1 shows such an intermediate keypoint detection. [48]

The contributions of the work presented in this chapter can be summarized as follows:

- Our proposed training method makes the TokenPose predictions independent of the provided keypoint tokens. Trained with our method, it can detect a subset of the known keypoints without the necessity of all tokens being present.
- We extend the TokenPose model to detect arbitrary intermediate points. The desired points are encoded in human understandable keypoint vectors which are embedded in the token space through a learned linear transformation.
- Experiments show that our method can detect arbitrary intermediate keypoints of ski jumpers while maintaining the detection performance of the standard keypoints of ski jumpers. Furthermore, the method also works on another sports dataset with triple and long jump images and on the COCO [50] dataset.

6.2 Related Work

In many sports disciplines, computer vision is a valuable tool for analyzing athletes. In addition to the works discussed in the previous chapters, we would like to highlight another method focussing on skiing athletes: the approach proposed by Wang et al. [91], which estimates the poses of freestyle skiers and offers pose correction along with exemplar-based visual suggestions to the athletes.

In sports, 2D HPE is a very common technique among computer vision analysis applications. For many years, the approaches with the best scores on leaderboards of common benchmarks like COCO [50] or MPII Human Pose [1] were based on convolutional neural networks, therefore we used them in the last chapters. Contrary to these mostly fully convolutional approaches, TokenPose [48] is a Transformer [89] based approach for Human Pose Estimation. It is usable without any convolutions, but it achieves the best and

state-of-the-art results with a part of a High Resolution Net (HRNet) [92] as a feature extractor. The basic Transformer [89] architecture takes sequences of 1D tokens as an input. In order to deal with 2D images or feature maps, Vision Transformer [18] proposes to embed small image patches by a learned linear projection to 1D token vectors. This approach is used by TokenPose. Additionally, learnable keypoint tokens are appended to the image tokens and used as the Transformer input. The output of these keypoint tokens is then transformed through an MLP to heatmaps.

6.3 Method

Our model is based on TokenPose [48], which is a combined convolutional and Transformer architecture explained in Section 5.4. Since TokenPose achieves the best results with an HRNet feature extractor, we use it, too. Basically, the method and architecture proposed in this chapter are applicable to all TokenPose variants. We choose TokenPose-Base, since it provides a good trade-off between execution time and performance.

6.3.1 Additional Tokens for Intermediate Keypoints

In the standard TokenPose architecture (see Section 5.4), a unique keypoint token is learned for each standard keypoint. These tokens are appended to the image patches and fed jointly through the Transformer network. In the end, the outputs of the Transformer network that correspond to these tokens are converted to heatmaps with an MLP. Hence, the naive approach to add more keypoints is to create a token for each added keypoint and train the network on the larger keypoint set. [48]

6.3.1.1 Learning Independent Keypoint Tokens

This method has disadvantages. First, it always detects all intermediate keypoints at all times, even if only a subset of intermediate points is desired. In order to achieve a fine-grained detection of any intermediate keypoint, lots of new keypoint tokens have to be added. Removing any of the unnecessary tokens from the Transformer input sequence in order to receive only the necessary detections fails, as shown by evaluations in Section 6.4. We suppose the reason is that the Transformer network correlates all embeddings of the input with each other, visual tokens as well as keypoint tokens. Therefore, the result for each keypoint is dependent on the intermediate representations of the feature maps corresponding to the other keypoint tokens. If these tokens are not given in the input sequence, the Transformer misses necessary information to generate precise predictions. This dependence is a desired effect in TokenPose, as the intermediate results of neighboring keypoints help the model to detect occluded keypoints. This effect is called *constraint cue* in TokenPose. [48]

We can alter the model so that it can cope with our scenario. In each training step, we randomly select a subset of the keypoints, whereby the number of selected keypoints is random as well, and permute them. As an input to the Transformer model, we use only the tokens that correspond to the selected subset of keypoints, the other tokens are

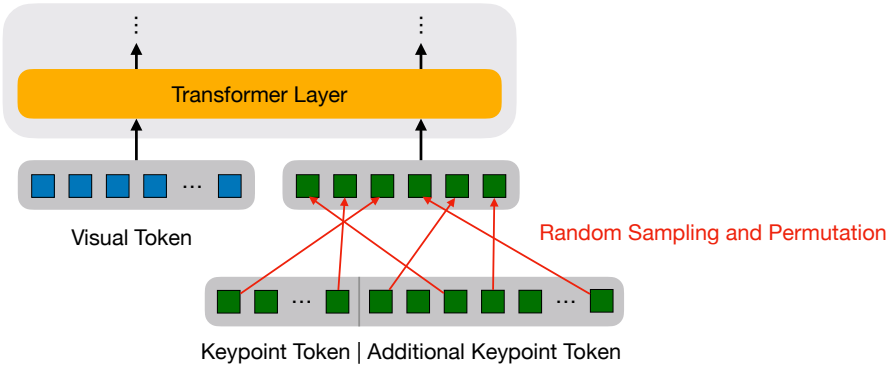


Figure 6.2: Schematic representation of keypoint sampling and permutation. For each additional keypoint, a token is added. During training, a random subset of all keypoint tokens is selected and randomly permuted. Only the selected keypoints are appended to the visual tokens as an input to the Transformer network.

not present in the input sequence. Figure 6.2 visualizes this technique. Consequently, the loss is calculated based on the sampled keypoints. Solely permuting the keypoint tokens is not sufficient, as Transformer networks are independent of the input sequence order to a certain extent. Permutation and random keypoint sampling is necessary that the Transformer learns to be quite independent of the present keypoint tokens, but evaluations show that there is still a slight performance drop if less keypoint tokens are used (see Section 6.4). [1]

6.3.1.2 Analysis of Keypoint Tokens

Our goal is to detect arbitrary intermediate keypoints directly via the network. We can use the described naive approach to achieve a relatively fine-grained spacing of the intermediate keypoints. However, the computational complexity of the Transformer increases quadratically with the number of tokens. Hence, there is an upper limit for the number of tokens that can be used, and it is infeasible to detect all possible intermediate keypoints. An idea to solve this problem is to create keypoint tokens based on the learned neighboring tokens. To verify the feasibility of this approach, we investigate the inner product matrix of the learned keypoint tokens, which shows the similarity of the tokens to each other. A detailed analysis of this matrix reveals a problem. In Figure 6.3, the inner product matrix of left and right ski with nine equally spaced intermediate keypoints is displayed. The matrix shows that the similarity of neighboring keypoint tokens is mostly high, but smaller than the similarity between the corresponding left and right keypoint token. Hence, it is not possible to design further combined keypoint tokens to detect arbitrary intermediate keypoints as there is an interference between tokens corresponding to left and right.

[1] The method described here is functional but not optimal. A more effective approach involves using a modified attention mechanism similar to cross-attention [9]. At the time of this work, we were unaware of this technique, but we present it in Section 7.4.3.

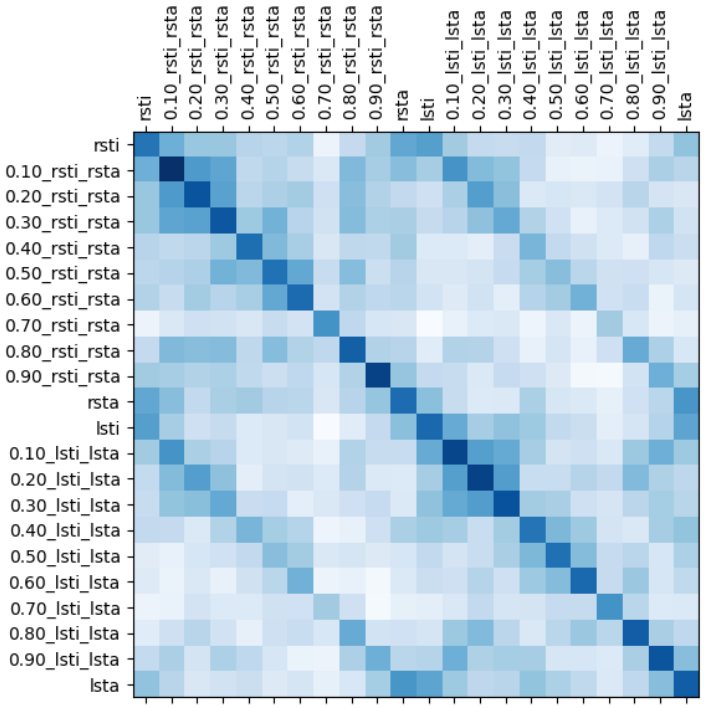


Figure 6.3: Inner product matrix for left and right ski with intermediate points. *rsti/lsti* stands for right/left ski tip, *rsta/lsta* stands for right/left ski tail. α*_sti_sta* means the keypoint is located on fraction α of the line between *sti* and *sta*. With increasing similarity, the color darkens, indicating a high similarity between corresponding left and right keypoints, which can be seen by the dark-colored diagonal squares in the top right and bottom left part of the matrix.

Figure 6.4 shows another aspect of the learned tokens. We create a token that is designed as the first x values from the right ski tip token and the next $d-x$ values from the right ski tail token, whereby d is the embedding size. If we use this token as an input, the model generates a heatmap with two peaks at the positions of both keypoints. We find that the value of x, for which the heatmap peaks shift from the first token to the second token detection is highly dependent on the keypoints and the inner logic of the tokens. If we use neighboring intermediate keypoints for this strategy the results are better, but the detections still produce either two heatmap peaks or jump between the two learned keypoints. Therefore, it is not possible to design keypoint tokens for generating fluent intermediate keypoint predictions. In order to generate continuous intermediate keypoints, we would need to learn a huge amount of intermediate keypoints which is infeasible, especially if we want to preserve the initial performance of the network.

6.3.2 Intermediate Keypoints Encoded in Vectors

Therefore, we design another architecture, which is visualized in Figure 6.5. At first, features are extracted from the images with an HRNet [92] backbone. The feature

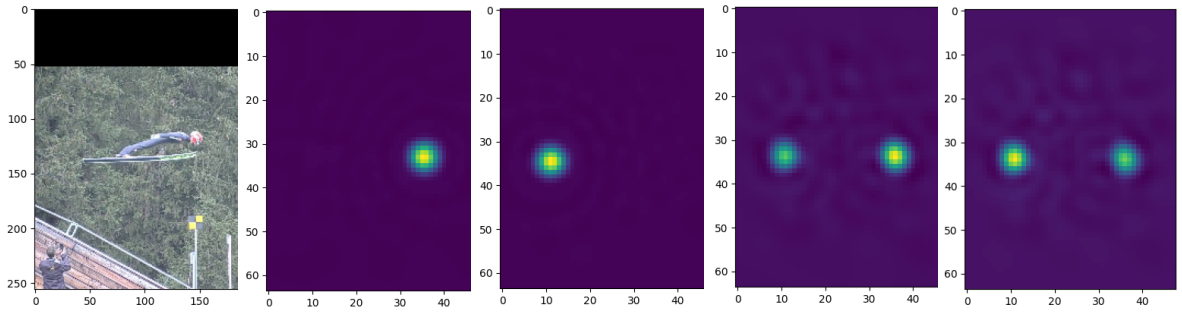


Figure 6.4: Detections with different tokens. The first image is the input image. The second image shows the detection of the right ski tip, the third image the detection of the right ski tail. Image four shows the result if we use the first approx. 60% of the values from the ski tip token and the last values from the ski tail token. Image five shows the result with the first approx. 40% values from the ski tip and the last entries from the ski tail token.

maps are split into feature patches and embedded through a linear projection. 2D sine positional encoding is added to the resulting visual tokens. Instead of appending learnable keypoint tokens to the sequence of visual tokens, we use a method similar to the feature patch embedding. We encode the keypoints as keypoint vectors and use a linear projection to create embeddings of the keypoints. Hence, the model learns the transformation from keypoint vectors to keypoint tokens. Before appending the keypoint tokens to the input sequence, we randomly sample and permute the tokens, just as described in Section 6.3.1.1. Before the Transformer layers, we always add the positional encoding to the visual tokens, but not to the keypoint tokens as we want them to be independent of the order. Like in TokenPose [48], we keep the outputs of the Transformer corresponding to the keypoint tokens and use an MLP with shared weights to generate heatmaps for the desired keypoints.

The keypoint vectors are designed in a way that all keypoints on skeleton lines are representable. If a dataset has n annotated keypoints per person, a keypoint vector v for this dataset has length n, hence $v \in \mathbb{R}^n$. If we want to detect a standard keypoint, e.g., keypoint i, then

$$v_k = \begin{cases} 1, & k = i \\ 0, & k \neq i \end{cases} \quad k = 1, ..., n$$

Let the line between keypoints i, j be a skeleton line l_s, e.g. the forearm, upper arm, thigh, lower leg, neck, etc. If a keypoint should be detected that lies on fraction α of the line between keypoints i and j, v is defined as

$$v_k = \begin{cases} 1 - \alpha, & k = i \\ \alpha, & k = j \\ 0, & k \neq i \wedge k \neq j \end{cases} \quad k = 1, ..., n$$

If $\alpha = 1$, the keypoint is located at the end on the line from i to j, hence the desired keypoint equals keypoint j. If $\alpha = 0$, the keypoint definition is equal to keypoint i.

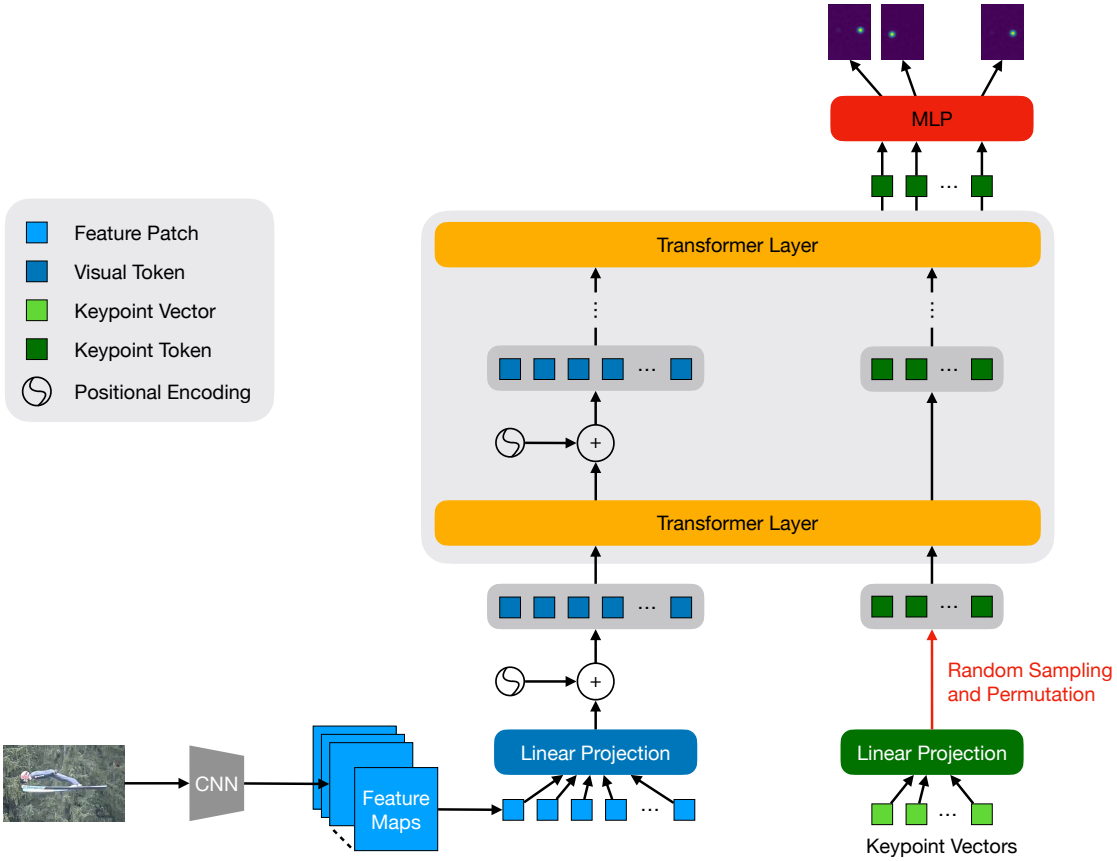


Figure 6.5: Model architecture with keypoint vectors. Image features are extracted with a convolutional neural network, split into feature patches and transformed to visual tokens using a learned linear projection. Keypoint vectors are treated similarly. They are transformed to keypoint tokens through a learned linear projection. Both linear projections are independent of each other. Positional encoding is added to the visual tokens, but not to the keypoint tokens. Keypoint tokens are randomly sampled and permuted before they are fed through the Transformer network. A full attention is calculated with visual and keypoint tokens. An MLP is used to transform the resulting keypoint tokens to heatmaps. [50]

For our training, we use all standard keypoints and randomly generate intermediate keypoints p_{inter}, sampled from all skeleton lines of the dataset with α sampled uniformly from $[0, 1]$. An example for the resulting keypoint vectors for the ski jump dataset is visualized in Figure 6.6.

With this method, it is also possible to generate keypoints that are located on the straight line between two intermediate keypoints p_{inter}. If we take the COCO dataset, for example, and want to generate keypoints on the spine, these keypoints are located on the line between keypoints i and j, whereby keypoint i is located in the middle of the two shoulder keypoints s_1, s_2 and keypoint j in the middle of the left and right hip keypoint h_1, h_2. Hence, if we want to detect a keypoint of fraction α on the line between

i and j, our keypoint vector v consists of zeros apart from entries $v_{s_1} = v_{s_2} = (1-\alpha)\cdot 0.5$ and $v_{h_1} = v_{h_2} = \alpha \cdot 0.5$. So, we can design any intermediate keypoint that lies between annotated keypoints.

head	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
shoulder	0	1	0	0	0	0	0	0	0	0	0	0.5	0	0	0	0	0
elbow	0	0	1	0	0	0	0	0	0	0	0	0.5	0	0	0	0	0
hand	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
hip	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0.8
knee	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0.2
ankle	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
right ski tip	0	0	0	0	0	0	0	1	0	0	0	0	0.5	0.2	0	0	0
right ski tail	0	0	0	0	0	0	0	0	1	0	0	0	0.5	0.8	0	0	0
left ski tip	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0.7	0.5	0
left ski tail	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0.3	0.5	0
	standard keypoints											generated keypoints					

Figure 6.6: Keypoint vectors for the ski jump dataset. The keypoint vectors are displayed in the columns. The first eleven vectors correspond to the standard keypoints. The last six keypoints are generated, whereby the number of generated keypoints is chosen randomly. The first generated keypoint lies on the upper arm, the second and third keypoint on the right ski, keypoint four and five on the left ski and the last keypoint on the thigh. For example, the last keypoint is located at 20% of the length on the line between hip and knee, hence it is closer to the hip.

6.3.3 Exponential Moving Average

The validation score has a high fluctuation rate throughout the training, even after convergence. In order to reduce fluctuation and achieve a model with a stable result, we keep an exponential moving average (EMA) of our model. This is the same technique as used for the mean teacher semi-supervised learning in Section 4.3.2. Let w^t be the weights of our model and w^t_{EMA} be the weights of our EMA model. We initialize the EMA model with the exactly same weights as the original model

$$w^0_{EMA} = w^0.$$

Let γ be the EMA rate, then all weights of the EMA model at training iteration t are calculated as

$$w^t_{EMA} = \gamma \cdot w^{(t-1)}_{EMA} + (1-\gamma) \cdot w^t.$$

The EMA model can be seen as a temporal ensemble of the models from the last iterations, with more recent weights having a larger impact after a warm-up phase.

6.4 Experiments

All experiments are based on the TokenPose-Base architecture [48]. At first, feature maps are extracted using the first three stages of a HRNet-w32 network [92]. The largest feature map of the HRNet output, which has $\frac{1}{4}$ of the input resolution, is used as input to the Vision Transformer network for each image. All input images are resized to 256×192 and the resulting feature maps are split into patches of size 4×3. The patches are embedded to vectors of size 192, and we use 12 Transformer layers with 8 heads. We use 2D sine positional encoding, which is added to the visual tokens and corresponding intermediate representations before each Transformer layer, meaning before the calculation of keys, queries and values for the Self-Attention. To obtain the keypoint coordinates from the heatmaps with resolution 64×48, we use the *DARK* method presented by Zhang et al. [102]. It uses a Taylor series expansion of the heatmap around its initial maximum to achieve sub-pixel accuracy.

6.4.1 IAT-Skijump v2

As described in Chapter 3, ski jumpers analyze their body and ski angles during the flight phase for performance improvement. These angles, also called flight parameters, are derived from the detected keypoints on the ski jumpers and their skis. These keypoints can be robustly estimated by a combination of using multiple detections per camera view and heuristic filtering. The detection of intermediate keypoints can improve the robustness of the angle estimation even more, since multiple points can be used for a single angle calculation. Therefore, we use the IAT-Skijump v2 dataset (see Section 5.3.2) as the primary dataset of interest in this chapter. We use the PCK as explained in Section 2.1.2.1 as the evaluation metric. We set the threshold to $t = 0.1$, which corresponds to approx. 5 cm in this dataset.

For all ski jump experiments, we use pretrained weights from the COCO dataset. At first, we train a standard TokenPose-Base model (see Table 6.1, first row). With an overall PCK@0.1 of 80.7%, the Transformer model achieves a similar performance in comparison to a pure HRNet-w32 model, which scores 80.8% PCK. If we use the EMA model like described in Section 6.3.3, the total PCK rises to 81.9% and exceeds the HRNet score (see Table 6.1, second row). This shows that the temporal ensemble included in the EMA model improves the model performance. In the next experiments, we will stick to the EMA model for our results.

In order to detect continuous keypoints on the body parts of the ski jumpers and the skis, we add 36 intermediate keypoints to the model. Three equally spaced intermediate points are added to the neck, upper arm, forearm, thigh, lower leg and torso, as well as nine equally spaced intermediate points to left and right ski. A training on these keypoints lowers the detection score on the standard keypoints by 0.8%, as the model is now trained on a larger problem and can not focus on the standard keypoints. But regarding the performance on all keypoints, including the added keypoints, the performance rises by 4.1% compared to the base model evaluated on the standard keypoints (see Table 6.1, third row).

	Model	Adaptations	Inp.	K	Head	Shou.	Elb.	Hand	Hip	Knee	Ank.	rSTi	rSTa	lSTi	lSTa	Avg	Full
1	TokenPose	non-EMA	std		98.3	91.3	73.2	65.1	85.0	80.1	83.7	73.9	81.9	67.4	81.2	80.7	
2	TokenPose	EMA	std		98.9	93.2	75.6	65.9	87.0	81.6	83.8	74.6	82.2	68.9	81.6	81.9	
3	TokenPose	36 intermediate	all		98.7	93.5	74.5	64.8	86.6	81.3	81.8	74.7	80.8	68.3	80.0	81.1	86.0
4		keypoints	std		0.9	52.0	72.5	0.5	11.1	0.0	1.1	0.2	0.0	0.3	0.0	14.4	
5	TokenPose	token	all		98.3	91.2	71.6	63.5	83.5	78.6	75.8	69.5	77.4	62.8	76.0	77.9	83.2
6		permutation	std		10.8	84.7	8.7	0.2	33.3	0.1	6.1	21.9	37.2	21.0	37.8	23.0	
7	TokenPose	token permutation	all		98.9	92.9	74.1	62.9	85.8	80.2	81.1	73.3	79.8	64.9	78.4	80.1	85.3
8		& sampling	std		98.8	92.8	73.4	62.6	85.6	79.7	81.0	72.7	79.0	65.1	77.6	79.7	
9	Ours	token permutation &	all	✓	98.2	91.1	72.4	65.0	83.7	79.8	78.6	72.4	80.8	65.6	79.3	79.4	82.1
10		sampling of **std.** keypoints	std	✓	97.2	85.1	71.0	58.5	81.8	76.7	76.8	20.6	78.8	20.2	77.5	70.3	
11	Ours	token permutation	all	✓	98.9	93.5	75.0	65.8	87.4	82.0	83.1	75.7	81.9	68.7	81.0	81.8	84.8
12		& sampling	std	✓	98.9	93.4	74.8	65.6	87.3	82.2	83.2	75.2	82.0	68.9	81.1	81.8	
13		of **all** keypoints	sgl	✓	98.8	93.2	75.0	65.6	87.3	82.2	82.9	68.7	81.5	63.4	80.8	80.9	

Table 6.1: Recall values in % at PCK threshold 0.1 for the IAT-Skijump v2 dataset of all keypoints (head, shoulder, elbow, hand, hip, knee, ankle, right ski tip, right ski tail, left ski tip, left ski tail), and the average PCK over all 11 keypoints (second to last column). If more than these 11 standard keypoints are used during training, the PCK score including the generated points is given in the last column (Full). These values are not directly comparable, as all TokenPose methods are only able to detect some, but not all possible intermediate keypoints. If the keypoint vector model is used, this is indicated in the third column labeled *K*. The qualifiers *std*, *all* and *sgl* in the input column refer only to the used inference protocol and not to the training procedure. *std* means that only the keypoint tokens/vectors that correspond to the standard joints are used as an input during inference, *all* means that the full input as during training is used and *sgl* stands for single evaluation, meaning that a keypoint vector representing a single keypoint is passed to the model and all keypoints are obtained separately. All models are EMA models except for the first row.

However, these scores are only achieved if we use all 47 learned keypoint tokens in the input. Using only the keypoint tokens corresponding to the 11 standard keypoints instead of the 47 used during training, the score drops to only 14.4% PCK, although we evaluate with the same model (see Table 6.1, row four). In this evaluation, it is remarkable that the performance varies extremely between the keypoints. The elbow score is still high, while the score for the knee and the ski tails drops to zero.

Additionally permuting the 47 keypoint tokens during training results in a further performance drop for the evaluation with all 47 tokens present in the input. The detection score rises a little to 23.0% if we use the standard keypoint tokens only as in the last evaluation, but this is far too low for a usable model (see Table 6.1, rows five and six).

Including the random sampling and permutation technique changes this behavior. We randomly choose at minimum five keypoints and maximum all 47 keypoints in each training step. This method achieves a PCK of 80.1% evaluated on the standard keypoints but with all keypoint tokens in the input sequence, which is a little lower than without this method. But in contrast to all other methods, its performance drops only slightly if we evaluate with solely the standard keypoints in the input sequence, achieving a PCK of 79.7% (see Table 6.1, rows seven and eight).

To train our model that is based on keypoint vectors, we generate 1 to 30 additional keypoints on the neck, upper arm, forearm, thigh, lower leg, torso, left and right ski. The fraction α that defines the location on the body part is uniformly sampled between 0 and 1. Hence, we generate arbitrary intermediate keypoints during training. We permute and randomly sample at minimum five and at maximum all standard and generated keypoints. However, randomly sampling only the standard keypoints results in an inferior score (see Table 6.1, rows 9–13). Since the generated keypoints are already randomly produced, we expected that randomly sampling them should not have an influence, but the experiments show that this is not the case.

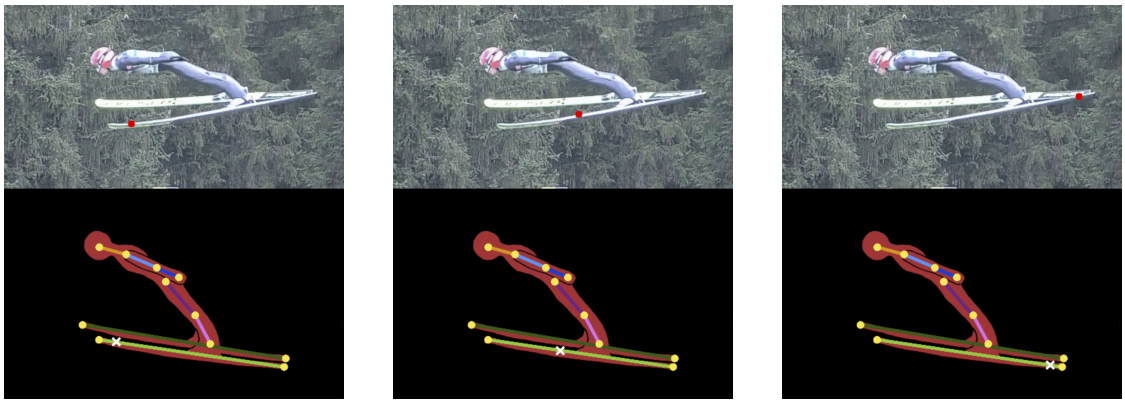


Figure 6.7: Examples for intermediate detections on the left ski. The white cross on the silhouette of the ski jumper in the lower image shows the selection of the intermediate keypoint. In the upper image, the corresponding keypoint is detected and displayed with a red circle.

	Model	Adaptations	Avg PCK All Input	Full PCK All Input	Avg PCK Std Input	Avg PCK Sgl Input
1	TokenPose	-	-	91.3		
2		50 intermediate keypoints	91.1	93.2	35.1	
3		token permutation & sampling	91.1	93.2	91.1	
4	Ours	token perm. & sampling of **std.** keypoints	91.3	92.5	90.4	
5		token perm. & sampling of **all** keypoints	91.7	92.9	91.7	91.5

Table 6.2: Recall values in % at PCK threshold 0.1 for the BiSp-Jump v2 dataset. The first column displays the average PCK of the standard keypoints with full input (all keypoint tokens/vectors) as used during training. The average PCK score including the generated points is given in the second column. The third column shows the average PCK of the standard keypoints with only the keypoint tokens/vectors of the standard keypoints present in the Transformer input and the last column the evaluation with keypoint vectors representing only single joints. We abbreviate permutation with perm. in this table.

The keypoint vector model with full random sampling achieves a PCK score of 81.8% independent of the keypoint vectors in the input. This is a very similar performance as training solely on the standard keypoints, but the model is capable of directly detecting arbitrary intermediate keypoints, which is proven by the higher full PCK score of 84.8%. If we evaluate the model only on generated keypoints (randomly between 1 and 30 randomly chosen arbitrary keypoints), we get a PCK score of even 86.3% (not in the table). This proves that the model can really detect arbitrary points. The model achieves also good results if only a single keypoint vector is used as an input, the average PCK is 80.9% in this case. Hence, the dependence on the other keypoints is reduced to a minimum and the model is really flexible for the sole detection of the desired keypoints. Figure 6.7 shows some examples for intermediate keypoint detections.

6.4.2 BiSp-Jump v2

We further evaluate the capabilities of our model on the BiSp-Jump v2 dataset that is introduced in Section 5.3.3. We evaluate again with PCK@0.1, which corresponds to approx. 6 cm in this dataset. The evaluation results are presented in Table 6.2. The TokenPose model with standard keypoints achieves a PCK score of 91.3% (see Table 6.2, first row). We add 50 intermediate keypoints on the neck, upper arm, forearm, thigh, lower leg, shoulder axis, hip axis, spine and the lines between neck and left/right hip as well as left/right shoulder and neck. The performance on the standard joints is similar, but the performance with solely the standard input keypoint tokens drops to 35.1% (see Table 6.2, second row). Uniformly sampling between five and all keypoints rises the detection score of the standard keypoints to 91.1% independent of the number of tokens in the input sequence (see Table 6.2, third row). The keypoint vector model is capable of increasing the performance further, even surpassing the standard model's performance slightly with 91.7% PCK independent of the keypoint vector input (see

Table 6.2, row five). For that model, we generate 5–50 keypoints uniformly distributed between all body parts during training. If we use keypoint vectors that represent only a single joint and evaluate these results, the model achieves an accuracy of 91.5% PCK, which is still better than the standard model.

6.4.3 COCO

Moreover, we evaluate on the COCO [50] dataset, which is a very common benchmark, but not a sports dataset. We have introduced it in Section 2.2.1. We use the standard evaluation metric for COCO, which is AP based on OKS, as described in Section 2.1.2.2. The results are displayed in Table 6.3. Our TokenPose training on COCO with the standard keypoints achieves an AP of 74.8% without the EMA model and an AP of 75.4% with the EMA model (see Table 6.3, first and second row). We add 53 points on the head, neck, upper arm, forearm, thigh, lower leg, shoulder axis, hip axis, spine and the lines between neck and left/right hip as well as left /right shoulder and neck. The performance drop using only the standard keypoint input tokens is less on the COCO dataset, but still significant. Using all tokens (standard and generated ones) in the input sequence, the model achieves an AP of 73.7% (see Table 6.3, row three). With the standard tokens only, the AP drops to 56.5% (see Table 6.3, row four). Using our keypoint vector model, the detection performance is nearly equal for full, standard, and single input (see Table 6.3, rows seven to nine). As the full PCK is higher than the standard PCK, this proves that the model is capable of precisely detecting arbitrary intermediate keypoints. The PCK in general is lower than the PCK of the TokenPose

	Model	Adaptations	Inp.	AP	AP^{50}	AP^{75}	AP^{M}	AP^{L}	AR	Avg PCK	Full PCK
1	TokenPose	non EMA	std	74.8	92.4	81.5	71.9	79.3	77.8	81.4	
2	TokenPose	EMA	std	75.4	92.5	82.5	72.5	79.9	78.3	81.6	
3	TokenPose	53 interm. keypoints	all	73.7	91.5	80.7	71.1	77.8	76.6	81.3	82.6
4			std	56.5	87.4	65.0	54.6	59.8	60.0	68.1	
5	TokenPose	token permutation & sampling	all	69.4	89.5	77.5	66.9	73.6	72.8	78.7	80.3
6			std	69.5	89.5	77.4	66.9	73.5	72.7	78.6	
7	Ours		all	73.7	91.5	80.6	71.2	78.0	76.6	81.0	81.8
8			std	73.6	91.5	80.6	71.2	77.9	76.6	81.0	
9			sgl	73.5	91.5	80.6	70.9	77.8	76.5	80.9	

Table 6.3: OKS results and average recall values at PCK threshold 0.1 in % on the COCO dataset. If more than the standard keypoints are used, the PCK score including the generated points is given in the last column. The qualifiers *std*, *all* and *sgl* in the input column refer only to the used inference protocol and not to the training procedure. *std* means that only the keypoint tokens/vectors that correspond to the standard joints are used as an input during inference, *all* means that the full input as during training is used (either 70 keypoints or keypoint vectors with generated keypoints) and *sgl* stands for single evaluation, meaning that a keypoint vector representing a single keypoint is passed to the model during inference and all keypoints are obtained separately.

model trained on the standard keypoints, but this is most likely caused by the training duration. We stop the training for all models after 1.2 million steps, but the keypoint vector model has not fully converged till that time as it learns slower due to the task of learning continuous points being more complex. Further, the full PCK values are not directly comparable, since the TokenPose models are only able to detect some, but not all possible intermediate keypoints.

6.5 Summary

At first, this chapter has proposed a training routine that makes it possible to train a TokenPose [48] model with additional keypoints, but independent of these additional keypoints during inference. For the standard TokenPose model, all keypoint tokens need to be present in the Transformer input, as the model learns the location of the keypoints not only from the image but also in dependence on the detections of the other keypoints. Random sampling and permuting the keypoint tokens in the input forces the model to learn the keypoint locations independent of the other keypoints.

This model has the disadvantage that only the intermediate keypoints that are trained can be detected afterwards. Designing the tokens such that the model is able to estimate arbitrary intermediate keypoints is not possible as the coherence of tokens for corresponding left and right keypoints is larger than the coherence of neighboring tokens. Hence, we proposed a novel architecture that uses keypoint vectors instead of keypoint tokens as an input. Keypoint vectors have the same length as the number of annotated keypoints in the dataset. They sum up to 1 and represent an arbitrary intermediate keypoint as a mixture of the standard keypoints. Keypoint vectors are embedded like the visual patches with a linear transformation in the embedding space. Instead of learning fixed keypoint tokens, our model learns this linear embedding.

Evaluations have shown that our keypoint vector model in combination with the random sampling strategy works as desired. The PCK on the standard joints is similar or even slightly better for all three experiments with different datasets. The PCK including arbitrary intermediate keypoints is even higher, hence, the model can really detect any desired intermediate keypoint. The detection of single keypoints is also nearly completely independent of other keypoints, which has been proven by an evaluation with only single keypoint vectors as an input to the Transformer.

7 Detecting Arbitrary Keypoints on Humans and Their Sports Equipment

Recall that nearly all HPE datasets consist of a fixed set of keypoints. Standard HPE models trained on such datasets can only detect these keypoints. If more points are desired, they have to be manually annotated and the model needs to be retrained. Therefore, in Chapter 6, we introduced an approach that leverages the Vision Transformer architecture to extend the capability of the model to detect intermediate keypoints along the skeleton of humans. These intermediate keypoints enable or improve certain applications, such as estimating body angles in ski jumping. However, intermediate keypoints alone remain limiting for more comprehensive analyses that require, for example, detecting the body outline. To address this, we introduce an approach for detecting arbitrary keypoints across the human body in this chapter. All approaches are based on the TokenPose [48] architecture, like the intermediate keypoint approach in Chapter 6. Since the task of detecting arbitrary keypoints is very challenging, we approach it in multiple steps. All the steps that are described in this chapter are based on the following publications, with some text passages directly taken from them:

Recognition of Freely Selected Keypoints on Human Limbs [54], Katja Ludwig, Daniel Kienzle and Rainer Lienhart, IEEE/CVF International Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), New Orleans, LA, June 2022.

Detecting Arbitrary Keypoints on Limbs and Skis with Sparse Partly Correct Segmentation Masks [55], Katja Ludwig, Daniel Kienzle, Julian Lorenz and Rainer Lienhart, Winter Conference on Applications of Computer Vision (WACV) Workshops 2023, Waikoloa, HI, January 2023.

All Keypoints You Need: Detecting Arbitrary Keypoints on the Body of Triple, High, and Long Jump Athletes [57], Katja Ludwig, Julian Lorenz, Robin Schön and Rainer Lienhart, IEEE/CVF International Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), Vancouver, BC, Canada, June 2023.

In the first step, we focus on arbitrary keypoints on the limbs of persons. Hence, we include arbitrary keypoints on the upper arm, forearm, thigh, and lower leg. We propose two different approaches to encode the desired keypoints.

1. Each keypoint is defined by its position projected onto the skeleton line and its relative distance from this line to the limb's edge

2. Keypoints are defined as coordinates on a normalized pose.

In both cases, arbitrary keypoints are defined relative to the corresponding segmentation mask of the body part, like explained in Section 5.1.3. Experiments show that both approaches achieve similar results to TokenPose on the standard keypoints and are capable of detecting arbitrary keypoints on the limbs.

In the next step, we propose a method to detect arbitrary keypoints on the limbs and skis of professional ski jumpers that requires a few, only partly correct segmentation masks during training. Since manual annotations are very costly, our approach is more usable if it only requires partly correct and not fully correct segmentation masks. The reason is that we use segmentation masks only to generate ground truth labels for the arbitrary keypoints, and partly correct segmentation masks are sufficient for our training procedure. Hence, there is no need for costly hand-annotated segmentation masks. We analyze different training techniques for arbitrary and standard keypoints, including pseudo-labels, and show in our experiments that only a few partly correct segmentation masks are sufficient for learning to detect arbitrary keypoints on limbs and skis. Moreover, we find that the standard full attention mechanism does not fit the needs of our task and propose another attention mechanism that is more suitable for detecting arbitrary keypoints.

Lastly, we focus on the disciplines of triple, high, and long jump, which require fine-grained locations of the athlete's full body. Therefore, we move one step further and detect arbitrary keypoints on the *whole body* of the athlete by leveraging the limited set of annotated keypoints and auto-generated segmentation masks of body parts. We now do not limit the body parts to the limbs as before. Evaluations show that our model is capable of detecting keypoints on the head, torso, hands, feet, arms, and legs. We further adapt our approach to include also bent elbows and knees, since such poses happen very frequently in triple, high, and long jump. We analyze and compare different techniques to encode the desired keypoints as the model's input and their embedding for the Transformer backbone.

7.1 Introduction

In this chapter, we extend the capabilities of the model introduced in Chapter 6 in multiple steps to detect arbitrary keypoints on the human body, since this enables more advanced video analyses in sports disciplines. Like in Chapter 6, our approach uses the capability of Transformer architectures to handle inputs of various length. The representations of the desired keypoints - standard as well as arbitrarily chosen - are converted to tokens. This sequence of tokens of any length is then appended to the visual tokens and the network predicts a keypoint for each token.

In a first step, we focus on arbitrary keypoints solely on the limbs of humans and create the model *V1*. We propose and evaluate different keypoint representations throughout this chapter. The first approach splits the representation into two parts. One part encodes the position of the projection of the desired keypoint onto the skeleton line l_s, which is the straight line between the standard keypoints that are enclosed by the

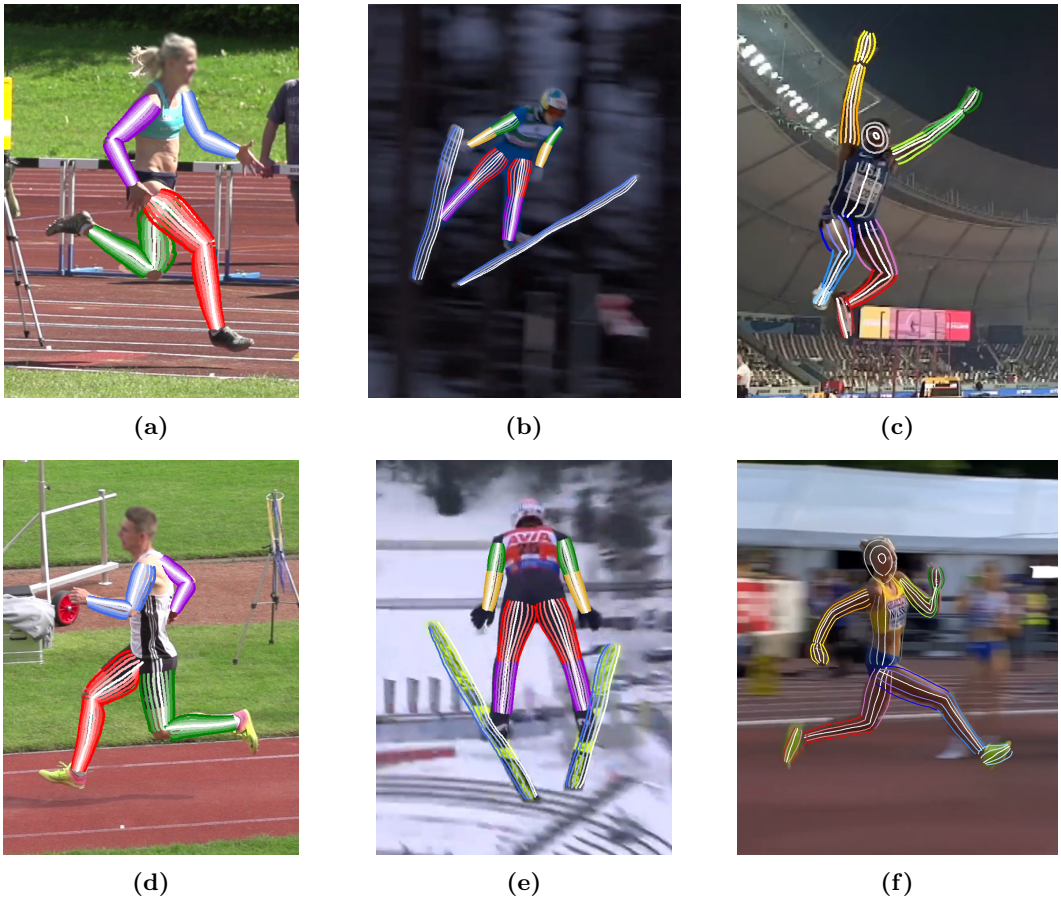


Figure 7.1: Two detection results for every step of our approach to detect arbitrary keypoints. The images in the first column (7.1a and 7.1d) display arbitrary keypoints on the limbs of triple and long jump athletes by model *V1*. The images show four equally spaced lines to both sides of each limb including the edge in pure color and the central line in white with a color gradient from one side to the other. The images in the second column (7.1b and 7.1e) show two detection results of arbitrary keypoints on the limbs and skis of ski jumpers using our model *V2*, visualized with four equally spaced lines as before. The images in the last column (7.1c and 7.1f) show detection results from our Jump-Broadcast dataset using our model *V3*, visualized with three equally spaced lines and colored as before.

corresponding body part. The second part encodes the distance of the keypoint from this intermediate keypoint p_{inter} relative to the distance of the boundary of the body part. We refer to this approach as the *vectorized keypoint* approach and the resulting model is called *V1-VK*. The second approach encodes each keypoint in normalized euclidean coordinates on a normalized pose (template T-pose). We call this the *normalized pose* approach and the trained model *V1-NP*. Both approaches open the possibility to design the keypoint representation such that desired arbitrary keypoints can be represented and therefore also detected by our models without any additional annotations or post-

processing steps. Figure 7.1 shows in the first column two examples for such detection results for the fixed as well randomly chosen keypoints of the athletes.

Secondly, we adapt the vectorized keypoint approach for TokenPose that it can also detect arbitrary points on the skis of ski jumpers. The first approach *V1* requires a segmentation mask for every image in order to detect arbitrary points on the limbs of humans. Since many images of ski jumpers are too far from the domain of common segmentation models, we are only capable of collecting a few segmentation masks of body parts that are partly correct and even fewer of the skis. The reason is that we avoid to costly annotate the images by hand but use existing segmentation models like DensePose [74, 95] to obtain the masks. We propose and analyze different methods to train a model with this small amount of segmentation masks. Results of our best model, called *V2*, can be seen in Figure 7.1 in the second column.

The methods presented in the first two steps are limited to the limbs (and skis, if applicable), which we try to overcome in the last step. Figure 7.1 shows in the last column that our final model *V3* is capable of detecting arbitrary points on the head, torso, arms, and legs including hands and feet. It also correctly estimates keypoints on bent limbs like elbows and knees.

The contributions presented in this chapter can be summarized as follows:

- We propose two different representations of arbitrary keypoints on human limbs. The first one is based on the location relative to the body part boundary and the keypoints enclosed by the body part, the second one uses the position on a normalized pose.
- Our model, based on the TokenPose architecture, uses one of the two representations to create appropriate tokens for detecting the desired keypoints. The model can deal with any number of keypoints.
- We propose a metric to evaluate the location of detected keypoints relative to the body part boundary, which is described in Section 5.2. Typical metrics like Percentage of Correct Keypoints (PCK) are not suitable to evaluate the model's sense of limb boundaries precisely.
- Our experiments show that approach *V1* can detect arbitrary keypoints on the limbs of humans while maintaining its performance on the set of standard keypoints. We evaluate the model on the COCO dataset and the BiSp-Jump v2 dataset.
- We propose an adapted representation for the vectorized keypoint query tokens in order to detect arbitrary keypoints on the skis of ski jumpers.
- Further, we improve our model such that keypoint outputs are independent of the number of keypoint tokens in the input sequence by adapting the used attention mechanism. Without this adaptation, the detection results depend on the other keypoint tokens in the input sequence.
- We release a new dataset with 2867 annotated ski jumpers (see Section 5.3.4).

- We analyze different methods to train model variants *V2* with only a few partly correct segmentation masks such that it is capable of estimating arbitrary keypoints on limbs and skis of ski jumpers while maintaining similar performance on the standard keypoint set.

- Our model's capabilities are further improved by enlarging the area of the body for which arbitrary keypoints can be detected by the feet, hands, torso, and head. We further improve our model such that it can detect keypoints on bent elbows and knees correctly, resulting in our final model *V3*.

- We propose different techniques for the model to encode the desired target keypoints. The head is either encoded based on a reference line or based on an angle.

- We release the Jump-Broadcast dataset, a new dataset with 2403 annotated triple, high, and long jump athletes sampled from 27 hours of 26 different TV broadcast videos (see Section 5.3.5).

- Experiments on the Jump-Broadcast dataset and the BiSp-Jump v2 dataset prove that our variants of *V3* are capable of detecting any desired keypoint on the body of athletes. We improve the evaluation scheme by using more keypoints and provide a detailed evaluation of the performance of keypoints located on the different body parts.

7.2 Related Work

In many sports disciplines, computer vision is a beneficial technique to analyze athletes in video footage of training or competition scenarios. We already presented related work regarding applications of computer vision in sports and especially individual sports in the last chapters.

The basic Transformer [89] architecture takes sequences of 1D tokens as an input. In order to deal with 2D images or feature maps, Vision Transformer [18] proposes to embed small image patches by a learned linear projection to 1D token vectors. This approach is used by TokenPose that we introduced in Chapter 5.4. Apart from TokenPose, there are other approaches that use the Transformer architecture for 2D HPE. Shi et al. [80] propose the first fully end-to-end multi person pose estimation framework based on Transformers. ViTPose [98] proves that pure ViT based HPE models are also capable of achieving state-of-the-art scores by adding a decoder with deconvolutions after the ViT layers. The HRFormer [99] architecture combines the ideas of the HRNet and the ViT and uses ViT layers while maintaining branches of different feature resolutions like the HRNet. Leveraging the idea of focusing on the part of the image where the person is located, Zeng et al. [101] cluster tokens of less important image areas like the background, while keeping many tokens for important areas. Ma et al. [59] follow a similar idea by deleting the tokens of unimportant image areas. They achieve a more lightweight architecture as a consequence.

Despite recent advances in Transformer models for 2D HPE approaches, we are not aware of any work that tackles the estimation of arbitrary novel keypoints while training the HPE network solely on a dataset with standard keypoint annotations and corresponding human segmentation masks.

7.3 Detecting Arbitrary Keypoints on Human Limbs

In Chapter 6, we introduce an approach to detect intermediate keypoints on the skeleton of humans. In this chapter, we extend this approach step by step to detect arbitrary keypoints on the human body. In the first step, we start with learning arbitrary keypoints on the limbs of humans, i.e., arms and legs.

7.3.1 Keypoint Generation

In order to detect arbitrarily selected keypoints on human limbs, we need to generate ground truth keypoints on the limbs. To achieve that, we use segmentation masks of upper arms, forearms, thighs and lower legs. As we want to generate keypoints that are distributed over the complete body part, we use the following generation scheme: Let p_i and p_j be the coordinates of two standard keypoints (e.g., left shoulder and left elbow joints) that are enclosed by the body part B (e.g., left upper arm). At first, we uniformly sample a ratio α of the skeleton line l_s between p_i and p_j, which results in an intermediate keypoint p_{inter}:

$$p_{inter} = \alpha \cdot p_j + (1 - \alpha) \cdot p_i \tag{7.1}$$

This method is identical to the keypoint generation of the intermediate keypoints in Chapter 6.

Next, we generate the line l_o that is orthogonal to the skeleton line l_s between p_i and p_j and goes through p_{inter}. This line has two intersection points c_l and c_r with the boundary of the body part segmentation mask B. The intersection point which is more left in the image is called c_l, the other intersection point c_r. Then, we sample $\tilde{\beta}$ from a normal distribution and define $\beta = \max(0, 1 - |\tilde{\beta}|) \in [0, 1]$. This ratio β corresponds to the distance from the projection point p_{inter} to the body part boundary, referred to as the *thickness*. With β, we create the final keypoint p_{arb} as follows:

$$p_{arb} = \begin{cases} (1 - \beta) \cdot p_{inter} + \beta \cdot c_l, & \tilde{\beta} >= 0 \\ (1 - \beta) \cdot p_{inter} + \beta \cdot c_r, & \tilde{\beta} < 0 \end{cases} \tag{7.2}$$

$\tilde{\beta}$ is drawn from a normal distribution in order to generate more keypoints on the body part boundaries, as this seems harder for the model to learn. Figure 7.2 shows some examples for such keypoint generations. Yellow points visualize p_i and p_j, a green point p_{inter} and the blue line l_o. The body part segmentation mask B is visualized by a red overlay. The mask intersection points c_l and c_r are displayed with blue points and the generated point p_{arb} with a red point.

Figure 7.2: Examples for the keypoint generation process on COCO images. The body part is visualized with a red overlay and the standard keypoints enclosed by the body part in yellow. The randomly selected projection point on the line between the standard keypoints is displayed in green and the orthogonal line in blue. The intersection points of the line with the edge of the body part are visualized in blue, while the red points visualize the final generated keypoints.

7.3.2 Keypoint Representations

In contrast to TokenPose which uses fixed learnable tokens, we need to learn an embedding function for the desired keypoint to a suitable keypoint token, as it is analyzed in Chapter 6. We propose two approaches for the input representation for this embedding function in the following sections.

7.3.2.1 Keypoint and Thickness Vectors

This approach is directly derived from the keypoint generation process. Each keypoint is represented by two short vectors, a *keypoint vector* and a *thickness vector*. For a dataset with n fixed keypoints, the keypoint vector $v \in \mathbb{R}^n$ for the keypoint p_{arb}, which is defined by α, β, $\tilde{\beta}$, i, and j as described in Section 7.3.1, is designed as follows:

$$v_k = \begin{cases} 1-\alpha, & k = i \\ \alpha, & k = j \\ 0, & k \neq i \wedge k \neq j \end{cases} \qquad k = 1, ..., n \tag{7.3}$$

This is equal to the representation in Section 6.3.2 for the intermediate keypoint p_{inter}. The second, novel representation vector is called *thickness vector*, $w \in \mathbb{R}^3$, and is defined according to

$$w = \begin{cases} (\beta, 1-\beta, 0)^T, & \tilde{\beta} >= 0 \\ (0, 1-\beta, \beta)^T, & \tilde{\beta} < 0 \end{cases} \tag{7.4}$$

The fixed keypoints of the dataset are represented with $\alpha = 0$ and $\beta = 0$. This way, the vector entries denote the mixing coefficients and therefore the importance of c_l, p_{inter} and c_r. The final representation of an arbitrary keypoint with n fixed keypoints is then $v \oplus w \in \mathbb{R}^{n+3}$, where $\oplus$ denotes the concatenation of the vectors.

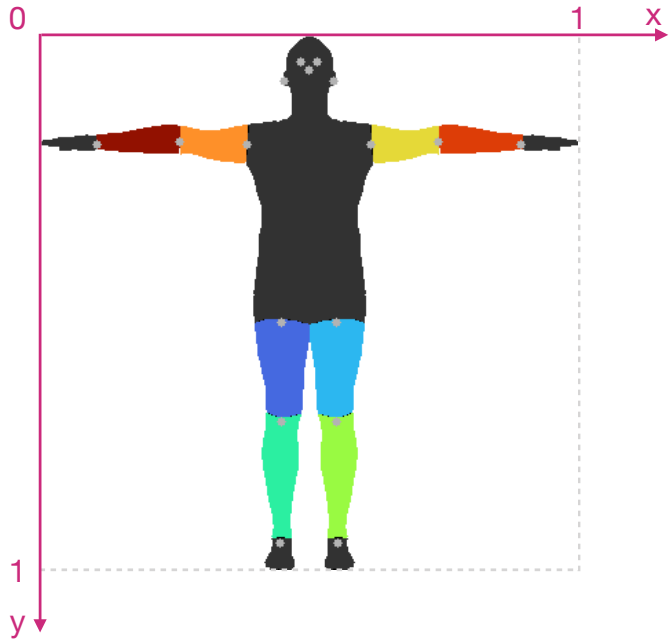


Figure 7.3: The used normalized pose depicted with the fixed keypoints from the COCO dataset. [53]

7.3.2.2 Normalized Pose

The normalized pose approach encodes the keypoints in normalized 2D-coordinates according to a normalized pose. Figure 7.3 visualizes the used normalized pose. The fixed keypoints from the COCO dataset are displayed in light gray. The body parts used in this section are colored, the rest of the body is visualized in black. During training, we generate ground truth keypoints p_{arb} as described in Section 7.3.1. We apply the same generation scheme based on the selected α, β, $\tilde{\beta}$, i, and j to the keypoints of the normalized pose. We obtain the coordinates p_{norm} from the resulting keypoints in the normalized pose, which are always in the interval $[0, 1]$ as they are normalized (see Figure 7.3). Hence, the normalized pose representation for each arbitrary keypoint is in $\mathbb{R}^2$.

7.3.3 Model Architecture

Our model architecture is closely related to the architecture in Section 6.3.2, which is again related to the TokenPose [48] architecture, but has important key modifications. Figure 7.4 visualizes the general architecture together with the adaption for the keypoint and thickness vectors, which will be explained later.

General architecture. At first, image features are extracted with a CNN. At the beginning of the Transformer, the feature maps are split into equally sized feature patches. The feature patches are embedded to visual tokens by a learned linear projection. This linear projection is applied independently to each feature patch. A 2D sine positional encoding is added to the visual tokens. Next, the keypoint tokens are appended to this

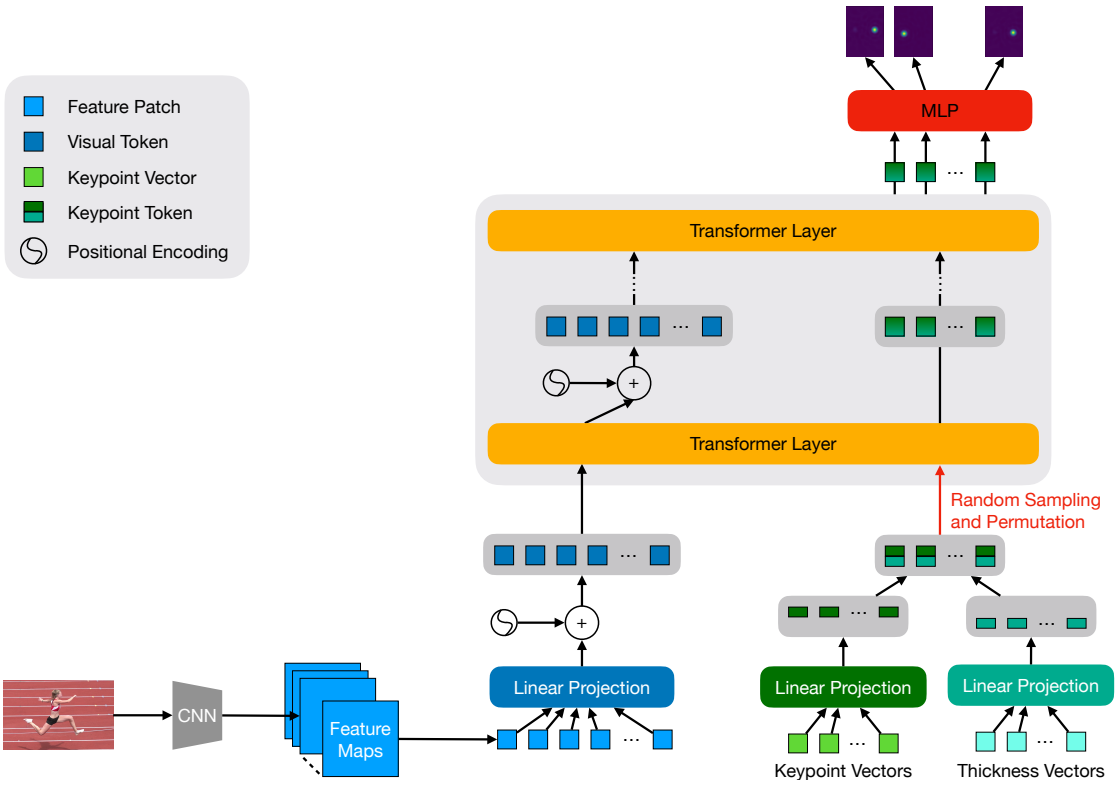


Figure 7.4: Model architecture with keypoint and thickness vectors, called *vectorized keypoints* approach. Image features from a CNN are split into patches and transformed to visual tokens via a linear projection. The linear projection is applied independently to all visual tokens. Keypoint and thickness vectors are also embedded via a linear projection, but only to half of the embedding size. Afterwards, they are concatenated to become the final keypoint tokens which are appended to the sequence of visual tokens. This sequence is the input to the Transformer network. Positional encoding is only added to the visual tokens. Random sampling and permutation applies only to the training phase.

sequence of visual tokens. The creation of these keypoint tokens is dependent on the representation type. We do not add positional encoding to the keypoint tokens as the order of the keypoints should not matter. In the end, a multi-layer perceptron (MLP) is used to transform the output of the Transformer corresponding to the keypoint tokens to 2D heatmaps.

Independent thickness tokens. In a first experiment, called *thickness token* approach in the following, we treat keypoint vectors and thickness vectors similar to feature patches. Both keypoint and thickness vectors are transformed to tokens of the full embedding size through two independently learned linear projections. Both keypoint and thickness tokens are then appended to the visual tokens. The problem with this approach is that the model is not able to match the corresponding keypoint and thickness

tokens without positional encoding. Therefore, it predicts the projection points p_{inter} instead of the desired points p_{arb}. We would need a positional encoding in order to match the tokens, but this is in contradiction to the desired independence of the order of the keypoints.

Combined keypoint and thickness tokens. Due to the mentioned problems, we use a different approach, which we call *vectorized keypoint* approach. Let m be the desired embedding size of the visual and keypoint tokens. Then, the keypoint vectors and the thickness vectors are embedded to tokens of size $\frac{m}{2}$ with two independently learned linear projections. These tokens are concatenated to the final keypoint tokens of size m, which combine the information from keypoint and thickness vectors. These keypoint tokens are appended to the visual tokens and then fed through the Transformer network. An illustration of this model can be found in Figure 7.4. During training, the tokens are first randomly sampled and permuted before being appended to the visual tokens, in the same way as described in Section 6.3.2.

Normalized pose tokens. The normalized pose coordinates are used similarly. In a first experiment, we embed them as well with a linear projection. However, experiments show that the performance is below the performance of the original TokenPose model. Therefore, we try to enhance the generated keypoint tokens by using an MLP instead of the linear projection in order to give the model more capacity to learn the keypoint semantics. This adaptation is visualized in Figure 7.5. The rest of the model is identical to the model for the vectorized keypoints approach (see Figure 7.4).

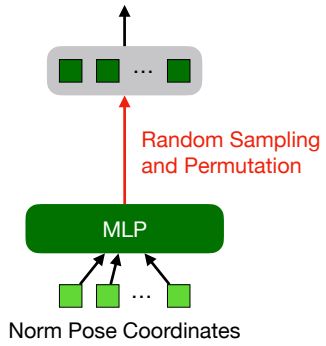


Figure 7.5: Model architecture adaptation for normalized pose representations. The normalized pose coordinates are transformed to the keypoint vectors via an MLP, which is applied independently to all coordinate pairs. Random sampling and permutation is only used during the training phase.

7.3.4 Thickness Metrics

Evaluations show that models predicting only the intermediate keypoints p_{inter} corresponding to the requested arbitrary keypoints achieve significant performance regarding typical metrics like the Percentage of Correct Keypoints (PCK) or the Object Keypoint Similarity (OKS) which are described in Section 2.1.2. A visual example for such a

Figure 7.6: Example predictions for the thickness token model. The predictions displayed in yellow are located only on the skeleton lines and do not consider the thickness of the body parts. This behavior motivates the need for the MTE and PCT metrics. Ground truth keypoints are displayed in red.

prediction can be found in Figure 7.6. The PCK and OKS metrics are based on the distance between the predicted keypoint and the ground truth keypoint. As the thickness of the limbs is relatively small, the distance between intermediate keypoints and desired points is also relatively small. This leads to a high performance regarding these metrics, although the model does not learn the semantic of the body part shapes. Therefore, we propose to use new metrics considering the thickness to measure the success of identifying arbitrary selected keypoints correctly. These metrics, called *Mean Thickness Error* (MTE) and *Percentage of Correct Thickness* (PCT), are described in Section 5.2.

7.3.5 Experiments

All our experiments use the TokenPose-Base [48] architecture configuration as a backbone. The CNN for feature extraction is an HRNet-w32 [92] pruned to its first three stages. We resize all input images to a size of 256×192 ($H \times W$). For the feature patches, we use the largest output feature maps of the HRNet, which are of size 64×48. These feature maps are split into patches of size 4×3, which results in 256 feature patches in total. We use 192 as the embedding size, equal to the TokenPose-Base implementation, and 12 Transformer Layers with 8 heads. As positional encoding, we use the same 2D sine encoding as used in TokenPose [48], which is added only to the visual tokens before each Transformer layer, hence before the calculation of keys, queries and values for the Self-Attention (see Figure 7.4). The MLP after the Transformer layers converts each output corresponding to the keypoint tokens to heatmaps of size 64×48. The final keypoint coordinates are retrieved with the DARK method [102].

7.3.5.1 COCO

Since OKS is the primary metric for COCO, we use this metric in our evaluations. Furthermore, we use the MTE and PCT metric with a threshold of 0.2 as described in Section 5.2 to measure the ability of the model to predict arbitrary keypoints at the right distance from the skeleton line. As the maximum error for the PCT metric is 2, we consider 0.2 as a good threshold for evaluations.

	Model	AP	AP^{50}	AP^{75}	AP^{M}	AP^{L}	AR	Avg PCK	Full PCK	MTE ↓	PCT ↑
1	TokenPose	**84.6**	**97.8**	**92.2**	**78.9**	**85.1**	**87.3**	84.1			
2	Thickness Tokens	82.8	**97.8**	91.0	76.9	83.3	85.8	83.0	71.0	79.2	6.3
3	Vectorized Keypoints (V1-VK)	84.0	**97.8**	92.1	78.3	84.3	86.7	**84.2**	**87.2**	**25.5**	**68.1**
4	Normalized Pose Linear	78.5	96.7	87.6	72.7	79.1	82.1	80.5	83.1	33.0	56.4
5	Normalized Pose MLP (V1-NP)	83.1	**97.8**	91.2	78.0	83.6	86.0	83.7	87.1	25.7	66.9

Table 7.1: OKS results, PCK@0.1 and thickness metrics results on our test set of the DensePose dataset. The Avg PCK is the PCK@0.1 metric on the fixed keypoints, the Full PCK the PCK@0.1 on the fixed and generated keypoints. MTE and PCT refer to the metrics proposed in Section 7.3.4. The TokenPose model is trained only on the fixed keypoints. The *thickness token* approach refers to the model with distinct tokens for thickness and keypoint vectors. The *vectorized keypoint* approach is described in Section 7.3.2.1. *Normalized pose MLP* refers to the approach with normalized pose representations and a four layer MLP for the embedding, *normalized pose linear* uses a linear projection.

Table 7.1 displays the results on the DensePose subset of the COCO dataset. The TokenPose baseline approach achieves the best results on the standard keypoints regarding the AP, but it is not capable of detecting arbitrary keypoints on human limbs (Table 7.1, first row). For the other proposed approaches, the focus is shifted from the standard keypoints to the arbitrary keypoints on the limbs, which is the reason for the small decrease in AP for OKS regarding the other approaches. The vectorized keypoint approach *V1-VK* achieves a slightly lower AP for OKS on the standard points, but the PCK for the standard keypoints is slightly higher and the PCK for all points including the generated keypoints (named *Full PCK* in Table 7.1) is even higher by absolute 3.0% (Table 7.1, third row). Figure 7.7 shows some qualitative results for the vectorized keypoint approach on the COCO dataset.

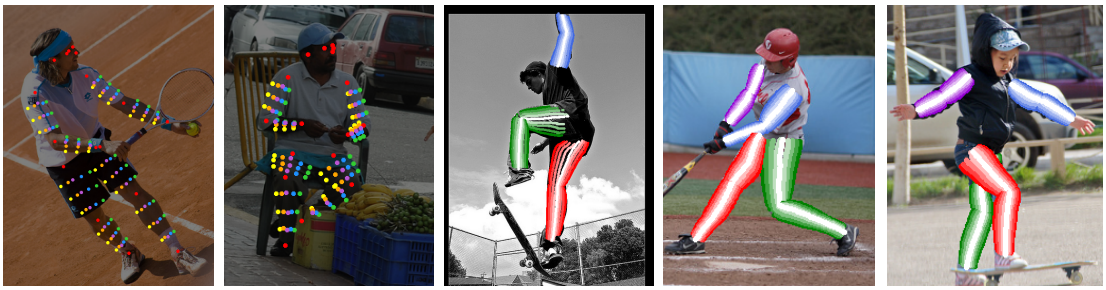


Figure 7.7: Examples for model predictions on the DensePose subset of the COCO dataset. The first two images show the fixed keypoints in red and a grid of three equally spaced keypoints along the skeleton line by five equally spaced keypoints along the thickness for each body part. The images are darkened for better visibility of the keypoints. The other three images show four equally spaced lines regarding the thickness on each body part. The skeleton line is colored white with a color gradient to the edges.

The full PCK for the approach with independent thickness tokens is lower than the full PCK for the vectorized keypoint model *V1-VK* by a large margin of absolute 16.2% (Table 7.1, second row). The reason is that the thickness token approach can not match the thickness tokens to the keypoint tokens as the Transformer is independent of the order of the input sequence. Figure 7.6 shows an example for this problem. Many keypoints lie in a distance from the ground truth that is valid for the PCK, therefore the full PCK is still quite high, although the detections completely fail the task. This is the reason why we propose the consideration of the MTE and the PCT, that we described in Section 5.2. These metrics measure if the model is able to estimate the keypoints at the right distance from the skeleton line to the body part boundary. Hence, these metrics can reveal such problems, which is not the case for PCK. For the thickness token approach, the MTE is 79.2, which is really high compared to the MTE of *V1-VK* with 25.5. The PCT metric makes the difference even clearer. Regarding the vectorized keypoint approach *V1-VK*, 68.1% of the detected keypoints are regarded as correct at a threshold of 0.2. This is over 10 times better than the PCT achieved by the thickness token approach.

Moreover, the normalized pose approach with single linear embedding achieves the worst results (Table 7.1, row four), but using a four layer MLP (model *V1-NP*) increases the AP by absolute 4.6% (Table 7.1, row five), which is only absolute 1.5% below TokenPose on the standard keypoints. Furthermore, the usage of an MLP improves all other metrics slightly, including the thickness metrics. Overall, the normalized pose MLP approach *V1-NP* performs slightly worse than the vectorized keypoint approach *V1-VK*, but yields similar results across all metrics.

7.3.5.2 BiSp-Jump v2

We further evaluate our method on the BiSp-Jump v2 dataset (see Section 5.3.3). In contrast to the DensePose split of COCO, this dataset does not contain segmentation masks. Therefore, we automatically generate segmentation masks with a trained model.

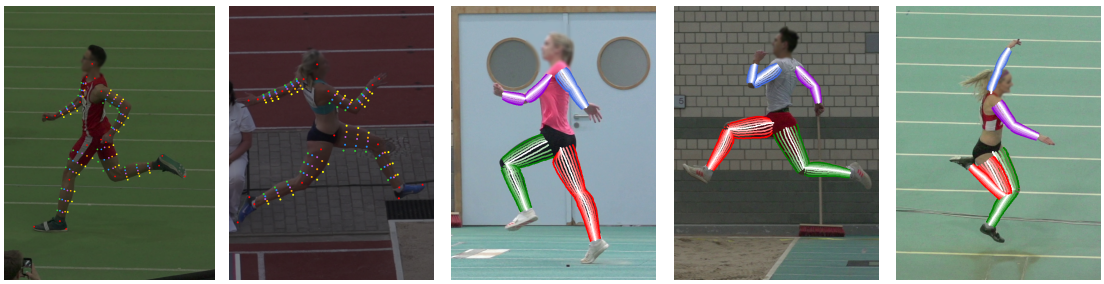


Figure 7.8: Qualitative results for the BiSp-Jump v2 test set. The first two images show the fixed keypoints in red and a grid of three equally spaced keypoints along the skeleton line by five equally spaced keypoints along the thickness for each body part. The images are darkened for better visibility of the keypoints. The other three images show four equally spaced lines regarding the thickness on each body part. The skeleton line is colored white with a color gradient to the edges.

We show in this section that these masks are sufficient to train a model for arbitrary keypoint detection on the limbs. Hence, there is no need to costly annotate sports datasets with body part segmentation masks in order to use our method. We use again the PCK metric with the distance between left shoulder and right hip as the torso size. Like before, we use $t = 0.1$, which corresponds to approx. 6 cm in this dataset. Additionally, we use the MTE and the PCT at a threshold of 0.2 to evaluate the thickness of the model's predictions.

The results for the BiSp-Jump v2 dataset are similar to the COCO results and displayed in Table 7.2. In comparison to the TokenPose model trained on the standard keypoints, the vectorized keypoint and the normalized pose approach achieve absolute 0.4% lower PCK on the fixed keypoints, but absolute 2.6% higher PCK if the generated arbitrary keypoints on the limbs are also considered. Compared to the DensePose COCO dataset, the vectorized keypoint model *V1-VK* achieves better results regarding the thickness of the limbs. The MTE is a third lower and the PCT is also a lot higher, absolute 13.3%. Furthermore, the difference in the performance between linear and MLP normalized pose approaches is lower. *V1-VK* also achieves the best results on this dataset, but the difference to the normalized pose MLP approach *V1-NP* is only marginal. In addition, Figure 7.8 visualizes some qualitative results for the BiSp-Jump v2 dataset, which proves that the model has learned a sense of thickness.

	Model	Avg PCK	Full PCK	MTE ↓	PCT ↑
1	TokenPose	**91.3**			
2	Vectorized Keypoints (V1-VK)	90.9	**93.6**	**16.2**	**81.4**
3	Normalized Pose Linear	90.3	93.5	17.0	79.0
4	Normalized Pose MLP (V1-NP)	90.9	**93.6**	16.8	79.8

Table 7.2: Recall values for the BiSp-Jump v2 test set in % at PCK@0.1. The first column displays the average PCK of the standard keypoints. The average PCK score including the generated points is given in the second column. The third column shows the MTE and the last column the PCT at threshold 0.2. The TokenPose model is trained only on the standard keypoints.

7.3.6 Summary

This section has proposed two representations for arbitrary keypoints on the limbs of humans. The first approach, called *vectorized keypoints*, represents each keypoint as a combination of the intermediate keypoint p_{inter} encoded in a keypoint vector and the thickness encoded in a thickness vector. The thickness indicates the distance of the desired arbitrary keypoint from p_{inter} to the body part boundary. The *normalized pose MLP* approach encodes the desired keypoint as 2D-coordinates on a normalized pose and uses a small MLP for the embedding to keypoint tokens.

Embedding both keypoint and thickness vectors independently and adding the resulting two tokens to the Transformer input sequence has led to the problem that the model

detected only the corresponding intermediate keypoints p_{inter}. This could be captured by low PCT scores, despite the quite high PCK scores. This proved the necessity of additional metrics like the MTE and PCT that we proposed in Section 5.2.

If the normalized pose is embedded not only with a linear layer but with an MLP, the normalized pose approach *V1-NP* achieved satisfactory results on both datasets. In comparison to the vectorized keypoint approach *V1-VK*, it performed slightly worse on all metrics. Quantitative and qualitative evaluations have shown that both proposed approaches can successfully detect arbitrary points on the limbs of humans. They achieved high PCT scores and low MTE values while maintaining high PCK (and OKS) scores on both the DensePose subset of the COCO dataset and the BiSp-Jump v2 dataset.

7.4 Arbitrary Keypoint Detection on Limbs and Skis with Sparse Partly Correct Segmentation Masks

Now that we have developed a method to detect arbitrary keypoints on limbs of humans in everyday life and for triple and long jump athletes, we want to extend our method to ski jumpers. A main challenge is that the segmentation masks for ski jumpers are not as accurate as for the other datasets. The masks that we are able to obtain automatically are sparse and only partly correct. Moreover, the skis are thin and can bend, which makes adjustments to our current model necessary. In this section, we describe the adaptations to our model and the evaluation on the ski jumping dataset.

7.4.1 Generation of Ground Truth Keypoints

We follow the strategy presented in the previous section to generate labels for arbitrary keypoints on the limbs. However, this method is not directly applicable for arbitrary points on skis because the straight lines between ski tips and ski tails do not necessarily lie entirely within the segmentation mask of the skis, depending on the perspective and the bending of the skis. As a consequence, p_{inter} might also lie outside the segmentation

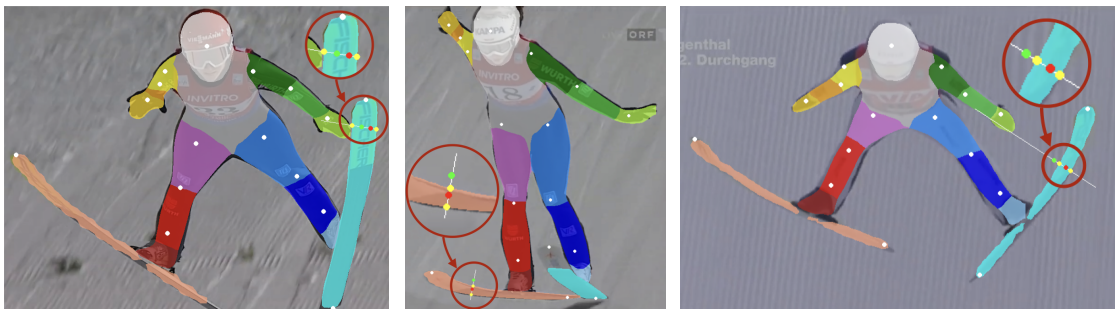


Figure 7.9: Examples for the ground truth keypoint generation process on skis. p_{inter} is visualized in green, the orthogonal line in white, c_l and c_r in yellow and the generated point in red. The middle and right image show that the intermediate keypoint p_{inter} can be located outside a ski mask.

mask of the ski. See the middle and right images of Figure 7.9 for examples. Hence, we randomly select a point p_{arb} between the intersection points with the segmentation mask boundary c_l and c_r, with a high probability that the point is located close to one of the intersection points and a lower probability that it lies in the middle, because evaluations show that it is harder for the model to learn the keypoints close to the boundary.

7.4.2 Keypoint Query Tokens

In order to represent arbitrary tokens on the segmentation masks, we use the *vectorized keypoints* approach *V1-VK* presented in Section 7.3.2.1 for the limbs, since it shows better results. Moreover, we adapt it for skis, too.

For a dataset with n keypoints, an intermediate keypoint p_{inter} corresponding to any arbitrary keypoint is encoded as a vector $v \in \mathbb{R}^n$. Let i, j be the indices of the keypoints enclosed by the corresponding body part or ski. Recall (see Eq. 7.3) that each intermediate keypoint p_{inter} can be formalized as $p_{inter} = \alpha \cdot p_i + (1-\alpha)p_j$ and v is created as

$$v_k = \begin{cases} 1-\alpha, & k = j \\ \alpha, & k = i \\ 0, & k \neq i \wedge k \neq j \end{cases} \qquad k = 1, ..., n \tag{7.5}$$

If a standard keypoint should be detected, it is either $\alpha = 1$ or $\alpha = 0$. The position of an arbitrary keypoint p_{arb} is now encoded relative to p_{inter}. We call the relative distance thickness. If p_{arb} is a point on a limb, it can be formalized as

$$p_{arb} = \beta \cdot p_{inter} + (1-\beta)c_{l/r}, \tag{7.6}$$

with $c_{l/r}$ being the intersection point on the same side of the skeleton line l_s as p_{arb}. If p_{arb} is a point on the skis, it can be formalized as

$$p_{arb} = \beta \cdot c_l + (1-\beta)c_r. \tag{7.7}$$

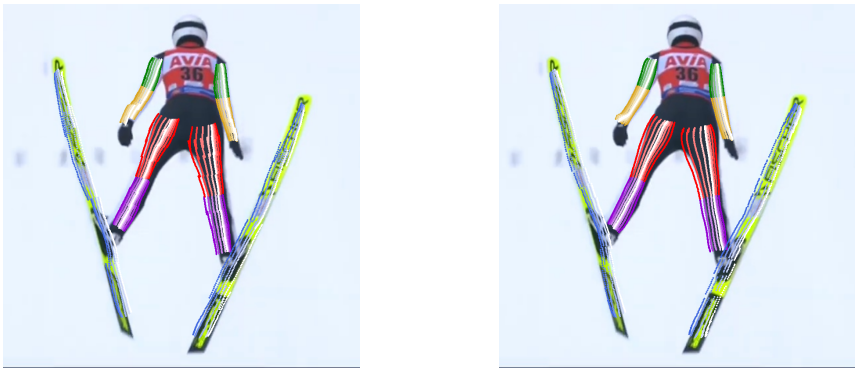
Furthermore, we define the thickness vector $w \in \mathbb{R}^3$ as

$$w = \begin{cases} (1-\beta, \beta, 0)^T, & p_{arb} \text{ is on limb} \wedge c_{l/r} = c_l \\ (0, \beta, 1-\beta)^T, & p_{arb} \text{ is on limb} \wedge c_{l/r} = c_r \\ (\beta, 0, 1-\beta)^T, & p_{arb} \text{ is on ski} \end{cases} \tag{7.8}$$

For standard points on limbs **and** skis, $(0, 1, 0)^T$ is used.

In this way, a keypoint vector v and a thickness vector w are designed for each desired keypoint p_{arb}. In the next step, as already described in Section 7.3.3, both vectors are converted via a learned linear projection to a vector of half of the embedding size used in the ViT. Then, v and w are concatenated to a single keypoint query token. All keypoint query tokens are appended to the sequence of visual tokens and then fed jointly through the Transformer network. A positional encoding is added only to the visual tokens before each Transformer layer. After the last layer, a small MLP with shared weights is used to convert the keypoint query tokens to heatmaps.

(a) Adapted att. (ours) **(b)** All points **(c)** 50 points



(d) 10 points **(e)** 50 points w/o sampling

Figure 7.10: Examples for model detections depending on the number of keypoint query tokens per model call. The images show four equally spaced lines regarding the thickness on each body part. For the limbs, the detected skeleton line is colored white with a color gradient to the edges. For the skis, the color gradient is from one side to the other. The keypoint query tokens are identical for all images. Image (a) is the result for our adapted attention, independent of the number of keypoint queries per model execution and without random sampling. Images (b) - (d) use the original full attention with random sampling like in Section 7.3.3, image (e) without random sampling during training. In image (b), all keypoints are computed with one model execution. In image (c) and (e), only 50 points are computed in one inference step and in image (d), 10 points are computed at once.

7.4.3 Attention Targets

The evaluations in Section 7.3.5 show that the method presented in Section 7.3.3 works well. During these evaluations, the number of keypoint query tokens used during inference is similar to the number of keypoint query tokens used during training. If a lot more tokens are used, the detection performance decreases. See Figure 7.10 for some examples. The model output for one keypoint query affects the other queries, which is an undesired behavior. The reason for this behavior is the attention mechanism. In

TokenPose, the attention of layer $i+1$ is calculated as

$$A(L^{i+1}) = softmax\left(\frac{L^i W_Q (L^i W_K)^T}{\sqrt{d}}\right)(L^i W_V), \tag{7.9}$$

where $L^i = (T^i_{vis}, T^i_{kp})$ are the visual and keypoint query tokens of the previous layer, $W_Q, W_K, W_V \in \mathbb{R}^{d\times d}$ are the learnable parameters to generate the queries, keys, and values and d is the dimension of the tokens. Hence, the attention is calculated between all tokens, so there is an information flow within the keypoint query tokens and from the keypoint query tokens to the visual tokens. Therefore, the keypoint query tokens have an influence on each other directly and through the visual tokens. In TokenPose, this is a desired behavior, as always the same keypoints are detected and the information of other keypoint tokens can help to detect occluded keypoints [48]. Section 7.3.5 shows that the detection performance is decreasing if the keypoint tokens corresponding to the standard keypoints are always present during training, but left out during inference. In that section, the solution is to include a random sampling and permutation of the keypoint query tokens, but this does not solve the problem of the undesired influence of the keypoint tokens on each other. A reason is that the learned attention weights still include an influence of keypoint tokens to other keypoint tokens, although the random sampling should prevent this. Therefore, we adopt the attention mechanism according to

$$\widehat{A}(L^{i+1}) = softmax\left(\frac{L^i W_Q (T^i_{vis} W_K)^T}{\sqrt{d}}\right)(T^i_{vis} W_V) \tag{7.10}$$

which is also visualized in Figure 7.11. The keypoint tokens serve only as the queries during the attention and the visual tokens as queries, keys, and values. This strategy is similar to the so-called cross-attention [9]. The information flow in the Transformer network is now restricted. Visual tokens can influence each other and keypoint tokens. A single keypoint token has an influence on its own next representation, but not on any other keypoint token nor on the visual tokens.

Hence, the position of a detected keypoint is only dependent on the image and independent of the other keypoints that should be detected at the same time. This is the desired behavior. Furthermore, the dimension of the softmax is now fixed to the number

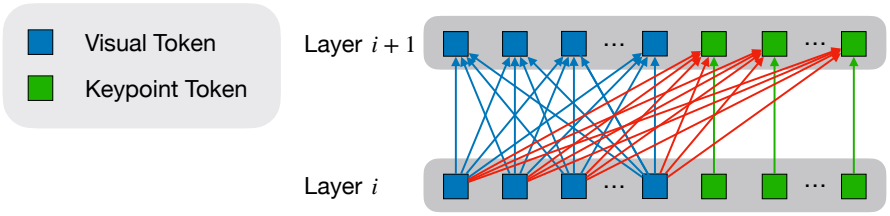


Figure 7.11: Visualization of the information flow in our adapted attention modules. The attention is computed such that only the visual tokens serve as keys and values. Hence, the visual tokens exchange information and keypoint tokens aggregate information from the visual tokens. The keypoint tokens do not influence each other and the visual tokens.

of visual tokens and independent of the number of keypoint tokens. Figure 7.10 shows that this adapted attention mechanism works as expected.

7.4.4 Training Strategies

As a baseline, we train a model with our adaptations on the images with available segmentation masks. As we only have a few masks available, and the training subset is thus small, this model underperforms regarding standard keypoints. Therefore, we evaluate different strategies for including the full dataset in the arbitrary keypoint training. One approach could be to fine-tune a segmentation model with our segmentation masks in order to generate segmentation masks for all images. However, this approach is infeasible as the segmentation masks of our dataset are only partially complete and only partially correct. Fine-tuning a segmentation model on these masks would not let the model learn useful masks. This is especially the case for skis because many images have annotations for only one, no ski, or only parts of the skis. For a direct training on arbitrary keypoints, this is not a problem. Arbitrary keypoints are only generated on available (possibly partial) segmentation masks. This does not deteriorate the model's performance. The only challenge are segmentation masks with incorrect borders, since this leads either to a wrong calculation of the intersection points $c_{l/r}$ and a mismatch of the thickness vector and the generated point, or to a generated point that does not lie on the limb/ski. However, our experiments show that the model can cope with this challenge and learns the correct points in most cases, because the number of correctly created points is by far larger than the number of false points.

7.4.4.1 Combined Training of Arbitrary and Standard Keypoints

The most straightforward approach is to use all available images for the training on the standard keypoints and the segmentation masks for the arbitrary keypoints. This strategy increases the performance on the standard keypoints a lot, but also deteriorates the ability to detect arbitrary points. Another technique includes the detection of intermediate keypoints p_{inter} as presented in Chapter 6. In order to generate intermediate keypoints, segmentation masks are not necessary. Hence, we can use all images for training on standard and intermediate keypoints and jointly train with arbitrary keypoints on the images with available segmentation masks.

7.4.4.2 Pseudo-Labels

The aforementioned approaches include the full training set during training, but the model still learns to detect arbitrary points only from a small subset. Therefore, we also experiment with pseudo-labels, since Chapter 4 shows their effectiveness. This means that we use a trained model in order to generate pseudo-labels of arbitrary points for all images without segmentation masks. With this strategy, it is possible to train a model on arbitrary keypoints with the whole training set. After convergence on the pseudo-labels, a fine-tuning is executed with arbitrary points generated from the available segmentation masks, because these ground truth keypoints are more precise than the pseudo-labels.

Another strategy is to add the pseudo-label training as a third part to the already described combined training approaches.

Looking at visualizations of the generated pseudo-labels reveals some errors. Therefore, our aim is to remove these wrong pseudo-labels. We observe that the network's scores have no direct relation to the correctness of the pseudo-labels. Hence, we use another technique to filter the labels. First, we obtain the model's predictions from the original image and some augmented variants. Second, we remove all keypoints with very low scores, since a very low score should indicate that a keypoint is not visible and augmentations like rotations might move the keypoint outside the augmented image. We remove all keypoints from the pseudo-labels with too few detections. Next, we transform the detections belonging to the augmented versions back to their location in the original image. Then, we calculate the standard deviation of these keypoints relative to the torso size. We use the standard deviation as the confidence measure instead of the network score. We select the pseudo-labels with the lowest standard deviation per body part, such that the number of pseudo-labels per body part is equal in the pseudo-label dataset. This approach is very similar to the approach presented in Section 4.3.1.2.

7.4.5 Experiments

Implementation details. The backbone for all experiments is TokenPose-Base [48] with three stages of an HRNet-w32 [92] for feature extraction like in Section 7.3.5. We crop the ski jumpers and resize all images to 256×192. Cropping is performed by creating the tightest bounding box containing all standard keypoints and enlarging its width and height by 20% to all sides. Visual and keypoint tokens are of size 192. We use 2D sine as positional encoding. After the Transformer layers, we use a two-layer MLP to obtain heatmaps of size 64×48 from the keypoint tokens. Keypoints coordinates are obtained from the heatmaps via the DARK [102] method. We pretrain our models with the COCO [50] dataset, either with TokenPose - only on the standard keypoints - or with the vectorized keypoints approach *V1-VK* using arbitrary keypoints on the limbs from Section 7.3.5. Additional to the model that is being trained, we keep an Exponential Moving Average (EMA) of the model's weights with an EMA rate of 0.99. The EMA model behaves like a temporal ensemble and achieves slightly better results than the original model. Therefore, we evaluate all experiments with the EMA model. The evaluation is performed on the single available ski jump dataset with segmentation masks, which is our Jump-Broadcast dataset described in Section 5.3.4. We evaluate on the test set with 560 images in total and 81 images with segmentation masks. We generate 200 arbitrary keypoints with a fixed seed for each image during the arbitrary keypoint evaluation, resulting in 16,200 total keypoints.

Evaluation metric. We evaluate using the PCK@0.1 with the distance from right shoulder to left hip as the torso size. For this dataset, this threshold corresponds to approx. 6 cm. As described in Section 7.3.5, the PCK is not sufficient to measure if the model predicts the thickness of the arbitrary points correctly. A model predicting only the corresponding intermediate keypoints p_{inter} would achieve a high PCK score although the thickness might be wrong, because the intermediate keypoints are close

	Method	Pre.	Std.	Seg.	Inter.	PL	Std. PCK	Full PCK	MTE ↓	PCT ↑
1	TokenPose	Std.	✓				77.2			
2	TokenPose	VK	✓				75.4			
3	V2-B	VK		✓			52.7	88.1	18.2	**77.7**
4	Std. & Seg. (V2-Std)	VK	✓	✓			77.1	90.1	18.3	76.6
5	Seg. & Inter. (V2-Inter)	VK		✓	✓		76.5	**91.8**	**17.5**	**77.7**
6	Seg. & Inter.	Std.		✓	✓		**77.8**	91.5	18.0	76.4
7	all PL	VK				all	76.3	90.4	18.7	76.0
8	fine-tune all PL	VK		✓		all	76.3	90.4	18.9	75.2
9	all PL & Std. & Seg.	VK	✓	✓		all	76.4	90.9	19.2	74.7
10	all PL & Inter. & Seg.	VK		✓	✓	all	76.9	91.0	18.4	75.6
11	80% PL	VK				80%	76.3	90.8	18.7	74.8
12	fine-tune 80% PL	VK		✓		80%	76.1	91.3	18.3	75.8
13	80% PL & Std. & Seg.	VK	✓	✓		80%	77.3	90.7	19.4	73.4
14	80% PL & Inter. & Seg.	VK		✓	✓	80%	76.7	91.4	18.1	75.3

Table 7.3: Recall values (precision is 1) for the Skijump-Broadcast test set in % at PCK@0.1. The second column labeled *Pre.* displays the pretraining, *Std.* refers to the pretraining with the standard keypoints, *VK* to the pretraining with the vectorized keypoints approach *V1-VK*, both on the COCO dataset. The third table section shows the used training steps. *Std.* means training on the standard keypoints, usable on the whole training set. *Seg.* refers to the training on arbitrary keypoints with available segmentation masks. *Inter.* stands for training on the intermediate keypoints which is also usable on the whole training set and *PL* refers to the pseudo-labels, whereby either all pseudo-labels are used or the 80% with the least standard deviation during filtering. The first column of the last table section displays the average PCK of the standard keypoints, evaluated on the test set containing images with and without segmentation masks. The average PCK score including the arbitrary points is given in the second column, the third column shows the MTE and the last column the PCT at threshold 0.2. These scores are evaluated on the test set with available segmentation masks.

enough to the ground truth points. Therefore, like in Section 7.3.5, the Mean Thickness Error (MTE) and the Percentage of Correct Thickness (PCT) are used as additional metrics.

Results. Table 7.3 displays the results for all experiments. For the TokenPose approach, we evaluate two pretrainings. The pretraining on the COCO dataset with the standard keypoints achieves a better standard PCK than the pretraining with the vectorized keypoint approach from Section 7.3 on the COCO dataset (see Table 7.3, first and second row). This is expected, as TokenPose detects only the standard keypoints of the skijump dataset.

Furthermore, we use the *vectorized keypoints* approach with a generation of 5 to 50 arbitrary keypoints for each image and with the improved attention mechanism described in Section 7.4.3. We call this approach *V2-B*. It achieves good results for the full PCK, the MTE and the PCT, but the standard PCK decreases by absolute 24.5% in comparison to TokenPose (see Table 7.3, third row). Therefore, we use the combined strategies like described in Section 7.4.4.1 to improve the standard PCK. In the first experiment

Figure 7.12: Qualitative examples for model detections. The images show four equally spaced lines regarding the thickness on each body part. For the limbs, the skeleton line is colored white with a color gradient from it to the edges. For the skis, the color gradient is from one side to the other. The model from experiment *Seg. & Inter.* is used to generate the images.

called *V2-Std* (*Std. & Seg.*, row four in Table 7.3), we alternately train on the standard keypoints and the arbitrary points. This leads to nearly the same PCK on the standard keypoints, but the PCT is absolute 1.1% lower than for *V2-B*. Training with the intermediate keypoints instead of the standard keypoints (experiment *Seg. & Inter.*, rows five and six in Table 7.3) leads to better results. With the vectorized keypoint pretraining, it achieves the same PCT as *V2-B*, while the PCK decreases only slightly compared to TokenPose. We call this approach *V2-Inter*

The results of *V2-Inter* seem promising. Therefore, we evaluate this strategy with two pretrainings, the vectorized keypoint pretraining and the standard keypoints pretraining. With the standard keypoints pretraining, the standard PCK is even higher than the standard PCK for the TokenPose approach which is trained only on the standard keypoints. But all other metrics decrease for this experiment. Therefore, we focus on the vectorized keypoints pretraining for all other experiments.

Hence, we use *V2-Inter* which is based on the vectorized keypoints pretraining in order to generate *pseudo-labels*, because it achieves the best results regarding the thickness metrics. We generate 1000 pseudo-labels for each image in advance and select 25 of them randomly in each training step. The results of the pseudo-label training with all pseudo-labels are slightly worse than the results of the other experiments (see Table 7.3, row seven). As we did not use the existing segmentation masks but only the pseudo-labels during that experiment, we execute a fine-tuning with these afterwards. We choose the best weights of the pseudo-label training and fine-tune with the segmentation annotations in order to improve the results, but unsuccessful (see Table 7.3, row eight). From the validation curve, we observe a decrease in the standard PCK from step to step. Therefore, we consider a combined training in the next experiment, training alternately on the arbitrary keypoints, the pseudo-labels, and the standard keypoints (experiment *all PL & Std. & Seg.*, row nine in Table 7.3) or the intermediate points (experiment *all PL & Inter. & Seg.*, row ten in Table 7.3). Training with the standard keypoints achieves lower scores for all metrics in this case, also for the standard PCK.

Including pseudo-labels in the training process did not lead to better results. A look at the quality of the generated pseudo-labels shows that some are wrong. Therefore,

we repeat the pseudo-label experiments with the best 80% of the labels. We use a filtering technique based on the standard deviation of the detected keypoints for multiple, differently augmented images described in Section 7.4.4.2. The augmentations that we use are horizontal flipping, 45° rotation (clockwise and counterclockwise) and scaling between 65% and 135%. We expected better results with more correct labels, but the results are similar (see rows 11–14 in Table 7.3).

All these experiments show that it is most important to have more images to train on. In our case, including pseudo-labels does not increase the number of images, because we can use all images already by training on standard keypoints or intermediate keypoints. Using the intermediate keypoints (experiment *V2-Inter*) results in the best scores because they are more similar to the desired arbitrary keypoints in comparison to the standard keypoints (experiment *V2-Std*). Pseudo-labels, no matter if filtered or not, do not increase the scores. Figure 7.12 shows some example predictions for different poses for the best performing experiment *V2-Inter*.

7.4.6 Summary

This section has proposed a method to detect arbitrary keypoints on the limbs and skis of ski jumpers. Since the segmentation masks in our datasets are only partly correct, many of them contain no or only one ski segmentation mask. Therefore, we could not fine-tune a segmentation network in order to generate segmentation masks for all other images. But for our method, this is not a problem, since keypoints are only generated on the available segmentation masks. Problematic are only segmentation masks with wrong borders.

The proposed approach is based on the vectorized keypoint approach *V1-VK* presented in Section 7.3. For the keypoints on the skis, we modified the technique because the intermediate keypoints p_{inter} do not necessarily lie in the middle of the skis. Therefore, we do not include the line of intermediate points and only use the intersection points with the segmentation mask. The evaluation metrics for the thickness are adapted accordingly.

We further adapted the attention mechanism so that the keypoint query tokens do not influence each other or the visual tokens. In this adjusted setup, only the visual tokens are correlated with each other and with the keypoint query tokens. Evaluations have shown that this adaptation, *V2-B*, successfully resolves the issue of performance degradation when using more keypoint query tokens than during training.

Moreover, we have experimented with different training strategies on both the segmentation mask dataset and the full dataset. Our results have shown that training a model *V2-Std* jointly on arbitrary and standard keypoints significantly improves the standard PCK compared to the base model *V2-B*, though it leads to a deterioration in PCT and MTE. Alternatively, training on the intermediate points instead of the standard keypoints (model *V2-Inter*) yields better results for PCT and MTE.

Hence, the model proposed in this section is capable of detecting arbitrary keypoints on the limbs and skis of ski jumpers. Moreover, it can be trained using only a few partly correct segmentation masks.

7.5 Detecting Arbitrary Keypoints on the Whole Body

In the last chapters, we introduced model variants to detect arbitrary keypoints on the limbs of humans and further on skis. We implemented strategies to train those models based on segmentation masks, but the masks can also be only partly correct, and it is sufficient to have only a few masks. In this chapter, we extend the approach to detect arbitrary keypoints on the whole body of humans and apply it to athletes performing triple, high, and long jump. Apart from the limbs, we now include the head, torso, hands, and feet. Moreover, we improve the keypoint detection on the limbs such that arbitrary keypoints on bend elbows and knees are detected correctly. We use the same TokenPose based architecture as in the last chapters. To be precise, we use the architecture with the adapted attention mechanism *V2-B* explained in Section 7.4.3. In this chapter, we evaluate two approaches for the arbitrary keypoint queries, similar to Section 7.3.2. To add more body parts and improve the detection of arbitrary keypoints on the limbs, we introduce new and enhance existing techniques to generate ground truth keypoints, which we will explain in the following. [1]

7.5.1 Ground Truth Generation

For the limbs, we follow the strategy described in Section 7.3 to generate ground truth keypoints. We quickly recap this method so that we can extend and reference it afterwards. Consider a body part which is enclosing two standard keypoints, as it is the case for the limbs. At first, we draw a vector from one standard keypoint to the other enclosing standard keypoint, which represents a directed skeleton line and which we will call v_s in the following. Second, an orthogonal line l_o to v_s is created and the furthest points c_l and c_r that lie on l_o and the segmentation mask are retrieved. In this chapter, contrary to the last chapters, c_l is always located on the left side relative to the orientation of v_s and c_r on the right side. Arbitrary keypoints are then located on the line between the intersection points c_l and c_r. We incorporate the same technique for the feet, using the toe tip and the heel as the keypoints that are enclosed by the body part. For the torso, we use the neck as the first enclosed keypoint and the virtual keypoint in the middle of the hip keypoints as the second one. Regarding the hands, this technique has the drawback that it only detects keypoints that lie between the enclosed keypoints. If we use the hand and wrist keypoint as enclosed keypoints, our model would not be able to detect the fingertips. Since the fingertips are not annotated, an alternative strategy is required, which will be explained in Section 7.5.1.1. The same applies to the head, where we use both this strategy and an additional one (see Section 7.5.1.2).

Since our segmentation masks are automatically generated, many of them contain errors. We remove the masks with large errors by hand, but keep many masks with small errors like having two left hands and no right hand (see the first image of Figure 5.5). Another error that we observe very often is a left/right confusion at the boundary of body parts, especially the legs, meaning that at the boundary of the left body part,

[1] The code for this chapter is publicly available on GitHub: `https://github.com/kaulquappe23/all-keypoints-jump-broadcast`.

a small stripe is labeled as the right body part and vice versa. This can be seen in the fourth and fifth image of Figure 5.5. These errors lead to the problem that the furthest intersections $c_{l/r}$ of l_o can be located at a completely wrong position. We overcome this problem by determining c_l and c_r as the last points that lie both on l_o and the segmentation mask before leaving a coherent area of the segmentation mask, viewed from the intersection of v_s and l_o.

7.5.1.1 Edge Body Parts: Head, Hands, and Feet

For body parts that are located at the edge of the human body, like the head, hands, and feet, the respective body part is not always enclosing two standard keypoints or the enclosed keypoints are not suitable for generating arbitrary keypoints on the whole body part: If the used standard keypoints lie on the edge of the body part, the described generation mechanism can create arbitrary keypoints on the whole body part. For the head, hands, and feet, we need to adapt the generation strategy if such keypoints are not available. In our dataset, such keypoints are available for the feet (toe tip and heel), but not for the hands and the head. We describe the generation strategy for the head and the hands in the following, using the head as an example. The strategy for the hands is analogous and can also be used for the feet if necessary for other datasets.

Standard keypoints that are enclosed by the head are the neck and the head keypoint itself. The head keypoint is located in the middle of the head in our datasets. Therefore, we cannot use it, since we would only be able to detect keypoints on the lower half of the head in this case. Hence, we use the following strategy, which is also visualized in Figure 7.13 on the left. At first, the vector v_s through the neck and head keypoint is created and extended to the head top p_{head_top} and the chest p_{chest}, visualized with orange dots in Figure 7.13 on the left. We select a random intermediate point p_{inter} on v_s (visualized with a green dot in Figure 7.13), draw the orthogonal line l_o to this line, determine the

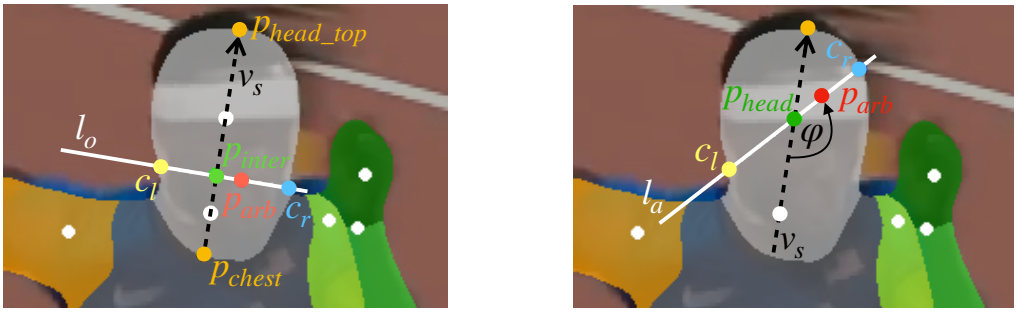


Figure 7.13: Two versions of generating ground truth keypoints on the head. On the left, the line through the head and neck keypoint is extended to the head boundary (orange), an orthogonal line to this line is drawn (white) and a keypoint on this line between the boundary points of the segmentation mask (blue and yellow) is chosen (red). On the right, a line (white) rotated around the head keypoint (green) at a random angle is generated and a keypoint on this line between the boundary points of the segmentation mask (blue and yellow) is selected (red).

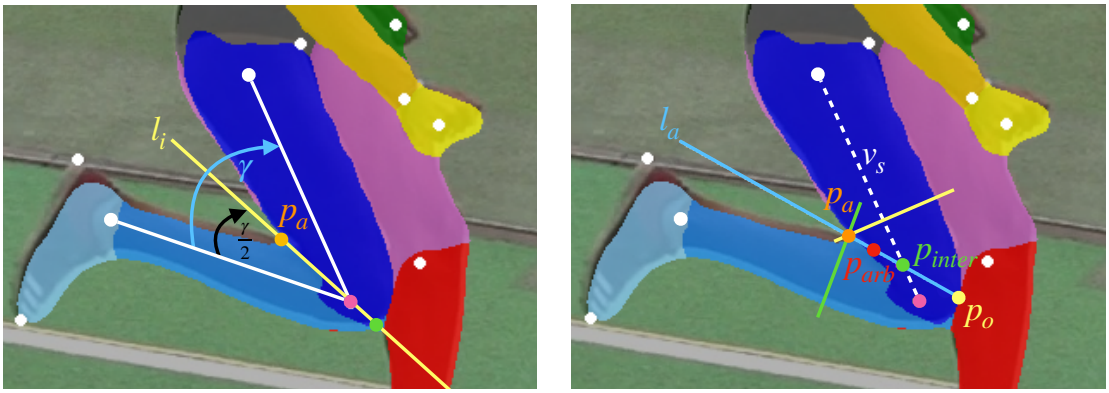
intersection points c_l and c_r with the boundary of the segmentation mask and select a random point p_{arb} between p_{inter} and $c_{l/r}$. Since we extend v_s to the boundary of the body part, we call this the *extension* method. In case of the hands, we use the wrist and hand keypoint as enclosed keypoints.

7.5.1.2 Head

Since the head is rather round, the described creation process seems counterintuitive close to the head top. The distance between c_l and c_r is decreasing and even approaching 0, the closer p_{inter} is to p_{head_top}. Hence, we propose a second strategy to generate keypoints on the head. We use the head keypoint as the center, choose a random angle $\varphi \in [0, 2\pi)$ and rotate a line l_a counterclockwise around the center according to φ, while $\varphi = 0$ corresponds to v_s. Then, we continue as before with detecting c_l and c_r as the intersection of l_a with the boundary of the segmentation mask and randomly choosing a point p_{arb} between the head keypoint and $c_{l/r}$, which is visualized in Figure 7.13 on the right. c_r is used for angles $\varphi \in [0, \pi)$ and c_l for $\varphi \in [\pi, 2\pi)$. Since we rotate for a certain angle, we refer to this as the *head angle* method.

7.5.1.3 Bent Limbs: Elbows and Knees

With the keypoint generation strategy on the limbs as used in Sections 7.3 and 7.4, arbitrary keypoints on the upper arm, forearm, thigh, and lower leg could be detected if they lie in between the enclosed keypoints. This works well in most cases, but not if the limbs are strongly bent, which happens frequently in the case of triple, high, and long jump. Therefore, we adapt the generation strategy to fit also the case of bent elbows and knees. For that purpose, we determine the point on the inner side of the bent joint at first, which we call the *anchor point* or p_a in the following. To retrieve that point, we calculate the bending angle γ of the joint, which is the angle enclosed by the line through the hip and the knee and the line through the knee and the ankle, or the angle enclosed by the line through the shoulder and the elbow and the line through the elbow and the wrist. Then, we generate a line l_i through the knee or elbow keypoint with half of the bending angle. We determine the intersections of l_i with the boundary of the segmentation mask and set the anchor p_a as the point on the side with the acute angle. The other point on the opposite side will be called p_o in the following. We unify the segmentation mask for the lower and upper body part in this case (see Figure 7.14a for details). In the next step, as visualized in Figure 7.14b, we rotate a line l_a for a random angle around the anchor point, starting in a 90° angle to the upper body part (yellow line in Figure 7.14b) and stopping at a 90° angle to the lower body part (green line in Figure 7.14b). We determine the intersection point p_{inter} of l_a with the vector v_s through the enclosed keypoints of the corresponding body part (upper or lower one) and chose a random final keypoint p_{arb} either between p_{inter} and p_a or between p_{inter} and p_o. We do not discriminate between the segmentation mask for upper and lower body part in the whole process since the boundary is often detected rough, and it is somehow



(a) Anchor generation **(b)** Keypoint generation

Figure 7.14: Visualization of the keypoint generation process on bent limbs. (a) Visualization of the anchor generation at a bent knee. We generate a line with half the bending angle γ through the knee keypoint, visualized in yellow. Then, we determine the intersection of that line with the segmentation mask boundary (orange and green) and select the anchor point within the acute angle, so the anchor is the orange point in this image. (b) Visualization of the keypoint generation on a bent knee. The anchor point is visualized in orange, the line l_a in blue. The intersection point p_{inter} is colored green and the point p_o yellow. The final keypoint is visualized in red. The line l_a always lies somewhere between the yellow and the green line.

not clearly defined where the thigh/upper arm ends and the lower leg/forearm begins. With this strategy, arbitrary points on bent limbs can be generated.

7.5.2 Query Encoding

With the described strategies, it is possible to generate arbitrary ground truth points on the whole body of humans. In this section, we analyze the same two approaches as in Section 7.3 to encode the desired keypoint in the token domain. In the following, we describe how we include the head, torso, hands, and bent limbs in the approaches.

7.5.2.1 Keypoint Vectors

The *keypoint vector* approach uses multiple vectors to encode the desired keypoint. For the limbs, it is explained in detail in Sections 7.4.2 and 7.3.2.1. We quickly recap this mechanism to easily explain its extensions. Let the dataset contain n keypoints, then the first vector called *keypoint vector* has dimensionality n and the second vector called *thickness vector* has dimensionality 3. Let p_i, p_j be two keypoints enclosed by a body part and p_{inter} be the orthogonal projection of the desired keypoint on the skeleton line v_s between p_i and p_j. Then $p_{inter} = \alpha \cdot p_i + (1-\alpha) \cdot p_j$. We set the entries belonging to p_i and p_j in the keypoint vector to α and $1-\alpha$, respectively. All other entries are 0. The thickness vector is created similarly. Let the final keypoint be $p_{arb} = \beta \cdot c_{l/r} + (1-\beta) \cdot p_{inter}$, then we set the middle entry of the thickness vector to $1-\beta$ and either the first or the last

entry to β, depending on whether c_l or c_r is chosen. See Equation 7.8 for details. This technique ensures that flipping the input image horizontally corresponds to a flipping of the thickness vector and switching the corresponding entries in the keypoint vector for left and right keypoints.

For the torso, we set the keypoint vector entry belonging to the neck to α and the entries belonging to the left and right hip joint to $0.5(1-\alpha)$, since we use the middle of the two hip keypoints as the reference point. In case of the extension method for the head, where we extend the line through the reference keypoints p_{head} and p_{neck} to the points p_{head_top} and p_{chest}, $p_{inter} = \alpha \cdot p_{head\_top} + (1-\alpha) \cdot p_{chest}$. We set the entries in the keypoint vector in the same way as before, meaning that the keypoint vector entry corresponding to p_{head} is set to α and the entry corresponding to p_{neck} to $(1-\alpha)$. We do not introduce new entries for the keypoints p_{head_top} and p_{chest}. The same applies to the hands with their respective reference keypoints. The thickness vector is created as explained before.

If the angle method is used for the head, this approach is not applicable as the creation logic is completely different. Therefore, we introduce a third vector called the *angle vector*. It has just one entry which is set to 0 in case of all other body parts, meaning that it is not used. For the head, its value q indicates the rotation angle φ in percent, starting with the line through head and neck keypoint and rotating counterclockwise around the head keypoint as described in Section 7.5.1.2. This means that the lines l_a for rotation angle percentages $0 < q <= 0.5$ and $0.5+q$ are identical. They differ in their corresponding intersection point. c_r is used for percentage q and c_l for $0.5+q$. The final keypoint can now be calculated as $p_{arb} = \beta \cdot c_{l/r} + (1-\beta) \cdot p_{head}$, and we set the thickness vector to $[0, 1-\beta, \beta]$ no matter which intersection point is used as this information is already encoded in the angle vector. We further set the entry of the keypoint vector corresponding to the head keypoint to 1.

For elbows and knees, we adapt the encoding slightly. In these cases, p_{inter} is not projected perpendicular (along l_o) on the line between p_i and p_j, but along the line l_a, and we further use l_a to determine the intersection points c_l and c_r and the thickness percentage β.

7.5.2.2 Normalized Pose

The second approach involves a normalized human pose, similar to the approach presented in Section 7.3. The normalized human pose contains the standard keypoints and the body part segmentation masks of a human in a T-shaped pose, visualized in Figure 7.15. In contrast to Section 7.3.2.2, we use feet turned outwards and hands downwards such that the maximum area of these body parts is visible in the segmentation masks of our T-shaped pose template. Each desired keypoint is now represented with the normalized x- and y-coordinates of this pose. Normalized coordinates mean that the upper left corner has coordinates $(0, 0)$ and the lower right corner $(1, 1)$. Hence, each keypoint is characterized by only two values in this encoding approach, no matter to which body part it belongs.

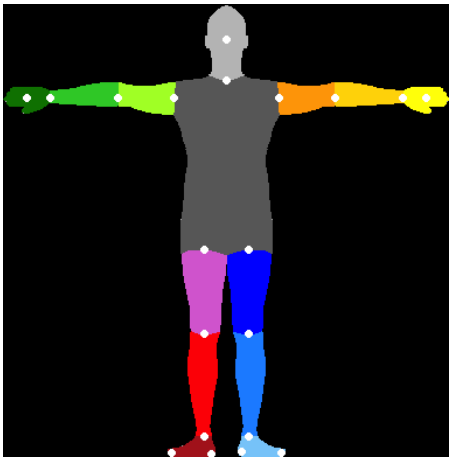


Figure 7.15: Visualization of the normalized pose. All used body parts are colored and the fixed keypoints are visualized in white.

7.5.3 Keypoint Token Embedding

Each keypoint encoding, no matter which of the three described options is used, needs to be converted to one single token of the same size as the visual tokens in order to be compatible with the ViT. This process is called *embedding*. The straightforward way is to use a linear layer to convert a vector of arbitrary length to a vector with the desired length. In case of the keypoint vector approach, we have more than one vector in the encoding. Hence, we have the option to concatenate the vectors to one vector at first and then embed them, or to embed them at first to tokens with a half or a third of the embedding size and concatenate them afterwards. Furthermore, apart from using a single linear layer, we try to enhance the embeddings by adding more layers combined with ReLU operations, hence incorporate a small MLP.

7.5.4 Experiments

As in the previous sections, the base architecture for all our experiments is TokenPose-Base [48] with an input image size of 192×256, an embedding dimension of 192 and a conversion of the final embeddings to heatmaps of size 48×64. Athletes are cropped before the images are fed through the network. We pretrain our model on the COCO dataset [50] and fine-tune it on our individual sports related datasets. Random flipping, random rotation of up to $45°$, random scaling in the range of $[0.65, 1.35]$, and color jitter are applied as augmentation techniques during training.

7.5.4.1 Evaluation Scheme

We use the Percentage of Correct Keypoints (PCK) and the Percentage of Correct Thickness (PCT) as evaluation metrics. We use PCK thresholds of $t = 0.1$ and $t = 0.05$

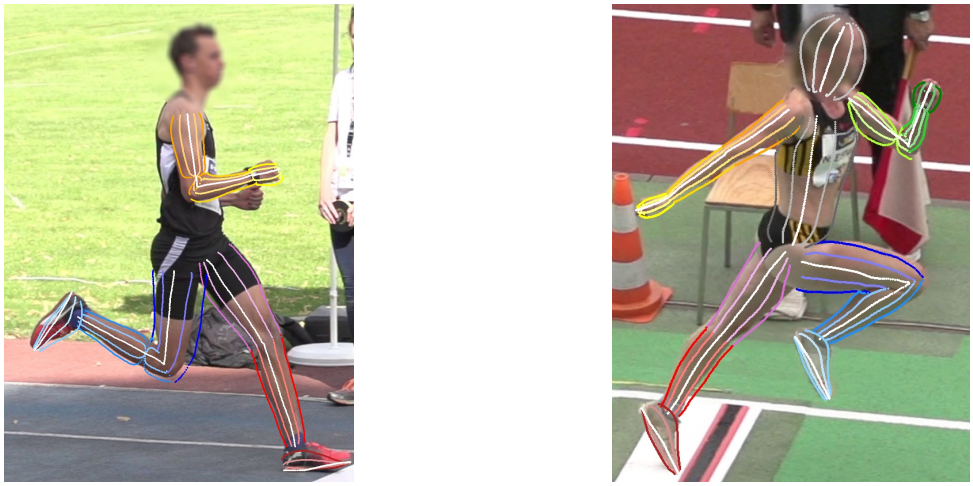


Figure 7.16: Collapsing points around the right elbow and knee (first image) and the left elbow (second image) in the case that c_l and c_r swap sides between the lower and upper body part.

in our evaluations, which corresponds to approx. 6 cm and 3 cm, respectively. For PCT, we use $t = 0.2$ as before.

In Section 7.3, we use a random number of up to 50 freely chosen keypoints per image during evaluation. In Section 7.4, we increase the evaluation strategy to a fixed number of 200 randomly created keypoints. In this section, we aim to evaluate the full range of possible keypoints together with the different encoding and embedding strategies for certain body parts. Hence, on each body part, we create a specific set of evaluation keypoints: we use thickness $0, 0.5$ and 1 and create 25 equally spaced keypoints with these thicknesses to both sides of the body part, resulting in 125 keypoints per body part. During evaluation, we treat elbows and knees as individual body parts, too. Hence, if all body parts are completely visible, this results in a total number of 2250 keypoints per image. The benefit of this strategy is that it is reproducible and hence provides a comparable metric for future work. It also weights all body parts equally and allows a separate comparison of the results for each body part.

7.5.4.2 Collapsing Elbow and Knee Keypoints

Elbows and knees connect the upper arm with the forearm and thigh with the lower leg, respectively. Including them in our method enables retrieving continuous keypoints along the whole arms and legs. Especially in these cases, it is very important to interpret v_s as a vector and define c_l and c_r relative to its direction.

This ensures that c_l and c_r are always located on the same side of the upper body part (thigh or upper arm) and the lower body part (lower leg or forearm). Without this method, c_l and c_r could swap the side between the lower and upper body part. This leads to an unwanted collapse of the keypoints around the swapping location, visualized

Enc.	E	C	PCK@ 0.1	PCK@ 0.05	PCT@ 0.2	head	torso	u.arm	elbow	f.arm	hand	thigh	knee	l.leg	foot
V3-VE	1	1	94.1	82.1	69.9	81.0	76.0	77.7	66.1	67.9	55.1	82.2	71.7	77.7	49.9
V3-VE	2	1	94.2	82.3	71.0	82.0	76.6	78.7	68.2	68.6	56.7	82.7	71.9	77.9	**52.4**
V3-VE	2	0	94.0	82.0	70.3	81.6	74.9	78.1	67.2	68.4	55.4	82.2	71.8	78.3	51.3
V3-VE	3	1	94.2	82.4	71.0	81.6	76.6	78.4	68.0	68.9	**57.1**	82.7	71.9	**78.6**	52.2
V3-VA	1	1	94.3	82.6	71.6	94.4	75.9	78.9	68.3	68.5	55.7	82.7	71.9	78.0	51.2
V3-VA	2	1	94.3	82.5	71.6	92.8	76.1	78.8	68.0	**69.1**	56.3	82.3	71.9	78.2	51.0
V3-VA	2	0	**94.5**	**82.7**	**71.9**	93.2	**76.9**	**79.0**	**68.4**	**69.1**	56.2	**82.8**	71.8	78.3	52.3
V3-VA	2	2	94.3	82.2	71.5	92.5	76.5	78.5	68.0	68.8	56.4	82.4	**72.0**	77.8	51.6
V3-VA	3	1	94.3	82.4	71.5	94.4	76.0	78.4	67.9	68.4	56.1	**82.8**	71.4	78.0	51.2
V3-NP	1	-	92.5	79.7	66.6	**94.6**	72.2	73.6	63.2	63.4	48.9	81.7	70.8	73.7	33.6
V3-NP	2	-	92.6	78.9	65.5	94.2	71.2	73.2	62.8	63.0	48.8	81.1	70.0	72.3	27.6
V3-NP	4	-	92.5	79.6	66.8	94.5	73.0	73.4	63.3	62.5	48.0	81.7	71.3	75.4	34.8

Table 7.4: Recall values for the BiSp-Jump v2 test set in % at PCK@0.1 and PCK@0.05 and PCT@0.2. These scores are evaluated on the test set with the fixed keypoints and on the described 125 keypoints per body part as far as these keypoints exist in the image. The first column indicates the used encoding: *V3-V* indicates the keypoint vector approach, either with the head extension strategy *V3-VE* or the head angle strategy *V3-VA*. *V3-NP* refers to the normalized pose approach. The second column (labeled E) names the total number of layers used for the embedding and the third column (labeled C) the number of layers that is executed before concatenation of the vectors. Hence, C denotes how many layers of E are executed before the concatenation. The third table section contains the average metric results over all keypoints in the test set, the fourth section lists the PCT scores separately for each keypoint type, left and right keypoints are combined. Best results are highlighted.

in Figure 7.16. Therefore, we always interpret v_s as a vector and define $c_{l/r}$ relative to its direction, which prevents such collapses.

7.5.4.3 BiSp-Jump v2

We first evaluate our methods on the BiSp-Jump v2 dataset. The quality of its images is higher than that of the Jump-Broadcast dataset since the cameras were installed specifically for the analysis tasks and not for TV broadcasts. It is labeled with the same 20 keypoints as the Jump-Broadcast dataset. Segmentation masks are also obtained with detectron2 [95] and kept for all images.

We execute four experiments with different embedding strategies using the keypoint vector encoding with the extension technique for the head (called *V3-VE*), five experiments with the angle technique (called *V3-VA*) and three experiments with the normalized pose encoding (called *V3-NP*). The results are displayed in Table 7.4. Overall, the different strategies achieve similar results. All are capable of detecting arbitrary keypoints on all body parts. Generally, the approaches with the vector encoding achieve better results for most body parts. The number of layers in the embedding process does also not make a large difference independent of the encoding type, which is in contrast to the results reported in Section 7.3. Concatenating before or after the embedding also

Figure 7.17: Detection results for images of the BiSp-Jump v2 dataset (first two images) and the *Jump-Broadcast* dataset (last four images), visualized with equally spaced lines to both sides of each body part including the outer boundary in pure color and the central line in white with a color gradient from the boundary to the skeleton line. The first two images are generated with the keypoint vector encoding using the extension strategy and the last four images with the angle strategy for the head in order to show the differences between the strategies. Occluded/overlapping body parts are omitted for clarity.

does not show a difference. For the angle strategy *V3-VA*, concatenating before the embedding performs slightly better, for the extension strategy *V3-VE*, concatenating after one layer is better. Hence, these differences are not significant and might just be due to typical training performance variance. A significant difference is observed regarding the head keypoint. *V3-VA* achieves an absolute improvement of over 10% for all experiments. Although the normalized pose approach *V3-NP* achieves worse results for all other body parts, its performance on the head keypoint is as good as *V3-VA*. The detection scores vary largely among different body parts. The feet and the hands seem the most challenging body parts, since the PCT is at most 52.4% and 57.1%, respectively. Head, thighs, upper arms, and lower legs are the body parts with the best scores over all experiments. These results might be explained by the fact that these body parts generally lead to larger (and therefore also thicker) segmentation masks, which make it easier for the network to learn and detect precise keypoints. Furthermore, their boundaries are easier to determine than for (maybe bent) elbows and knees or hands and feet. *V3-NP* has especially difficulties with the hands and feet. The reason might be that the area of

Enc.	E	C	PCK@ 0.1	PCK@ 0.05	PCT@ 0.2	head	torso	u.arm	elbow	f.arm	hand	thigh	knee	l.leg	foot
V3-VE	1	1	90.5	71.8	61.3	69.8	79.6	65.8	48.1	51.4	38.5	79.5	64.5	**67.5**	42.8
V3-VE	2	1	90.8	**72.1**	62.5	71.3	80.2	67.6	**49.4**	53.5	40.2	80.4	64.9	66.2	**47.9**
V3-VA	1	1	91.0	71.9	63.1	**83.5**	**80.4**	66.4	49.1	53.0	39.2	80.2	64.4	66.8	46.0
V3-VA	2	1	**91.2**	**72.1**	**63.3**	83.0	79.3	**67.7**	49.2	52.6	40.4	**80.8**	**65.0**	66.1	46.8
V3-VA	2	2	91.0	71.9	63.0	82.7	79.8	66.6	48.0	**53.9**	**40.6**	80.2	63.4	66.1	47.1
V3-NP	1	-	88.0	67.9	57.9	**83.5**	77.3	54.6	41.1	44.8	32.6	79.8	63.3	65.8	31.7
V3-NP	4	-	88.1	67.9	58.1	83.4	77.9	56.5	41.1	45.8	33.3	79.0	63.0	65.0	31.5

Table 7.5: Recall values for the Jump-Broadcast test set in % at PCK@0.1 and PCK@0.05 and PCT@0.2. These scores are evaluated on the test set with the fixed keypoints and on the described 2250 keypoints per body part as far as these keypoints exist in the image. The first column indicates the used encoding: *V3-V* indicates the keypoint vector approach, either with the head extension strategy *V3-VE* or the head angle strategy *V3-VA*. *V3-NP* refers to the normalized pose approach. The second column (labeled E) names the number of layers used for the embedding and the third column (labeled C) the number of layers that is executed before concatenation of the vectors. The third table section contains the average metric results over all keypoints in the test set, the fourth section lists the PCT scores separately for each keypoint type, left and right keypoints are combined. Best results are highlighted.

these body parts has the lowest size in the normalized pose template. The best overall PCT and PCK@0.05 score is achieved with *V3-VA* and a two layer embedding with concatenation right at the beginning. Qualitative results for this dataset are visualized in the first two images in Figure 7.17.

7.5.4.4 Jump-Broadcast

For the Jump-Broadcast dataset, we execute two experiments with the keypoint vector encoding and the extension strategy *V3-VE*, three experiments with the head angle strategy *V3-VA*, and two experiments with the normalized pose encoding *V3-NP*. The results are shown in Table 7.5. Since the quality of the images is worse than for the BiSp-Jump v2 dataset, the scores are generally lower. The effect that the results of the angle strategy *V3-VA* improve the score for the head can be observed for this dataset as well. The improvement is even slightly larger, with an absolute increase of over 11% for all variants. The score for the other body parts are similar across all variants of *V3-VE* and *V3-VA*. Using more layers or other concatenation techniques shows no significant difference. Keypoints on the hands and feet stay the most challenging ones in this dataset. For the keypoint vector encoding approaches *V3-VA* and *V3-VE*, the scores for the feet are better than the scores for the hands, which is different from the results of the BiSp-Jump v2 dataset. However, this changes for the normalized pose approach *V3-NP*, which might be due to the fact that the feet are slightly smaller in the pose template than the hands. Moreover, the scores of the elbow are significantly lower relative to the score of the upper arm and forearm compared to the other dataset.

The best scores are achieved for the head, the thigh, and the torso. *V3-NP* achieves again the lowest overall scores. The largest drop is also observable for keypoints on the feet, while the score for keypoints on the head are better than for the extension strategy. Furthermore, *V3-VA* achieves the best overall results (PCT, PCK@0.05 and PCK@0.1) with a two layer embedding and the concatenation after one layer. Example detections for this method are displayed in the last four images in Figure 7.17.

7.6 Summary

In this chapter, we have introduced a method to detect arbitrary keypoints on the whole human body. We started with an approach that is applicable to the limbs. Next, we extended this approach to the skis of ski jumpers and evaluated strategies to apply this approach to scenarios with only very few available segmentation masks. We found that in contrast to segmentation models, our method has the advantage to be trained with partly correct segmentation masks. This is especially useful when using automatically generated segmentation masks, which always contain errors.

In the last step, we have extended our approach to make our model capable of estimating arbitrary keypoints on all body parts, including limbs as before and further head, torso, hands, and feet. We further enhanced the approach to cope with bent limbs. We have evaluated our approach on two challenging datasets of triple, long, and high jump athletes. To generate ground truth keypoints for the hands, which have no second annotated enclosed keypoint at the border, we extend the line through wrist and hand keypoint to the boundary of the segmentation mask to obtain a second point. For the head, we use either the same technique, or we rotate a line around the head keypoint. We further calculate the points on the inner side of the elbow and knee joints and rotate around these points in order to generate arbitrary points on the elbow and knee, since these joints are often heavily bent during jumps and could not be estimated correctly with the first two approaches. We have evaluated our model with different encodings of the arbitrary keypoints, either as keypoint vectors with the extension strategy for the head or the angle strategy, or as keypoint coordinates of a normalized human pose. In the vector case, we have introduced a third vector in case of the angle strategy and evaluated the performance depending on the number of embedding layers used before we concatenate the embeddings of the single vectors. Evaluations on the BiSp-Jump v2 dataset and the Jump-Broadcast dataset have shown that the results for all variants are generally similar. The normalized pose approach achieves slightly lower scores regarding all experiments. Concatenating the vector embeddings later results in minimal better scores. For both datasets, keypoints on the hand are most challenging. The extension and angle approach for the head have strengths for different keypoints. For the BiSp-Jump v2 and the Jump-Broadcast dataset, the angle approach with a two-layer embedding achieves the best overall score. Hence, the final method proposed in this chapter is capable of detecting arbitrary keypoints on the whole human body, exemplary shown in the application of triple, long, and high jump athletes, including bent elbows and knees.

Part III

3D Human Pose and Mesh Estimation in Challenging Scenarios

8 Foundations

2D HPE is not enough for many applications. Especially for some high-performance sports, but also for 3D animation, gaming, fashion, and many other fields, a human pose in 3D is necessary. Further, for an even more precise estimation, a human mesh can be estimated. Human Mesh Estimation (HME) covers the human pose like 3D HPE and further includes the human shape.

However, estimating any kind of three-dimensional object from videos or images which are naturally only 2D is a challenging task. With multiple synchronized camera views of the same scene, the best quality of the 3D reconstruction can be achieved. However, in many cases, only a single camera view is available or possible. Therefore, monocular 3D HPE and HME are of high interest and an active area of research.

In this chapter, we introduce the necessary foundations for 3D human pose and mesh estimation. We start with the definition of the tasks for 3D HPE and HME. Next, we explain the human body model SMPL-X, which is the most commonly used model for HME. We further introduce the necessary metrics for evaluation, along with common everyday and sports datasets for 3D HPE and HME.

8.1 Task Definitions

At first, we define the two tasks of estimating a human in three dimensions: 3D Human Pose Estimation (3D HPE) and 3D Human Mesh Estimation (3D HME). Both tasks are well established within the field of computer vision and have been researched for many years. We will define the tasks in their typical form and explain the differences between them in this section.

8.1.1 3D Human Pose Estimation

Generally, 3D HPE is closely related to 2D HPE. The goal is to estimate the 3D pose of a human. The pose is represented by the 3D coordinates of certain desired keypoints on the human body. Hence, the output of a 3D HPE model are the x-, y-, and z-coordinates of each desired keypoint, just like 2D HPE with one additional coordinate. Formally, if we want to estimate k keypoints in 3D, we need a model $M_{\text{3D-HPE}}$ that operates on input I such that

$$M_{\text{3D-HPE}}(I) = p^{\text{3D}} \in \mathbb{R}^{k\times 3}. \tag{8.1}$$

The input I is either an image or a video. In contrast to 2D HPE where the output measurement unit are pixels, the 3D pose is often given in a metric unit, typically meters. We will stick to this convention and use meters throughout this thesis. In most cases,

the output is a single pose as denoted in the equation, even in the case when the input is a video. In such cases, the output is the pose of the central frame of the video. This also applies to the 3D HPE approach that we are using in the next chapter.

Estimating 3D keypoints instead of 2D keypoints is much more challenging. Inferring the depth from images or videos is not trivial and requires additional information. This information can be provided by multiple synchronized cameras, depth sensors, or other additional information. However, in many cases, only one camera view is available and no additional sensors. In this case, the depth information has to be inferred from the single image or video, which is even more challenging. Recent research focuses on the estimation of 3D human poses from monocular videos, since most 3D datasets and applications are based on videos. Using a video as an input, the temporal information can be used to infer the depth of the human pose. Since images contain a large amount of parameters, designing a model that can take long sequences of images as an input is not feasible. However, long sequences are necessary to achieve a good 3D pose result. Therefore, the currently most popular approach in 3D HPE is to first apply a 2D HPE model to each frame of a video to obtain a sequence of 2D poses for this video. The number of values defining a single 2D pose is much smaller than the number of values defining an image. In a second step, a long sequence of 2D poses is then used as an input for a so called *2D to 3D uplifting model* that estimates a single 3D pose. Formally, a 2D to 3D uplifting model $M_{\text{2D-to-3D}}$ is applied to a sequence of $2n + 1$ 2D poses $p^{\text{2D}}_{i-n}, ..., p^{\text{2D}}_{i+n} \in \mathbb{R}^{(2n+1)\times k\times 2}$ to obtain the 3D pose $p^{\text{3D}}_i \in \mathbb{R}^{k\times 3}$ of the central frame i:

$$M_{\text{2D-to-3D}}(p^{\text{2D}}_{i-n}, ..., p^{\text{2D}}_{i+n}) = p^{\text{3D}}_i. \tag{8.2}$$

The sequence length n is a fixed number and can usually not be changed. Models can be trained with different sequence lengths n, but this results in different models. During inference, the exact same sequence length as during training needs to be used. 2D HPE models achieve a high accuracy and are well established, since many datasets with annotations are available. Therefore, 2D to 3D uplifting models are currently the most common and successful approach for 3D HPE. A visualization of the 2D to 3D upsampling pipeline is shown in Figure 8.1.

8.1.2 3D Human Mesh Estimation

3D HPE models a human pose as a set of 3D keypoints and a skeleton. However, the real 3D pose is not completely defined by joint positions, the rotations of the body parts are missing in the 3D HPE representation. Further, the human body is not only defined by the pose, but also by the shape. The body shape is described by the surface of the body and can be represented by a mesh. Human mesh models have typically around 10,000 vertices and are much more complex than a skeleton. Estimating this amount of vertices is not feasible. Therefore, a human mesh is usually represented by a low-dimensional, parametrized model. The number of free parameters is kept low, and the model generates a final mesh based on the free parameters. Formally, an HME model M_{HME} operates on an input image I and estimates the parameters $q \in \mathbb{R}^w, w \ll v$ of a human mesh model M_{mesh}:

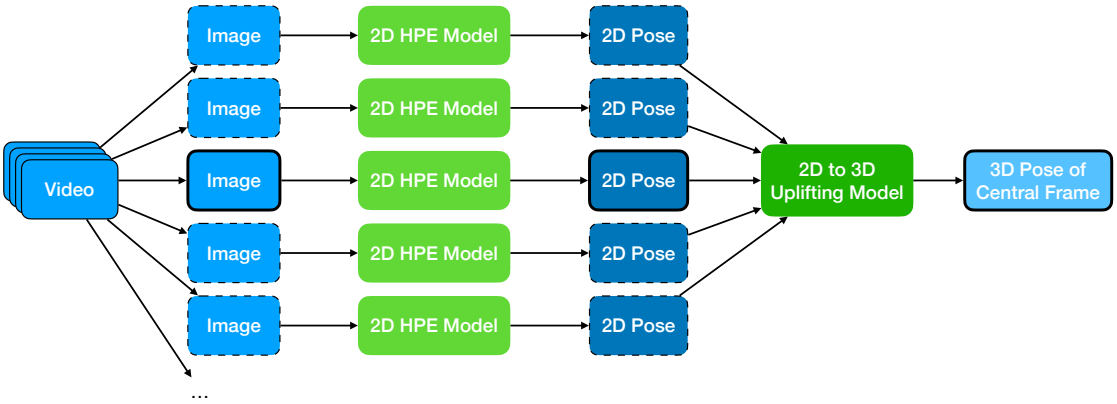


Figure 8.1: The 2D to 3D upsampling pipeline for 3D HPE.

$$M_{\text{HME}}(I) = q. \tag{8.3}$$

The human mesh model creates the mesh $m \in \mathbb{R}^{v \times 3}$, whereby v is the number of vertices of the mesh, and in the magnitude of $10,000$:

$$M_{\text{mesh}}(q) = m. \tag{8.4}$$

There are multiple different mesh models M_{mesh} used in research, but the most common model is the Skinned Multi-Person Linear model (SMPL) [51] and its successor SMPL-X [69], which we will explain in the next section.

8.2 SMPL-X Body Model

SMPL-X [69] is the currently most common body model used for 3D HME. The model is based on the SMPL model [51]. SMPL-X extends SMPL by adding a more detailed hand and face model, resulting in more vertices in the mesh. Generally, SMPL and SMPL-X work very similar, therefore we explain SMPL in the following and point out the differences to SMPL-X in the end.

The main idea of SMPL is the separation of the pose and the body shape component. A unique person is characterized by their body shape parameters β, and these parameters do not change if this person changes the pose. The pose parameters θ, in contrast, model only the pose and do not have an influence on the basic body shape. In contrast to 3D HPE, the SMPL pose parameters θ describe the rotations of the body parts around certain fixed joints. So-called blend weights $\mathcal{W}$ define how much influence the rotation of each body part has on each vertex of the mesh. The location of the joints can be regressed from the mesh and depend on the body shape. Formally, the joints are regressed with a regressor matrix $r \in \mathbb{R}^{k \times v}$ from the mesh m, which depends on the body shape parameters β: $p^{3\text{D}} = r \cdot m$.

However, stating that pose and shape can be modeled independently is not entirely correct. The pose causes some deformations to the body surface, e.g., the bending of

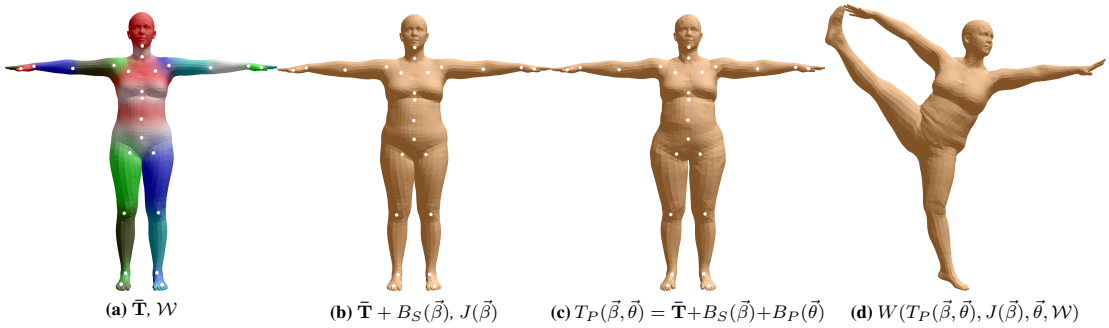


Figure 8.2: The steps of creating a human mesh from body shape parameters β and pose parameters θ with the SMPL model. [51]

the arms or legs. This is taken into account by the SMPL model as well. The final body mesh is created from the pose parameters θ and the body shape parameters β in the following way. The steps are visualized in Figure 8.2.

(a) The basic human body model is a pre-defined T-shaped pose $\overline{T}$ of a standard body shape. The blend weights $\mathcal{W}$ are also visualized in the figure.

(b) A shape blend shape function B_S takes the body shape parameters β and creates the blend shape offsets $B_S(\beta)$. Adding those to the basic body model results in a T-posed human mesh with the individual body shape. The joints can be regressed from the mesh with a joint location function J, which uses the regressor matrix r as explained above.

(c) The pose blend shape function B_P takes the pose parameters θ and creates the blend shape offsets $B_P(\theta)$. Adding those to the T-posed human mesh results in the human mesh with individual body shape and pose dependent deformations, called $T_P(\beta, \theta)$.

(d) The mesh is rotated around the joints $J(\beta)$ to create the final human mesh by a blend skinning function W. This function takes the blend weights $\mathcal{W}$ into account, which define the influence of the rotations on each vertex. This is especially relevant in border regions between two body parts.

The shape blend shape function B_S, the pose blend shape function B_P, and the joint regression function J are linear functions and learned from data. The SMPL model is trained with a large dataset of registered meshes. Some example meshes for varying poses and body shapes are shown in Figure 8.3. The standard T-pose mesh is also learned. Its shape and the body shape parameters are extracted via principal component analysis. In the case of the body shape parameters, the first principal component of the PCA executed on the body shape parameters is the first β parameter, and so on.

SMPL-X extends SMPL by adding a more detailed hand and face model, which are also learned from data. Further, the mesh is more fine-grained and contains more vertices.

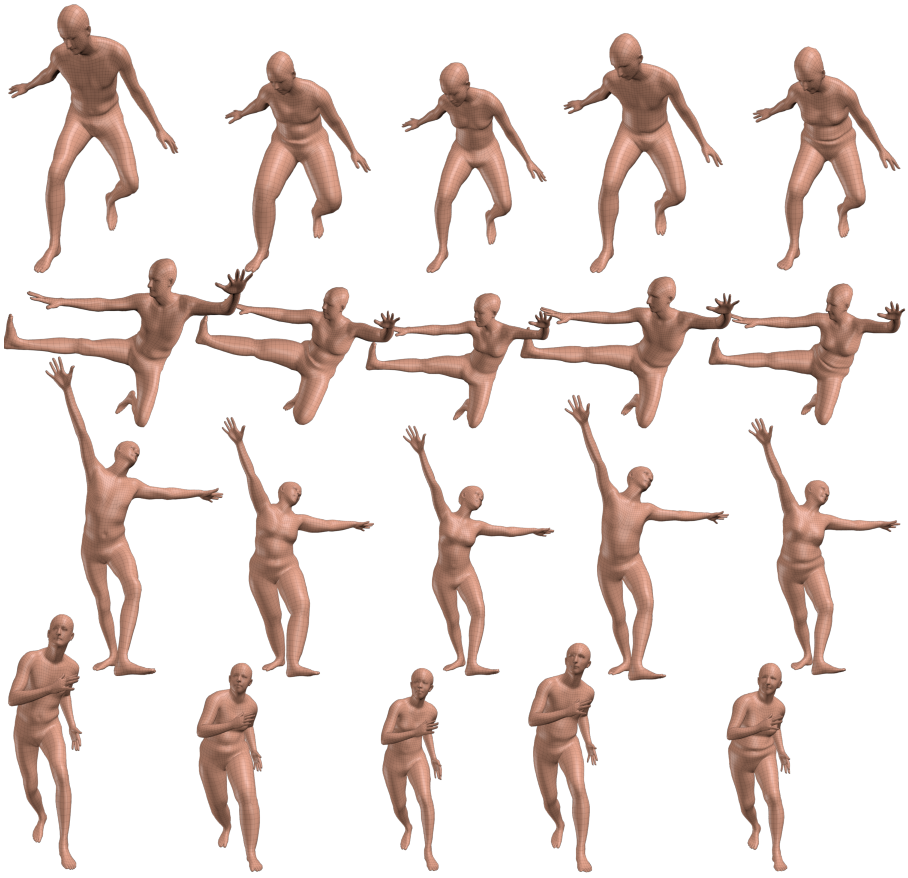


Figure 8.3: Example meshes generated by the SMPL model for varying poses and body shapes. The shape changes along the x-axis, the pose along the y-axis. [51]

Although we do not model the hands and face in this thesis, we use the SMPL-X model because of its finer mesh and because it is the more recent model.

Summarizing, the SMPL-X body model is a parametrized human mesh model that can generate a human mesh for a given pose and body shape. This mesh m is defined by the function $m = W(T_P(\beta, \theta), J(\beta), \theta, \mathcal{W})$. Since W, T_P, J and $\mathcal{W}$ are fixed functions for the SMPL-X model, we will abbreviate it in the following as $m = M_{\text{SMPL-X}}(\beta, \theta)$.

8.3 Evaluation Metrics

There are two most commonly used metrics for 3D HPE and an additional one for HME. These metrics are based on the distance error per joint (or vertex) of the estimated and ground truth joints (or vertices). Since the locations of the joints (or vertices) are given in relation to the real world, often in metric units, a mean distance error is meaningful and interpretable. We will describe the metrics in the following subsections.

8.3.1 Mean Per Joint Position Error (MPJPE)

The Mean Per Joint Position Error (MPJPE) is the most commonly used metric for 3D HPE. It calculates the mean distance error between the estimated and the ground truth joints, whereby the root of the coordinate system is shifted to a special root joint. The error is calculated for each joint and then averaged over all joints.

Let k be the number of joints to estimate, $P \in \mathbb{R}^{k\times 3}$ be the predicted pose and $G \in \mathbb{R}^{k\times 3}$ be the ground truth pose. P_i are further the coordinates of the i-th joint of the pose. Let r be the index of the root joint, which is usually the pelvis or hip joint. Then, the MPJPE is defined as follows:

$$\text{MPJPE}(P, G) = \frac{1}{k}\sum_{i=1}^{k} \left\|(P_i - P_r) - (G_i - G_r)\right\|_2, \tag{8.5}$$

The MPJPE is further averaged over all poses of the dataset to condense the metric into a single value.

8.3.2 Procrustes Aligned Mean Per Joint Position Error (PA-MPJPE)

The Procrustes Aligned Mean Per Joint Position Error (PA-MPJPE) is a commonly used variant of the MPJPE that only measures the inner structure of the estimated pose. It first aligns the predicted pose to the ground truth pose with a Procrustes transformation. The Procrustes transformation is defined by a rotation, a scaling, and a translation to align two point sets optimally. Thus, it is a linear transformation. The transformation is calculated in a way that the sum of the squared distances between the corresponding points of the two point sets is minimized. The PA-MPJPE is then calculated as the MPJPE between the aligned predicted pose and the ground truth pose. Hence, the PA-MPJPE is the lowest MPJPE that can be achieved by rotating, scaling, and translating the predicted pose. The MPJPE already accounts for some translation errors by shifting the root joint to the origin, but the PA-MPJPE accounts for all translation, scaling and rotation errors. Therefore, it does not measure how well the scaling and rotation of the pose is estimated, but only the inner structure.

Let P, G, and k be defined as in Section 8.3.1. Let $R \in \mathbb{R}^{3\times 3}$ be a rotation matrix, $s \in \mathbb{R}^{+}$ be a positive scaling factor, and $t \in \mathbb{R}^3$ be a translation vector. Then, the PA-MPJPE is defined as follows:

$$\text{PA-MPJPE}(P, G) = \min_{R,s,t} \frac{1}{k}\sum_{i=1}^{k} \left\|sRP_i + t - G_i\right\|_2, \tag{8.6}$$

Like the MPJPE, the PA-MPJPE is averaged over all poses of the dataset and typically given as a single value.

8.3.3 Mean Vertex Error (MVE)

The Mean Vertex Error (MVE) is a common metric for 3D HME. It is similar to the MPJPE, but instead of measuring the error per joint, it measures the error per vertex

of the mesh. Hence, in contrast to (PA-)MPJPE, it measures the precision of the whole mesh, and not only the precision of the keypoints. To calculate it, both meshes are aligned at a root joint, which is most typically the hip or pelvis joint. The error is calculated as the mean distance between the estimated and the ground truth vertices. Let M be an estimated human mesh with v vertices $M_i, i = 1, ..., v$ and G the corresponding ground truth mesh. Let P_r be the ground truth root keypoint and $\widehat{P}_r$ the root keypoint of the estimated mesh. The MVE is defined as follows:

$$\text{MVE}(M, G) = \frac{1}{v} \sum_{i=1}^{v} \left\| (M_i - \widehat{P}_r) - (G_i - P_r) \right\|_2, \tag{8.7}$$

The main difference between the MPJPE and the MVE is the amount of points, 3D poses consist only of a few joints, whereas 3D meshes consist of thousands of vertices. Moreover, the root keypoint does not necessarily have to be a vertex of the mesh, it is mostly regressed with the vertices of the mesh and lies inside the body.

8.3.4 Problems

MPJPE and PA-MPJPE have some drawbacks. First, they are very sensitive to outliers. If all joints despite one are estimated correctly, but a single one very poorly, this can lead to a high error. Considering averaging over the whole dataset, this outlier problem also persists with poorly estimated poses. A single pose with a very high error can lead to a high total MPJPE or PA-MPJPE, although all other poses are estimated well. Hence, single outliers can affect the total metric to a large extent. Further, the metrics do not take into account undetected poses. There is no option to include missing poses, despite just adding an arbitrary pose, which has the problem of being an outlier, like described before. This problem also applies to MVE. A metric like PCK for 2D HPE would be more suitable in these cases, but is not used in 3D HPE nor HME. We stick to the metrics that are typically used in the literature, but the drawbacks should be kept in mind.

8.4 Datasets

For 3D tasks, creating datasets is far more difficult than for 2D tasks. The annotations are not only the 2D coordinates of the joints, but also the depth information. This requires additional sensors or multiple camera views. The most common ways are to create the annotations by 2D annotations and triangulation with multiple camera views [67], or through marker-based systems [38, 26]. Therefore, 3D datasets are much more complex and expensive to create. For that reason, less 3D datasets than 2D datasets exist. Another alternative are synthetic datasets. Large advances in the quality of synthetic datasets have been made in the recent years, e.g., the diversity of the poses and body shapes have increased, and the images are more realistic, with background and humans matching better. However, the quality of synthetic datasets is still not as good as real datasets, but they can help, e.g., by using them as a large-scale pretraining. In this section, we introduce some datasets for 3D HPE and HME that we use in our work.

8.4.1 Human3.6M

Human3.6M [38] is a large scale 3D human pose dataset. It contains 3.6 million 3D poses of 11 subjects performing 15 different everyday actions like discussing, taking a photo, talking on the phone, etc. The dataset is captured with a high-resolution multi-camera setup consisting of four calibrated cameras recording at 50Hz. All recordings are captured in an indoor studio, resulting in the same background for each video. 3D annotations for 17 joints are available for all frames and created by a marker-based motion capture system. In practice, not all the annotations are public, so only the accessible data is used for training and evaluation. This results in the training dataset consisting of all actions from 7 subjects and the evaluation dataset of all actions from 3 subjects. The dataset is widely used in research and is the most commonly used dataset for 3D HPE.

8.4.2 MPI-INF-3DHP

MPI-INF-3D Human Person [64] is another 3D human pose dataset. It is recorded in a green screen studio with 14 cameras. Hence, the background can be replaced with any background, leading to more realistic scenarios than the single studio background from Human3.6M. The dataset contains 3D poses of 8 subjects performing 8 different actions. It is further intended to cover a wider range of motions, poses, camera viewpoints and backgrounds than Human3.6M. The dataset is smaller than Human3.6M, but still large scale with over 1.3 million frames. The dataset is also annotated with 17 joints, the annotations are obtained with a markerless motion capture system.

8.4.3 AMASS

The Archive of Motion Capture as Surface Shapes (AMASS) [63] is a large dataset of human motion capture data. The dataset contains 4D sequences (time and 3D coordinates) of 3D human poses and shapes. It is a pure motion capture dataset, meaning that there is no image or video data. The dataset unifies 15 different optical marker-based motion capture datasets and contains a large variety of different human activities. It consists of more than 40 hours of motion data, covers over 300 subjects and 1,100 motions. The dataset can not be used for computer vision tasks directly, since it does not contain vision data. However, it can be used for 2D to 3D uplifting models and other tasks that do not require vision data.

8.4.4 AGORA

The AGORA dataset [68] is a synthetic, large scale 3D human pose and mesh dataset. 4,240 commercially available, high-quality, textured human scans in diverse poses and natural clothing together with accurate SMPL-X body model fits are used to create the dataset. In total, the dataset contains 14,000 training and 3,000 test images created by rendering between 5 and 15 people per image. In total, AGORA consists of 173K individual person instances.

8.4.5 fit3D

Due to high velocities and a great variation of poses, sports is a challenging scenario for all kinds of human pose and shape estimation. The fit3D dataset [26] is a dataset which consists of videos from gym sports exercises with repetitions and is annotated with human meshes. It is the only sports dataset with public SMPL-X annotations. The public training set contains 8 different subjects recorded from four synchronized camera perspectives. The 3D ground truth poses are obtained with a marker-based motion capture system. The SMPL-X meshes are created by fitting to the markers, multi-view image evidence and body scans. The 3D skeleton consists of 25 joints, including the popular Human3.6M joints. All subjects perform 47 exercises with multiple repetitions, categorized in warm-ups, barbell exercises, dumbbell exercises and equipment-free exercises. Example images are provided in Figure 8.4.

Figure 8.4: Example images from fit3D.

8.4.6 ASPset

ASPset [67] is a 3D human pose dataset. It consists of various different sports motion clips recorded from three camera perspectives. 17 different amateur subjects each perform 30 sports-related actions, which results in a total of 510 action clips. Performed actions contain throwing, catching, swinging a bat/racquet, etc. The footage is recorded outdoors and markerless. Using no markers results in slightly worse annotation quality. On the other side, having no markers in the footage is more realistic. Training HPE models on marker-based datasets can lead the models to rely on the markers. The annotations for 17 joints per person are created by 2D HPE models, triangulation, and some correction mechanisms based on temporal continuity. In total, the dataset consists of 109,901 unique poses. A significant difference to daily life datasets like Human3.6M is the amount of dynamic motion. The authors of the dataset calculate the average motion of the Human3.6M dataset to be $0.27\,m/s$, whereas the average motion of ASPset is $1.21\,m/s$. This is a significant difference and makes the dataset more challenging. Example images from the dataset are shown in Figure 8.5.

Figure 8.5: Example images from ASPset.

9 3D Human Pose and Mesh Estimation with Consistent Body Shapes

In Chapter 7, we focussed on detecting more than the standard keypoints of humans in 2D. We showed that it is possible to estimate keypoints on the border of body parts, the centerline and anywhere in between. This is a first step towards achieving a full understanding of the human body with neural networks. Detecting the standard joints results in a skeleton, but there is no information about the body shape and outline of the person. With our approach of Chapter 7, we capture the shape of each body part and let the 2D HPE model learn a kind of structure of the human body. However, this approach has still some drawbacks. It is a 2D model, meaning that we can identify body parts, borders and keypoints in pixel coordinates of each image, but we do not know how the persons are positioned in the 3D world. Further, the border of body parts depends on the perspective. If the image is taken from the front, the borders of the legs are left and right side of thigh and lower leg. In contrast, if the image is taken from the side, the borders of the legs are the front and back side of it. Hence, the position of the borders on the human body varies with different perspectives. To solve this problem, we need to know where the front, left, right and back side of the human are. This is only possible by introducing the third dimension into the task, at least in the annotations. We go one step further and leverage the research in the field of HME for our purpose. Already having a realistic, adaptable human mesh model solves errors like the output of completely impossible, wrong estimates. However, we find that the currently best performing HME models perform not as good as (3D) HPE models in challenging scenarios like sports regarding the accuracy of the joint estimates. We try to tackle this problem, among others, in this chapter, which is based on the following publication, with some text passages directly taken from it:

Leveraging Anthropometric Measurements to Improve Human Mesh Estimation and Ensure Consistent Body Shapes [56], Katja Ludwig, Julian Lorenz, Daniel Kienzle, Tuan Bui, and Rainer Lienhart, IEEE/CVF International Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), Nashville, TN, June 2025.

Another significant problem that we tackle with our work is the following. The basic body shape (i.e., the body shape in T-pose) of a person does not change within a single video. However, most state-of-the-art Human Mesh Estimation (HME) models output a slightly different, thus inconsistent basic body shape for each video frame. Furthermore, we find that state-of-the-art 3D HPE models outperform HME models

regarding the precision of the estimated 3D keypoint positions. We solve the problem of inconsistent body shapes by leveraging anthropometric measurements like taken by tailors from humans. We create a model called A2B that converts given anthropometric measurements to basic body shape parameters of human mesh models. By combining the A2B model results with the keypoints of 3D HPE models using inverse kinematics (IK), superior and consistent human meshes can be obtained. We evaluate our approach on challenging datasets like ASPset or fit3D, where we can lower the MPJPE by over 30 mm compared to state-of-the-art HME models. Further, replacing estimates of the body shape parameters from existing HME models with A2B results not only increases the performance of these HME models, but also guarantees consistent body shapes. Moreover, using A2B body shape parameters instead of HME models' estimates not only increases the performance of these HME models, but also leads to consistent body shapes. IK created meshes with A2B body shapes can in turn be used for fine-tuning HME models on datasets without mesh annotations, which improves the HME performance by 12 mm on ASPset.[1]

9.1 Introduction

Creating an accurate 3D human mesh from monocular images or videos creates new opportunities in fields like 3D animation, gaming, fashion, sports, etc. In many of these application fields, videos are of main interest. While applying HME to videos, analyses of results of state-of-the-art HME models show that the basic body shape of the meshes of the same person differs from frame to frame.[2] Worse, an analysis of currently used 3D mesh and pose datasets reveals the same inconsistencies in the provided ground truth data. For a precise body posture analysis, as it is necessary in many sports disciplines, an exact model of the athlete's body shape is required. Therefore, most professional athletes are measured anthropometrically during performance assessments today. Moreover, the body shape of an actor performing motions for 3D animations needs to be consistent as the basic body shapes does not change during performances. Thus, the changing body shapes of HME models for the same person are highly unwanted and simply wrong.

Our work aims to create a single perfectly fitting basic body shape for each human and reuse it for all videos with this person. Measuring the human body has already been done for centuries to fit suits or dresses perfectly to a specific body shape. In many applications, measuring the person in action beforehand would add only a marginal overhead, but improves the results dramatically. For this reason we propose to use these measurements. Body shape parameters of common human mesh models like SMPL-X [69] are not human interpretable, as discussed in Section 9.4. Therefore, it is not possible to obtain the perfect body shape parameters by anthropometric measurements. Hence, we train a machine learning model (called *A2B*, anthropometric measurements to body

[1] The code for this chapter is publicly available at `https://github.com/kaulquappe23/a2b_human_mesh`.

[2] The body shape in a given pose is usually modeled by a basic body shape (given in T-pose) plus an additional pose-dependent deformation. We call the basic body shape just body shape in the following, as the pose-specific correction is computed from the pose and does not need to be estimated.

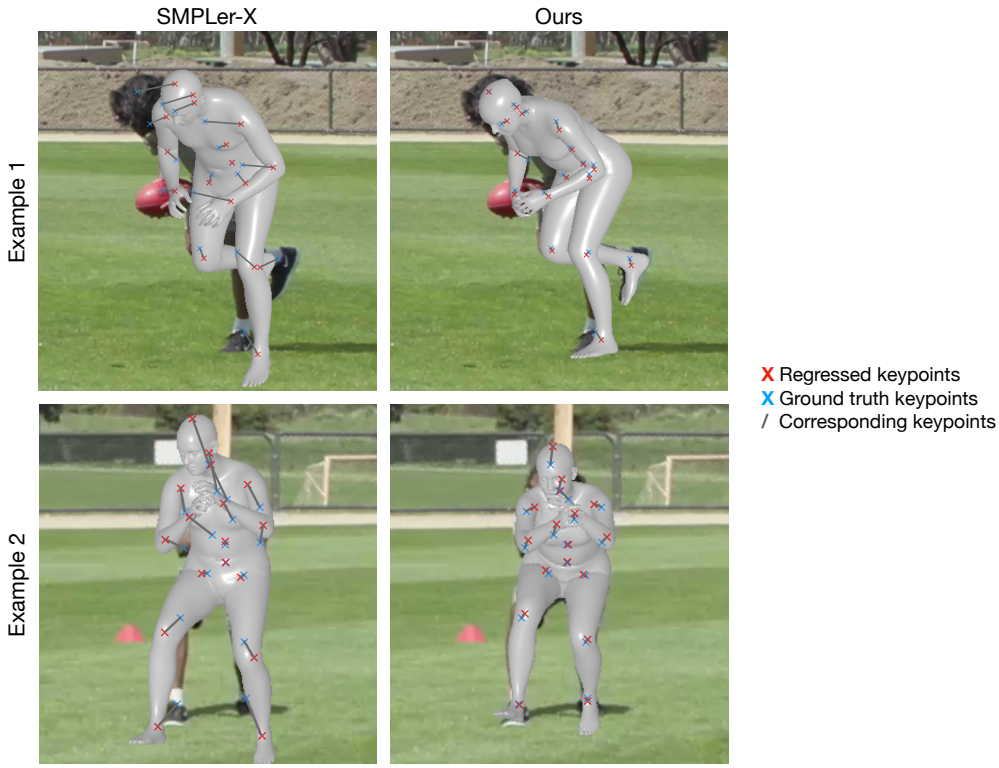


Figure 9.1: Two qualitative examples from the ASPset sports dataset. The result from a state-of-the-art HME model, SMPLer-X [7], is shown on the left, the result from our model on the right, respectively. ground truth joints and estimated joints are color-coded. Corresponding joints are connected.

shape) to translate those measurements into body shape parameters for HME. With this approach, measuring a person once creates the body shape that can be used for all frames in all evaluation videos.

Contemporary HME models are performing well on everyday data. However, in more challenging scenarios like sports, their performance is inferior to fine-tuned state-of-the-art 3D HPE models. 3D HPE models only predict 3D keypoints resulting in a stick-figure pose, whereas HME models output a posed mesh including the human's surface. Due to the lack of ground truth meshes, HME models cannot be trained on datasets with solely 3D keypoint annotations. The usage of synthetic data is emerging in the field, but is not applicable to challenging or specific scenarios like sports. In this chapter, we propose a solution to that problem. With our A2B model and anthropometric measurements, we can now create the body shape parameters of humans needed for HME. We further apply inverse kinematics (IK) to produce the rotations that are missing in the 3D stick-figure model that is created by 3D HPE models. IK takes 3D poses as an input and outputs pose and shape parameters. Discarding the IK shape parameters and replacing them with our A2B body shape, we are able to generate human meshes that have a consistent body shape and a precise pose. The main goal is to estimate the best possible human mesh with a consistent body shape by adding the marginal overhead of measuring humans.

Since our main focus is sports, this overhead is negligible, as professional athletes are commonly measured anyway. We show qualitative examples of our model and a state-of-the-art HME model, SMPLer-X [7], in Figure 9.1. Moreover, we show that by replacing the estimated body shapes of existing HME models with our A2B results, the results of the models can be improved and a consistent body pose is achieved. Our approach is generally applicable to any HME problem. We choose sports datasets to validate our proposed approach, since the poses in sports are challenging for existent HME models and athletes are measured. The performance of existing HME models is currently not good enough to use them in performance assessments of athletes, which we try to change.

Our contributions can be summarized as follows:

- We reveal inconsistencies in the ground truth data of ASPset [67] and fit3D [26]. The body shape of a single person varies mistakenly in the ground truth.
- We create and evaluate different models to convert between anthropometric measurements and SMPL-X body shape parameters. We call these models *A2B*. We introduce a possibility to convert from those measurements to body shape parameters. The reverse direction, *B2A*, is a deterministic function of the human mesh, as the anthropometric measurements are a direct result of the mesh.
- We analyze and compare the performance of existing HME models on ASPset and fit3D. Replacing the estimated body shape parameters (and keeping the pose) of each HME model with A2B body shape parameters increases the performance of all models.
- With fine-tuned state-of-the-art 2D and 3D HPE models [98, 21], IK, anthropometric measurements, and our A2B model, we estimate accurate human meshes with a consistent body shape. We show that this approach achieves superior results to state-of-the-art HME models, although still evaluated on the inconsistent ground truth.
- We generate pseudo ground truth meshes by applying IK to 3D ground truth poses. With these meshes, fine-tuning HME models is now possible. We show exemplary with one HME model that such a fine-tuning can improve the results for the specific dataset.

9.2 Related Work

Human Mesh Estimation (HME) is an active area of research. Body models like SMPL [51] and its successor SMPL-X [69] are broadly used. An in-depth explanation of SMPL-X can be found in Section 8.2. The advantage of these models is that they decouple human pose and shape. The pose parameters θ give the rotation of the joints relative to the parent joint. The shape parameters β model the basic body shape. At first, a mesh is created with a linear mapping from β parameters to a T-shaped pose. Next, some pose-specific shape deformations are applied, and then the mesh is rotated at the joints according to the θ parameters.

Along introducing SMPL-X, Pavlakos et al. [69] propose the first HME model that estimates SMPL-X meshes from images. This model, called SMPLify-X, detects 2D image features and then fits an SMPL-X model to these. To achieve that, they incorporate a pose prior trained on a large motion capture dataset and an interpenetration test. A more recent model for HME is Multi-HMR [2]. It predicts 2D heatmaps for person centers and based on that the human mesh with a human prediction head. OSX [49] is another HME model. It uses a component aware Transformer that is composed of a global body encoder and local decoders for face and hands. Cai et al. [7] introduce a generalist foundation model for HME called SMPLer-X, which is trained on a large amount of datasets and mainly uses Vision Transformers. There is a multitude of other recent HME models, some focussing more on whole-body HME [16, 24, 65], others on multi-person HME - either with a two stage approach using a person detector and a single person human mesh estimator [14, 29, 73], or a single stage approach estimating the meshes of all persons at once [72, 85, 103].

Choutas et al. [15] realized that existing HME models focus more on the body pose than the shape, although the shape is equally important for many applications. They propose SHAPY, a model that uses anthropometric and linguistic attributes to create accurate body shapes. Moreover, Sarkar et al. [77] introduce SoY, which contains specific loss functions to enhance the output's body shape accuracy. They further propose a refinement step during test time that enhances the shape quality even more. KBody [104] follows a predict-and-optimize approach for HME. It leverages virtual joints in order to achieve that and focuses on pixel alignment to accurately capture the shape of the human. AnthroNet [71] propose a new body model that is learned with an end-to-end trainable pipeline. It takes anthropometric measurements as an input to learn a mesh model that accurately captures shapes of humans, but this model is different from the commonly used SMPL-X model. Sengupta et al. [78] include a learned linear regressor from anthropometric measurements to body shape parameters in their image based HME model.

Inverse kinematics (IK) are common in the field of robotics. In the last years, 3D HME approaches leveraged IK to enhance their output. HybrIK [44] transforms 3D joint coordinates to relative body-part rotations for 3D HME by using a twist-and-swing decomposition. HybrIK-X [42] further enhances HybrIK with expressive face and hands. Cha et al. [8] leverage IK to tackle the challenge of person-to-person occlusions in images with interacting persons. PLIKS [79] (Pseudo-Linear Inverse Kinematic Solver) approaches HME as a model-in-the-loop optimization problem by analytically reconstructing the human model via 2D pixel-aligned vertices in an IK-like manner. NIKI [41] tries to unite robustness to occlusions and pixel-aligned accuracy in non-occlusion cases by modeling bidirectional errors. NIKI can learn from both the forward and inverse processes, whereby it emulates the analytical inverse kinematics algorithms with the twist-and-swing decomposition for better interpretability.

Although HME is an active area of research, it is yet not common in computer vision for sports. Due to high velocities and a great variation of poses, sports is a challenging scenario for all kinds of human pose and shape estimation. The fit3D dataset [26] is a dataset which consists of videos from gym sports exercises with repetitions and is anno-

ASPset				fit3D			
Measure	σ	r. σ	r. range	Measure	σ	r. σ	r. range
head	0.91	5.98%	57.91%	head	0.73	2.73%	17.52%
hip width	1.71	9.48%	85.46%	hip circ.	0.87	0.84%	8.17 %
forearm	1.99	8.37%	92.04%	forearm	0.34	1.40%	9.24%
upper arm	1.72	6.29%	66.35%	arm	0.76	1.51%	9.42%
lower leg	1.44	3.60%	41.36%	lower leg	0.52	1.31%	13.80%
thigh	1.65	4.23 %	35.46%	thigh	0.43	1.17%	11.74%
				height	1.60	0.94%	8.69%
				β param.	0.64		

Table 9.1: Ground truth data analysis for ASPset (left) and fit3D (right): Standard deviation σ, relative standard deviation $\frac{\sigma}{avg}$ and relative range $\frac{\max - \min}{avg}$ of anthropometric measurements. Standard deviations are given in cm, but not for the β parameters. The values are averaged between left and right body parts and between all persons from the respective dataset that we use in our evaluations (see Section 9.5). The β parameter standard deviation is averaged over all β parameters.

tated with human meshes. AIFit [26] is a tool trained on fit3D which can reconstruct 3D human poses, reliably segment exercise repetitions, and identify the deviations between standards learned from trainers, and the execution of a trainee to give feedback to trainees. Other sports datasets only consist of 3D joint annotations, like ASPset [67] or SportsPose [37]. SportsCap [11] is an approach for simultaneously capturing 3D human motions and understanding fine-grained actions from monocular challenging sports videos. Moreover, Baumgartner et al. [3] make use of partial sports field registration to enhance 3D HPE results for athletes running on a track. More common are daily life datasets like Human3.6M [38] or MPI-INF-3DHP [64]. They consist of everyday activities like discussing, walking, talking on the phone, etc., which are not as challenging as sports activities with faster motions and a broader range of poses.

9.3 Analysis of Errors in 3D Human Ground Truth

Each person has a specific basic body shape that does not change over a short time period. Therefore, the SMPL-X body model decouples the human pose encoded by θ parameters from the basic body shape encoded by β parameters. Deformations to the basic body shape that are caused by the current pose are modeled separately. Therefore, it makes sense to assign a single set of shape parameters β to a person for a given short time period such as a recorded action to describe his/her shape. Further, there are lengths that can be calculated from 3D joints that should never change, since individual bones of humans are rigid and should not be deformed by different poses. Our approach enforces a single set of shape parameters per person and immutable bone lengths.

As a first step, we analyze if the ground truth data of our used datasets fulfills these properties. In this work, we use ASPset [67] and fit3D [26], since both datasets consist

MPI-INF-3DHP							
3D joint annotations				SMPL-X annotations (NeuralAnnot)			
Measure	σ	r. σ	r. range	Measure	σ	r. σ	r. range
head	0.19	1.03%	2.08%	head	0.21	0.75%	4.87%
hip width	0.22	0.89%	1.80%	hip circ.	1.16	1.16%	9.13 %
forearm	0.21	0.87%	1.77%	forearm	0.45	1.80%	9.75%
upper arm	0.29	0.90%	1.82%	arm	0.83	1.59%	8.19%
lower leg	0.60	1.49%	3.06%	lower leg	1.05	2.56%	11.54%
thigh	3.83	7.91 %	41.90%	thigh	0.77	2.02%	9.47%
				height	2.76	1.56%	8.24%
				β param.	0.18		

Table 9.2: Ground truth data analysis for MPI-INF-3DHP [64]. Bone length analysis based on the 3D joint locations (left) and on SMPL-X annotations by NeuralAnnot (right). Standard deviation σ, relative standard deviation $\frac{\sigma}{avg}$ and relative range $\frac{\max - \min}{avg}$ of anthropometric measurements are reported. Standard deviations are given in cm, despite for the β parameters. The values are averaged between left and right body parts and between all persons in each dataset. The β parameter standard deviation is averaged over all β parameters.

of videos with fast changing poses and 3D ground truth. Results for the Human3.6M [38] and MPI-INF-3DHP [64] datasets are presented in the next paragraph. For ASPset, we analyze bone lengths, since it has only ground truth annotations for 3D joints. For fit3D, ground truth SMPL-X β parameters are available, hence we can analyze the β parameters directly and further the derived anthropometric measurements. These values are the output of our deterministic B2A function: It generates a standard T-pose with the given β parameters and computes 36 anthropometric measurements from the resulting mesh. Results of our ground truth analysis for a subset of the anthropometric values are shown in Table 9.1. We can see that the ground truth itself is not consistent. The deviations are larger for ASPset. Although we have ground truth SMPL-X meshes for fit3D, every β parameter of a single person has a standard deviation of 0.64 on average.[3] ASPset reveals a standard deviation of 1.44 cm to 1.99 cm for all limbs (forearm, upper arm, lower leg, thigh).

Compared to the average length of such a body part, this is a large deviation. The deviation in relation to the average size is given by the relative standard deviation in the table. The relative range denotes the range of the lengths that appear in the ground truth in relation to the average length. This value is not robust to outliers, but shows that lengths contained in the ground truth vary a lot, especially the length of the forearm and the hip width for ASPset, which have a relative standard deviation of over 8% and a relative range of over 85%. This is a relevant flaw in the ground truth shape annotation, since based on the model, the ground truth shape should be consistent for

[3] Averaged standard deviation means (in the whole chapter) that the standard deviation is calculated per person over all video sequences, and the mean of the resulting standard deviations is calculated afterwards.

each human. Nevertheless, we use the given inconsistent ground truth for our evaluations for comparability with related work and as we have no good means to correct them. The reader should keep this in mind.

We further analyze the ground truth shape consistency for the common datasets Human3.6M [38] and MPI-INF-3DHP [64]. We find that for Human3.6M, the bone lengths derived from the 3D annotations are fixed and contain no errors, but not for MPI-INF-3DHP. Therefore, we do not report the deviations of 3D joint annotations for Human3.6M, since there are none. We further evaluate the SMPL-X annotations for both datasets provided by NeuralAnnot [66] which are used by HME models as ground truth for training. See Tables 9.2 and 9.3 for details. In total, the deviations are smaller than for the sports datasets. This might be due to the fact that the poses are less dynamic and the movements are slower. The original 3D joint annotations of Human3.6M are well-prepared and contain no error, but the annotations of MPI-INF-3DHP contain errors, especially the thigh length is very inconsistent. The SMPL-X annotations provided by NeuralAnnot that are broadly used for HME training also contain inconsistencies for both Human3.6M and MPI-INF-3DHP. For example, the standard deviation of the height of a person is 2.76 cm for MPI-INF-3DHP and 3.40 cm for Human3.6M, which should not be the case. Therefore, we want to encourage future research in the field of 3D human pose and mesh data collection to try to eliminate these flaws in the provided ground truth.

Human3.6M: SMPL-X annotations (NeuralAnnot)			
Measurement	σ	r. σ	r. range
head	0.41	1.51%	10.28%
hip circ.	1.24	1.19%	8.90%
forearm	0.83	3.30%	27.93%
arm	0.77	2.58%	22.88%
lower leg	0.43	1.18%	12.20%
thigh	0.66	1.27%	9.43%
height	3.40	2.06%	15.66%
β param.	0.20		

Table 9.3: Ground truth data analysis for Human3.6M [64]: Analysis of SMPL-X annotations by NeuralAnnot. Standard deviation σ, relative standard deviation $\frac{\sigma}{avg}$ and relative range $\frac{\max - \min}{avg}$ of anthropometric measurements are reported. Standard deviations are given in cm, despite for the β parameters. The values are averaged between left and right body parts and between all persons in each dataset. The β parameter standard deviation is averaged over all β parameters.

9.4 From Anthropometric Measurements to Body Shape

Humans have been measured for centuries [19]. Tailors know exactly which measurements to take for perfectly fitting a suit or dress to the body shape of a customer. In sports, it is already common practice that professional athletes are measured for precise

performance assessments. Measuring a human is easy and well understood. In contrast, the parameters of the body shape for human mesh models like SMPL-X [69] are not human interpretable. The β parameters describe the principal components of the human body shape with typically around 10 to 16 values and are the result of a PCA executed on the human meshes of a training dataset while learning the SMPL-X model. Fixing all β parameters except for one and looking at the results lets human observers get a notion of what this parameter might mean, see Figure 9.2 for an example, but in total, the β parameters and their interactions are not well interpretable. Therefore, we want to leverage the well established technique of measuring humans to create precise body shape parameters for the commonly used SMPL-X human mesh model. We call our approach to convert from 36 Anthropometric measurements to Body shape parameters *A2B*. Since there is no known relation between anthropometric measurements and β parameters, our aim is to learn this mapping. The reverse direction, *B2A*, is a deterministic function of the human mesh, as the anthropometric measurements can be measured from the mesh.

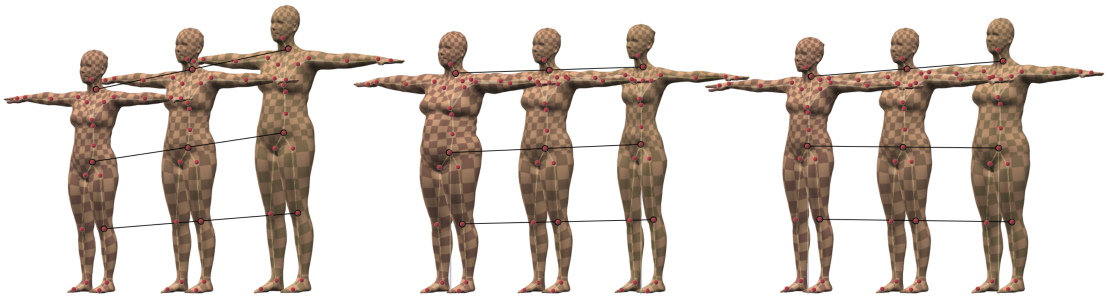


Figure 9.2: Visualization of changing the first three β parameters while keeping the rest of the β parameters fixed. [51]

9.4.1 Data Generation

We select the anthropometric measurements for our models based on the selections of AnthroNet [71] and an anthropometry study of the U.S. army [31]. In total, 36 measurements are selected. They can be categorized into 23 lengths and 13 circumferences. All measurements are taken based on the standard SMPL-X T-pose. The reference landmarks are chosen by matching the vertices on the default mesh with the landmarks defined by the anthropometric survey of the U.S. army personnel [31]. A visualization of the landmarks can be found in Figure 9.3. The lengths are calculated by computing the Euclidean distance between two landmarks, or the difference along the coordinate axis pointing upwards for certain heights. The lengths are visualized in Figure 9.4. Table 9.4 lists the enclosing landmarks for each length. To measure the circumferences, we adopt the code from [5]. For each measurement, a plane is created, the intersection between the mesh and the plane are extracted and the convex hull of the result is calculated. During this process, the mesh is restricted to the body part to be measured. A visualization of the circumferences can be found in Figure 9.5 and a list of the landmarks and the normal vectors spanning the plane in Table 9.5.

Idx	Length Name	From	To
1	Shoulder width	Left shoulder tip (left acromion)	Right shoulder tip
2	Back torso height	Cervicale	Back belly button
3	Front torso height	Suprasternale (top of the breastbone)	Belly button
4	Head	Head top	Cervicale
5	Midline neck	Chin	Suprasternale
6	Lateral neck	Center between the ears	Cervicale
7	Height	Head top	Center between heels
8/9	Hand right/left	Center between middle and ring finger	Stylion rotated downwards
10/11	Arm right/left	Acromion	Wrist
12/13	Forearm right/left	Elbow	Stylion rotated downwards
14/15	Thigh right/left	Outer point at the femur (Trochanterion)	Knee cap
16/17	Calf right/left	Knee cap	Ankle
18/19	Foot width right/left	Small toe	Big toe
20/21	Heel to ball right/left	Heel	Ball
22/23	Heel to toe right/left	Heel	Big toe

Table 9.4: Definitions of lengths by their two enclosing landmarks.

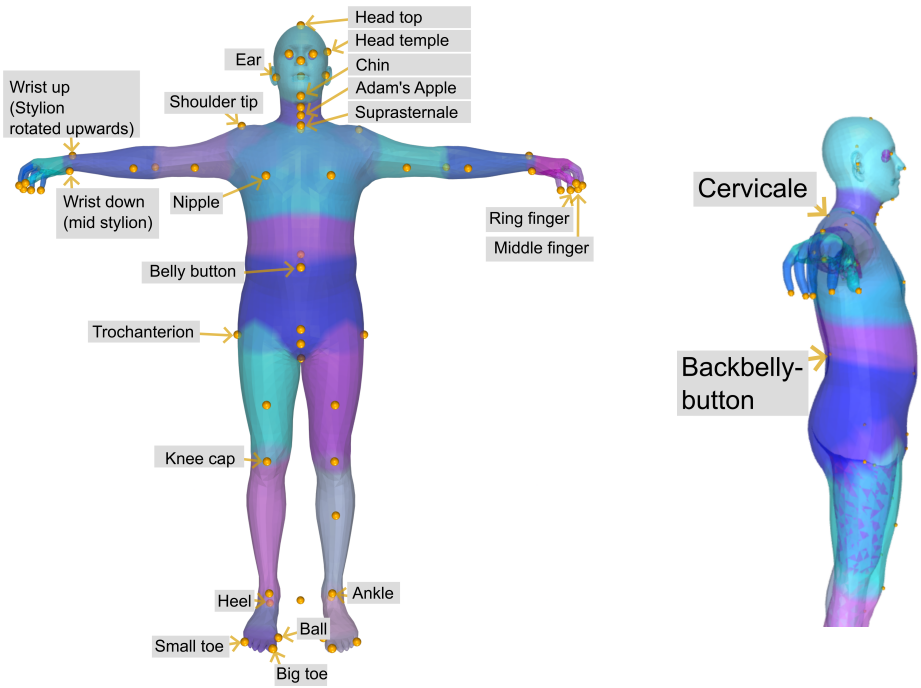


Figure 9.3: Visualization of the used landmarks for a standard T-pose SMPL-X mesh in front view (left) and side view (right).

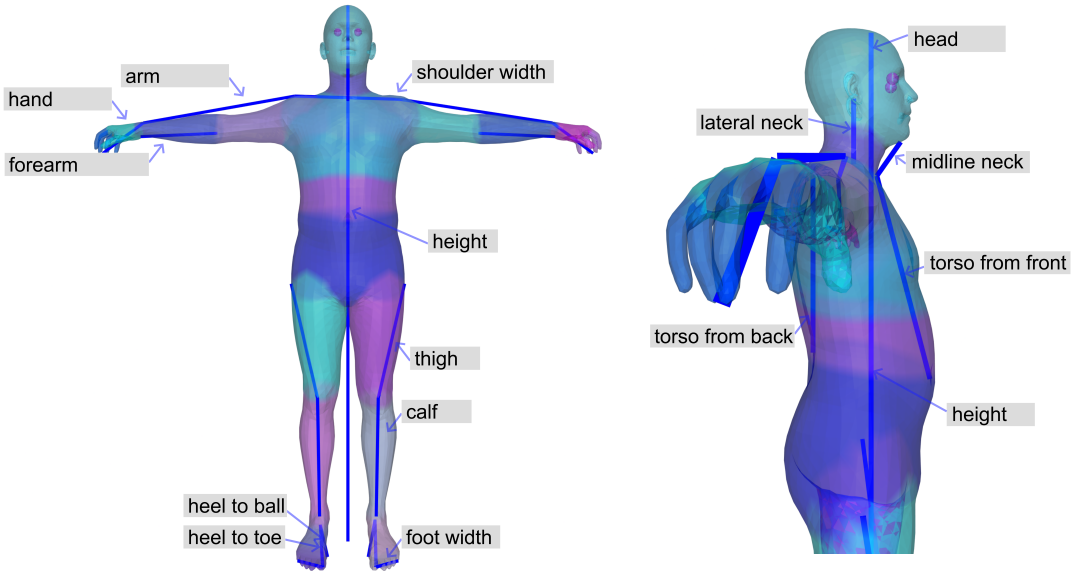


Figure 9.4: Visualization of used lengths for a standard T-pose SMPL-X mesh in front view (left) and side view (right).

Many existing datasets provide a wide range of different poses, but most incorporate the same humans. For learning a conversion model from anthropometric measurements to β parameters, we need a lot of samples for different humans, no matter the pose. With given shape parameters, we can use the B2A function to compute the measurements. Recall, B2A is a deterministic function measuring the anthropometric measurements from meshes in T-pose.

Because many different body shapes are required for the learning process, we use the AGORA [68] dataset. It consists of 1447 male and 1588 female subjects. We are not able to use the larger dataset from AnthroNet [71], since it uses its own mesh model and the authors did not publish their conversion to the SMPL-X model, which we want to use as it is most commonly used in research. Although comparably large, 1447/1588 subjects is still only a little amount of data to learn

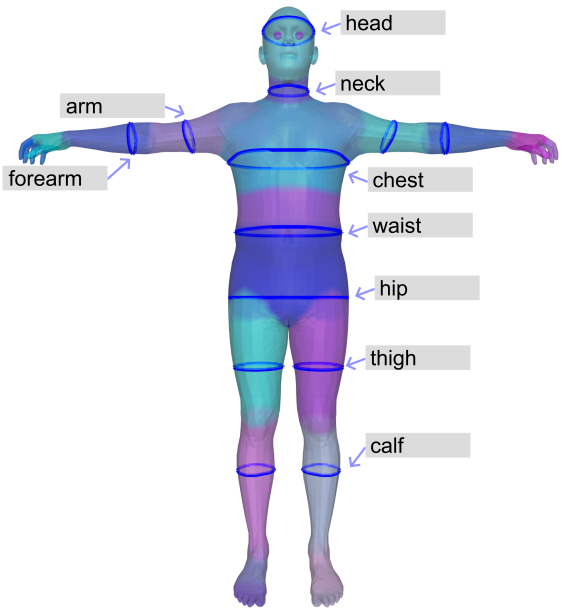


Figure 9.5: Visualization of used circumferences for a standard T-pose mesh in front view.

Idx	Circumference	Normal Vector	Position
1	Waist	Up	Belly button
2	Chest	Up	Nipple
3	Hip	Up	Pubic bone
4	Head	Up	Head temple
5	Neck	Spine to head	Adam's apple
6/7	Upper Arm	Shoulder to elbow	Center of the bicep
8/9	Forearm	Elbow to wrist	Widest point of the forearm
10/11	Thigh	Up	Center of the thigh
12/13	Calf	Up	Widest point of the calf

Table 9.5: Definitions of circumferences by landmarks and the normal vector spanning the plane.

a model. Hence, we analyze the β parameters in the AGORA dataset with the aim to randomly sample more data with realistic body shapes. Histograms (see Figure 9.6) of the occurring β parameters show that their distribution roughly follows a normal distribution. Therefore, we train our models with randomly sampled data according to these distributions, either assuming a normal distribution fitted to the histograms or a uniform distribution with the same minimum and maximum values as in the data analysis. This means that we sample each β parameter according to the selected distribution, create the mesh according to the sampled values and derive the measurements with B2A. With this strategy, we can create a dataset with as many subjects as we need. As we do not expect the analyzed AGORA data to cover the full range of human body shapes, we also train with extended distributions, meaning that we increase the standard deviation σ to $\alpha_n\sigma$ in the case of a normal distribution or stretch the interval by a factor α_u in case of a uniform distribution.

9.4.2 Models

We use the same number of β parameters for each gender as used in the AGORA dataset, meaning 11 for male, 10 for female, and 16 for neutral subjects. With 36 measurements as input values and 10–16 output values for our A2B models, the dimensionality of the data is low. Therefore, we experiment with Support Vector Regression (SVR) and with small neural networks (NNs). We split the AGORA dataset in an 80% train, 15% test and 5% validation subset. For SVR, we additionally randomly sample $10,000$ subjects for training. We use a hyperparameter search based on the validation split to determine the optimal settings, which leads us to a radial basis function kernel, an error margin of $\epsilon = 0.012$ and a regularization constant of $C = 3791$. For the NNs, we randomly sample new data in each iteration. The hyperparameter search for the NNs results in a model with 4 layers, 330 neurons per layer, tanh as activation function, and Xavier Glorot as initialization. We use mean squared error on the model output (the β parameters) as training loss.

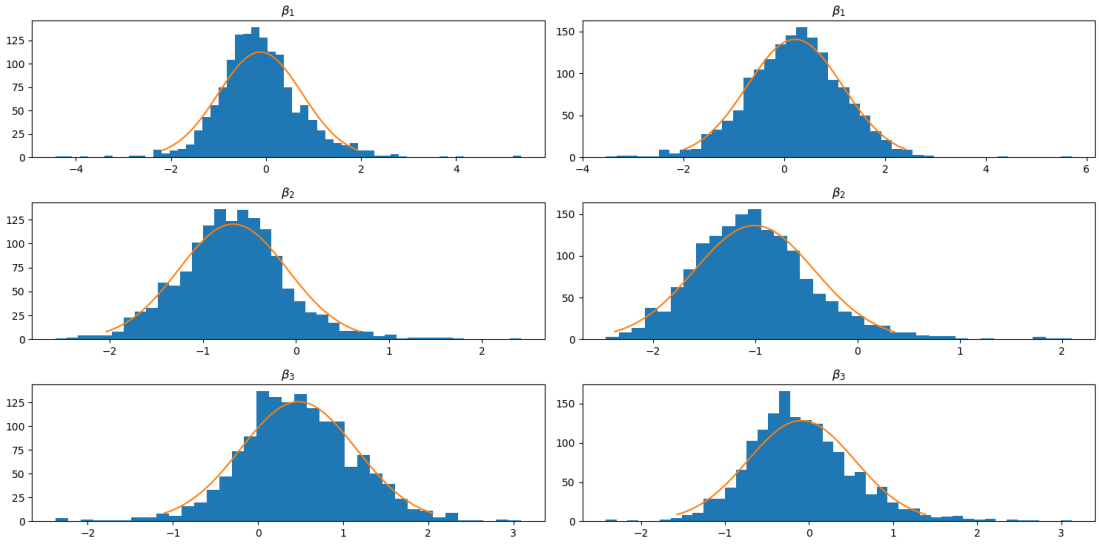


Figure 9.6: Histograms and fitted normal distribution (orange) for the first three β parameters for all male (left) and female (right) subjects of the AGORA [68] dataset.

9.4.3 Results

We train each model (NN and SVR) for each gender and with different dataset variants: We train solely on the AGORA train split, as well as on uniformly and normally distributed randomly sampled data according to the data analysis, and we further extend the range of the data as described in Section 9.4.1 with $\alpha_n = \alpha_u = 1.5$. The results are displayed in Table 9.6. We evaluate the performance of our models in two ways. At first, we calculate the error of the predicted and ground truth β parameters. Second, we calculate the mean deviation of the anthropometric measurements of the meshes from the predicted and ground truth β parameters (A). Therefore, this evaluation can further be seen as a kind of cycle consistency evaluation of A2B (our learned model) and B2A (the deterministic measuring function). Figure 9.7 provides a visualization of the evaluation scheme. The part that is also included in the training is highlighted.

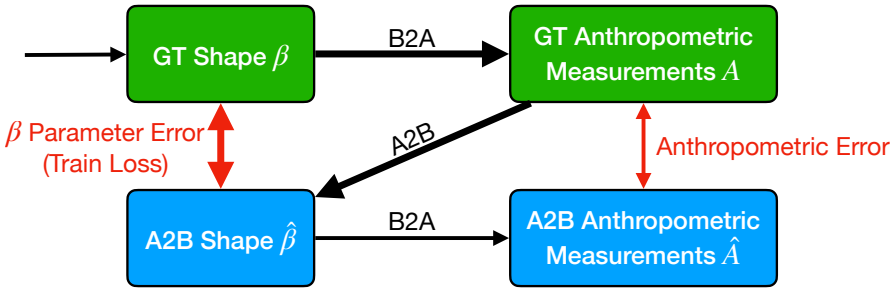


Figure 9.7: Visualization of the A2B evaluation and training procedures. The training part is highlighted with thicker arrows.

The anthropometric error A is our main metric as these values reflect the desired body shape given as an input by the user and are further interpretable. The β parameters are arbitrary in their scale. For all genders and SVR, using an extended uniformly sampled dataset works best. For the NNs, a uniformly sampled dataset works best for male (m) and female (f) genders and an extended normally sampled dataset for the neutral (n) meshes. The results for the neutral model are worse in general, especially in the case of the NNs, which might be due to the fact that the neutral model needs to express a more diverse range of body shapes. Furthermore, the SVR achieves better results for all genders. Thus, we use these models for all datasets, without any fine-tuning or adaptation to specific datasets.

		β-Error			A-Error		
Model	train data	m	f	n	m	f	n
NN	AGORA	9.11	13.9	24.0	0.814	0.934	1.459
NN	normal	2.62	4.34	18.0	0.356	0.392	1.711
NN	normal ext.	1.87	3.69	**14.8**	0.248	0.285	**1.384**
NN	uniform	5.08	**1.25**	18.3	**0.243**	**0.268**	2.381
NN	uniform ext.	**1.61**	3.20	16.3	0.274	0.419	1.774
SVR	AGORA	2.56	16.1	3.82	1.659	5.195	2.557
SVR	normal	4.08	17.8	59.0	2.975	4.303	14.63
SVR	normal ext.	0.210	4.60	6.27	0.280	1.090	2.211
SVR	uniform	0.0396	0.0350	**0.162**	0.124	0.284	0.214
SVR	uniform ext.	**0.0252**	**0.0193**	0.306	**0.082**	**0.136**	**0.164**

Table 9.6: Results of our A2B models on the test split of the AGORA dataset. The first block (β) shows the error if we take the ground truth β parameters, derive anthropometric measurements (B2A), input them into the A2B models and evaluate the MSE of the predicted β parameters in the scale 10^{-3}. The second block (A) calculates B2A from the predicted β parameters and evaluates the mean difference between the ground truth and predicted anthropometric measurements (all 36) in mm. Results are given for m(ale), f(emale), and n(eutral) models.

9.5 Leveraging A2B Model Results for HME

Now that we have trained the A2B models, we can use them to generate precise body shape parameters upfront and reuse them for every evaluation of a specific person. In the next section, we describe how the A2B results can be used to improve existing HME models (see Section 9.5.1). Further, we introduce a new approach to HME (see Section 9.5.2). We leverage the good performance of a sequence-based 2D-to-3D uplifting HPE model and convert the 3D stick-figure poses to human meshes with the help of our A2B models. With this approach, we achieve superior results compared to existing HME models. However, we want to emphasize that our approach is not exactly comparable to existing ones since it uses the additional information of anthropometric measurements.

Since the performance of existing HME approaches is not good enough to be used for sports analyses, our main goal is to achieve the best possible performance with marginal additional information. As professional athletes are measured anyway, this results in no actual overhead in these use cases.

We evaluate all models on ASPset [67]. This 3D human pose dataset consists of various different sports motion clips performed by different subjects and recorded from three camera perspectives. We evaluate on the test set, which contains two subjects and 30 videos for each subject. In the test set, only one camera perspective is public, so we evaluate on this perspective. Evaluating SMPL-X meshes for ASPset is non-trivial. Regressing standard SMPL-X joints from SMPL-X meshes is built-in, but for all other keypoint definitions it is necessary to define a custom regressor. Since there is no regressor available for ASPset, we create a custom SMPL-X regressor with the method presented by Russo et al. [75] for the keypoints head, head top, neck, l./r. hip, and torso. For left ankle and elbow, we mirror the corresponding right regressor, as it is not the exact mirrored version in the standard regressor.

We further evaluate on fit3D [26], since this is the only sports dataset with public SMPL-X annotations. Meshes and SMPL-X joints are evaluated since they are available. We select a subset of 37 SMPL-X joints. Since our focus is mainly on the body and not on the hands and face, we remove a lot of these joints. The used joints for the MPJPE calculation are pelvis, left hip, right hip, spine1, left knee, right knee, spine2, left ankle, right ankle, spine3, left foot, right foot, neck, left collar, right collar, head, left shoulder, right shoulder, left elbow, right elbow, left wrist, right wrist, left index, left thumb, right index, right thumb, left big toe, left small toe, left heel, right big toe, right small toe, right heel, right eye, left eye, right ear, left ear, nose. Further, we consider only the main body pose for Mean Vertex Error (MVE) calculation. This means that we ignore the specific pose of the hands and face, since our focus is mainly on the body and we do not estimate the hand and face pose in our approach.

Hence, we achieve a fair comparison with this evaluation scheme. For both datasets, we do not have access to the athletes to take their anthropometric measurements. Therefore, we simulate athlete measuring by measuring the ground truth meshes. Details can be found in Section 9.5.2.2. Since there is no ground truth available for the official test set evaluation on the evaluation server of fit3D, we split the official training dataset into a training, validation, and test set for our evaluations. We perform a leave-one-out cross validation and average the results.

Sports datasets differ from most commonly used everyday activity datasets in the aspect that the poses are more diverse and the motions are faster, which makes sports datasets more difficult. In some cases, the poses are so difficult that some models do not detect a human at all. This makes a fair evaluation hard, since the standard MPJPE metric takes the mean of the joint position errors. Assuming a default pose for all frames where no person is detected would result in a very high error that shifts the mean enormously. Hence, we report the MPJPE only on the frames where persons are detected. Since mostly difficult frames are omitted, this will result in a slightly easier setting for methods that find fewer persons, but we include the number of missing frames in our results for comparison.

	Model	orig. ↓	σ ↓	A2B ↓ NN m	SVR m	NN n	SVR n	no r. ↓
ASPset	SMPLer-X	86.0	2.9	78.9	78.5	**78.4**	78.5	0.11%
	OSX	92.3	0.2	89.6	**89.3**	89.4	89.6	0.10%
	Multi-HMR	102.5	3.6	100.0	100.3	**99.3**	99.5	0.44%
	SMPLify-X	138.2	13.0	127.7	127.4	**126.8**	126.9	0.02%
fit3D	OSX	94.2	3.9	88.9	88.6	**87.1**	87.2	3.45%
	Multi-HMR	74.6	3.3	69.6	69.6	**68.0**	68.4	1.54%

Table 9.7: MPJPE results in mm for existing models on the test splits of ASPset (top) and fit3D (bottom). The second column (*orig.*) contains the original results, the other columns show results with replaced β parameters from our **A2B models with pseudo ground truth anthropometric measurements** as input and either gendered (g) or neutral (n) meshes, and the percentage of frames with no result (no r.). The σ column displays the mean standard deviation of the body height per subject in cm for the original results, while all A2B body shapes have $\sigma = 0$.

9.5.1 Improving HME Model Results

A major problem for HME based analyses is a varying basic body shape within a single video. Existing HME models output different β parameters for each frame. Exemplarily, we show the standard deviation of the body height of one subject in Table 9.7. Recall that these measurements and β parameters are based on a T-pose mesh, hence varying poses have no influence on measuring and β parameters. Using β parameters generated with A2B models solves this problem. The necessary 36 measurements are either measured from the human directly as already done for most professional athletes, or real measurements are simulated as explained in Section 9.5.2.2. We combine existing HME models with the body shape estimated by our A2B models by replacing the estimated β parameters with the ones predicted by the A2B models. We select three recent well performing models on the AGORA dataset (SMPLer-X [7], OSX [49], Multi-HMR [2]), and the first HME method developed by the SMPL-X authors, SMPLify [69]. Since SMPLer-X is trained on the official training data of fit3D, an evaluation with this model is not meaningful, and we omit it here. Moreover, SMPLify-X is not state of the art anymore and achieved the worst results for ASPset. Therefore we omit it, too.

The first evaluation contains the original result from the respective model, and evaluations where the pose from the model is kept, but the β parameters are replaced with the A2B body shape parameters with pseudo ground truth input. Results are displayed in Table 9.7. The results for the MVE of the meshes for fit3D are included in Table 9.9. We can see that for all models, replacing the estimated β parameters by β parameters from our A2B models with pseudo ground truth input leads to an improvement. For one model, the gendered meshes outperform the neutral ones and for all other models, the neutral meshes perform best. We use the correct gender (male or female) of the subject in the gendered results in all tables of this chapter. Interestingly, the NN outperforms the SVR for all neutral experiments, although the SVRs achieved better results on the AGORA dataset evaluation. The reason could be that the difference in the anthropo-

	Model	orig. ↓	median ↓	A2B ↓				no r. ↓
				NN m	SVR m	NN n	SVR n	
ASPset	SMPLer-X	86.0	86.0	85.9	**85.7**	86.0	86.0	0.11%
	OSX	92.3	92.4	92.4	**92.2**	92.3	92.4	0.10%
	Multi-HMR	102.5	**102.1**	102.6	103.0	**102.1**	102.2	0.44%
	SMPLify-X	138.2	133.6	133.8	**133.5**	133.6	133.5	0.02%
fit3D	OSX	94.2	**93.0**	95.0	94.8	93.2	**93.0**	3.45%
	Multi-HMR	74.6	**73.9**	75.8	76.1	**73.9**	74.1	1.54%

Table 9.8: MPJPE results in mm for existing models on the test split of the ASPset (top) and fit3D (bottom) datasets. The second column contains the original results, the other columns results with replaced β parameters. Either the median β parameters are used or the results from our **A2B models with median anthropometric measurements from the respective model** as input. The percentage of frames with no result is mentioned in the last column (no r.).

metric error between the models is very small and not significant for downstream tasks. SMPLer-X achieves the best results for ASPset and Multi-HMR for fit3D, both with a significant margin. OSX performs worse on fit3D than on ASPset, but Multi-HMR performs better by a large margin and surpasses OSX. All methods benefit from our A2B β parameters based on pseudo ground truth with MPJPE improvements from 11 mm to 3 mm regarding both datasets and MVE improvements of approx. 8 mm regarding fit3D.

Although it is not our main goal, we further evaluate the capabilities of a fixed body shape without available ground truth measurements to ensure consistent body shapes in the case that no measurements are available. The simplest approach is to use the median of the β parameters across all frames of the respective model. However, the β parameters have no real meaning. Therefore, we compare this approach to taking the median of the anthropometric measurements of the generated meshes and then converting them to β parameters via the A2B models. Results are displayed in Table 9.8. For SMPLer-X and OSX, using the median β parameters lead to equal or even worse results on ASPset. Regarding ASPset, using our A2B models increases the performance of all models slightly. Switching from the neutral output that these models all have to a gendered model works best in most of these cases, but the neutral A2B models also lead to a marginal improvement. Regarding fit3D, using the median β parameters already enhances the MPJPE and MVE results. Using β parameters from an A2B model leads to the same improvement for both metrics, OSX achieves the best results with SVR and the neutral model, Multi-HMR with NN and the neutral model. Hence, our A2B models can be used to improve the results of existing models in most cases.

Anthropometric measurements can further be used to easily convert between neutral and gendered (male or female) models. In contrast, β parameters are not transferable between models of different genders. Using the same β parameters for a male and neutral model would result in a completely different body shape. Therefore, until now, the conversion could only be achieved by minimizing the MVE between meshes of different

genders in an iterative process and obtaining the new β parameters with the least MVE after the process finishes. Now, we can use the B2A function to obtain measurements for a mesh of one gender and apply the A2B model of the other gender to these anthropometric measurements in order to get the corresponding β parameters for the other gender.

9.5.2 HME with Sequence Based 3D HPE and A2B

All evaluated HME models are working image-wise. In contrast, state-of-the-art 3D HPE models obtain 2D poses from all video frames and take long sequences of 2D poses as an input, which helps to capture movements precisely. The models are called uplifting models, since they lift 2D pose sequences to 3D pose sequences. We use the efficient state-of-the-art 3D HPE model *uplift and upsample* (UU) [21] to estimate the 3D poses on videos. To estimate the required 2D poses from the video frames, we use ViTPose [98], a state-of-the-art 2D pose estimation model. It is important to note that UU operates on pose sequences instead of single frames like the HME models in Section 9.5.1 and can leverage the information of neighboring frames to estimate a more precise pose. Since we have ground truth 3D joints available, we can fine-tune the models (ViTPose for 2D HPE and UU for 3D HPE) on our data. This is also necessary to adapt the model to the dataset specific joint definitions since many 3D HPE models like UU are pretrained on datasets like Human3.6M [38], but those joint definitions do not match ASPset nor fit3D. We fine-tune both 2D and 3D HPE models on the training subsets. On the test subsets, UU achieves an MPJPE of 63.85 mm on ASPset and an MPJPE of 29.60 mm on fit3D, which is better than the best existing HME model for both datasets (see Section 9.5.1). However, UU only outputs 3D joints, no meshes. Moreover, a stick-figure 3D pose is not sufficient to model the pose parameters θ of the SMPL-X mesh, since some rotations are missing. Hence, it is impossible to calculate the necessary rotation parameters directly from the UU result.

9.5.2.1 Inverse Kinematics for Full Pose Estimation

Therefore, we use the well established approach of inverse kinematics (IK) with a pose prior to obtain the missing rotations by fitting an SMPL-X mesh to the 3D joint locations estimated by UU. IK outputs the best SMPL-X parameters (β and θ) that fit the mesh to the given k 3D joint locations $p^{3\mathrm{D}}$. Hence, IK can be formally defined as the following:

$$\mathrm{IK}(p^{3\mathrm{D}}) = \beta, \theta \quad \text{with} \quad \beta, \theta = \arg\min_{\beta,\theta} \mathcal{L}_{\mathrm{IK}}(p^{3\mathrm{D}}, \beta, \theta). \tag{9.1}$$

The output of the IK algorithm is a solution to the optimization problem, which is the best fitting SMPL-X mesh to the given 3D joint locations under additional constraints. Finding the optimal solution is achieved by a gradient descent minimization approach, since the SMPL-X body model is a linear model.

We use the inverse kinematics approach with a VPoser extension, as proposed in the code by Pavlakos et al. [69]. VPoser is a learned prior for human poses, since

the raw SMPL-X model definition allows impossible poses for humans. VPoser has learned plausible poses from the large AMASS [63] dataset and helps IK to generate only plausible poses. IK finds the best SMPL-X parameters (β and θ) that fit the mesh to the given 3D joint locations by minimizing the error $\mathcal{L}_{IK}$ which consists of three parts:

$$\mathcal{L}_{IK} = \lambda_1 \mathcal{L}_{joint} + \lambda_2 \mathcal{L}_{VPoser} + \lambda_3 \mathcal{L}_{\beta}. \tag{9.2}$$

The first loss term, $\mathcal{L}_{joint}$, calculates the distance between the given joint locations and the regressed joint locations from the mesh. Let $m = M_{\text{SMPL-X}}(\beta, \theta)$ be the SMPL-X mesh and $p^m = r \cdot m$ the regressed 3D pose from the mesh. Then,

$$\mathcal{L}_{joint} = \sum_{j=1}^{k} \left\| p_j^{\text{3D}} - p_j^m \right\|^2. \tag{9.3}$$

The other two loss components, $\mathcal{L}_{VPoser}$ and $\mathcal{L}_{\beta}$, represent the sum of squared values of the VPoser and β parameters, respectively. Penalizing the β parameters in this manner is reasonable, as they are designed in the SMPL-X model to follow a normal distribution centered at zero. Consequently, values closer to zero are statistically more probable, which is enforced by the loss to encourage realistic body shapes.

VPoser is an autoencoder that encodes pose parameters θ into a lower-dimensional latent space. The original θ parameters, represented in an axis-angle format, describe 3D rotations for 21 joints, resulting in a 63-dimensional vector. In contrast, the VPoser latent space has 32 dimensions and is also normally distributed. Therefore, a lower sum of squared elements in the VPoser latent space indicates a pose that aligns more closely with the learned distribution.

The weighting factors are set to $\lambda_1 = 10$, $\lambda_2 = 0.0007$, and $\lambda_3 = 0.01$ in our experiments. We use relatively low values for λ_2 and λ_3, since sports datasets incorporate extreme poses and our main interest is to achieve the most perfect pose.

We execute IK per frame, which results in a slight jitter between the frames, but leads to more accurate joint positions. Since IK needs multiple iterations to adjust the standard T-pose parameters to achieve a pose that is roughly close to the desired UU pose, we speed up the process by initializing the pose and shape parameters with the result from the previous frame if available. This also enhances the final result slightly. We acknowledge that IK is relatively slow regarding the runtime, but our main focus is the precision. For sport analysis, which is our focus, the runtime is not critical, but a very precise result is crucial.

9.5.2.2 Generation of Ground Truth Shape Parameters

As we do not have access to the athletes of ASPset and fit3D to obtain real anthropometric measurements, we need an alternative to simulate this process. For ASPset, as a first step, we run IK on the ground truth 3D joint locations. From the generated meshes, we obtain the necessary anthropometric parameters with B2A. Then, we use the median values of these measurements as the ground truth anthropometric values. We call these parameters *pseudo ground truth* throughout this chapter, since this is not directly the

ground truth, but obtained from IK executed on the ground truth 3D joint locations and the B2A computation from the created meshes. These parameters are used in this chapter to generate the pseudo ground truth β parameters by A2B prediction.

We do not have access to the athletes of the fit3D dataset either. Therefore, we need ground truth data to mimic measurements. Obviously, there is no ground truth available for the official test set evaluation on the evaluation server. We therefore split the official training dataset into a training, validation, and test set for our evaluations. We perform a leave-one-out cross validation, therefore all eight athletes from the official training dataset are used in our evaluation. With this selection, we have real ground truth shape parameters available. We do not use these directly, since this would skip the measuring process that is needed in real applications. Further, the ground truth data is not consistent (see Section 9.3). Therefore, we apply B2A and use the median measurements over time in order to simulate the measuring process and obtain a single set of anthropometric measurements per person. In real applications, this step is omitted because the anthropometric parameters can be measured directly from the athletes before starting the recording.

We consider this strategy as a valid method for evaluations, since our main goal is to improve the HME performance as much as possible with only marginal overhead. Our main focus is sports, which contains extreme poses that let existing HME models fail, sometimes even to detect a human at all. As professional athletes are measured anyway, the additional effort for the measurements is negligible in this context.

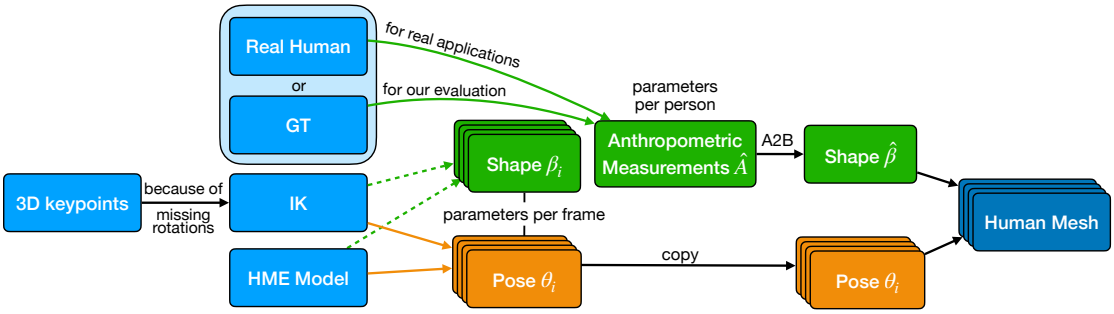


Figure 9.8: Overview of our inference pipeline. The pose and shape parameters are obtained either from IK applied to UU results (Sec. 9.5.2.3) or from an HME model (Section 9.5.1). In real applications, the anthropometric measurements will be taken directly from the humans. For our evaluations, we use the ground truth shape parameters and further experiment with the shape parameters of the respective model (IK or HME).

9.5.2.3 Experiments and Results

We evaluate different experiments in Table 9.9. For comparison, we mention the UU 3D HPE performance (first rows for each dataset in Tab. 9.9). These results correspond to stick-figure poses and not the required meshes. Therefore, they are not directly comparable to the other results.

DS		*inconsistent shape* pose	orig.	σ	*consistent shape (ours)* measure/shape	NN g	SVR g	NN n	SVR n	median	σ	no r. ↓
ASPset	1	UU	63.9		no mesh							
	2	IK-UU	67.5	3.0	ground truth	56.4	56.6	**55.1**	55.2	-	0.0	0.0%
	3	IK-UU	67.5	3.0	IK-UU	66.9	66.6	67.3	67.1	67.2	0.0	0.0%
	4	SMPLer-X	86.0	2.9	ground truth	85.9	85.7	86.0	86.0	-	0.0	0.11%
fit3D	1	UU	34.3	-	no mesh							
	2	IK-UU	38.5 / 46.3	8.7	ground truth	41.2 / 47.5	41.3 / 46.9	38.8 / **45.3**	**38.7** / **45.3**	-	0.0	0.0%
	3	IK-UU	38.5 / 46.3	8.7	IK-UU	42.6 / 51.2	41.6 / 48.6	39.8 / 47.9	39.8 / 47.8	39.9 / 47.8	0.0	0.0%
	4	Multi-HMR	74.6 / 76.1	3.3	ground truth	69.6 / 67.8	69.6 / 67.6	68.0 / 67.9	68.4 / 68.8	-	0.0	1.54%
	5	Multi-HMR	74.6 / 76.1	3.3	Multi-HMR	75.8 / 77.3	76.1 / 76.7	73.9 / 75.6	74.1 / 75.8	73.9 / 75.5	0.0	1.54%
	6	OSX	94.2 / 89.0	3.9	ground truth	88.9 / 83.4	88.6 / 82.3	87.1 / 81.2	87.2 / 81.1	-	0.0	3.45%
	7	OSX	94.2 / 89.0	3.9	OSX	95.0 / 91.7	94.8 / 90.2	93.2 / 87.8	93.0 / 87.6	93.0 / 87.6	0.0	3.45%

Table 9.9: MPJPE results in mm on the test splits of ASPset (top) and fit3D (bottom) of our approach compared to the respective best HME model(s). For fit3D, we calculate the MVE in mm, too, since we have ground truth meshes available. We display it as the second value in every column. The *pose* column indicates the origin of the pose. The *orig* column contains the result as it is estimated from the method indicated in the *pose* column (with inconsistent body shapes). The right block contains the results with the originally estimated β parameters replaced by consistent ones. The *measure/shape* column indicates which anthropometric measurements are used for the A2B computation (i.e., which β parameters are used for the median computation) whose results are the replacement β parameters in the last five columns. We highlight the overall best results for *estimated* meshes with *consistent shapes* in bold and underline the best (MPJPE and MVE) results in each line. We further add the mean standard deviation of the body height and the percentage of frames with no result as in Table 9.7.

Our main approach is shown in the second rows in Table 9.9. We evaluate the results of IK applied to the UU joint locations with original, median, and pseudo ground truth based A2B β parameters. The real-world scenario corresponds to the following: Ground truth measurements can be measured from the athlete directly and the 3D pose can be estimated with UU and IK. The β parameters are estimated with the A2B models. A visualization of this pipeline can be found in Figure 9.8.

We include the best result(s) from existing HME models in the respective last rows for comparison and provide qualitative results in Figure 9.1, 9.9, and 9.10. Our approach outperforms the best existing HME model for both datasets by a large margin. But we

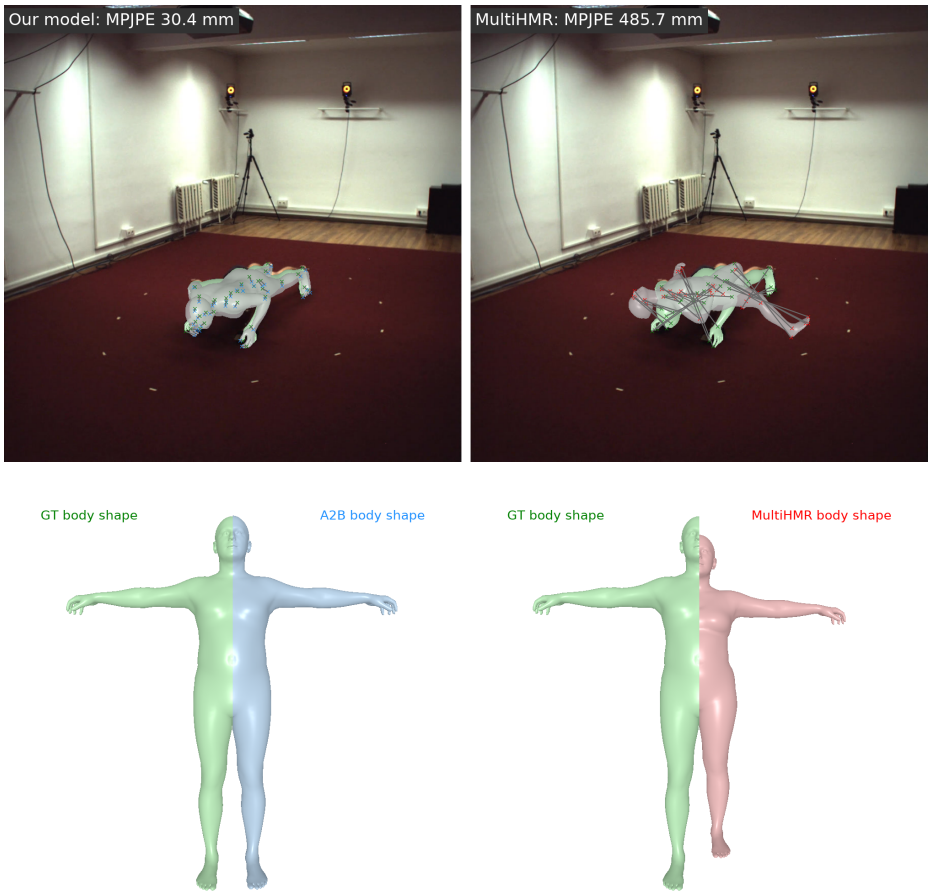


Figure 9.9: Example detection result of our approach compared to Multi-HMR for fit3D. In the upper row, we display the ground truth meshes in green and the estimated meshes in gray. The ground truth joints are also displayed in green while the estimated joints from our model are visualized in blue. The Multi-HMR joints are shown in red. Corresponding joints are connected. We display the exact MPJPE values in the top left of each frame. In the lower part, we show the estimated body shapes in T-pose. The ground truth body shape is shown in green and the estimated body shape from our model in blue. The Multi-HMR body shape is shown in red.

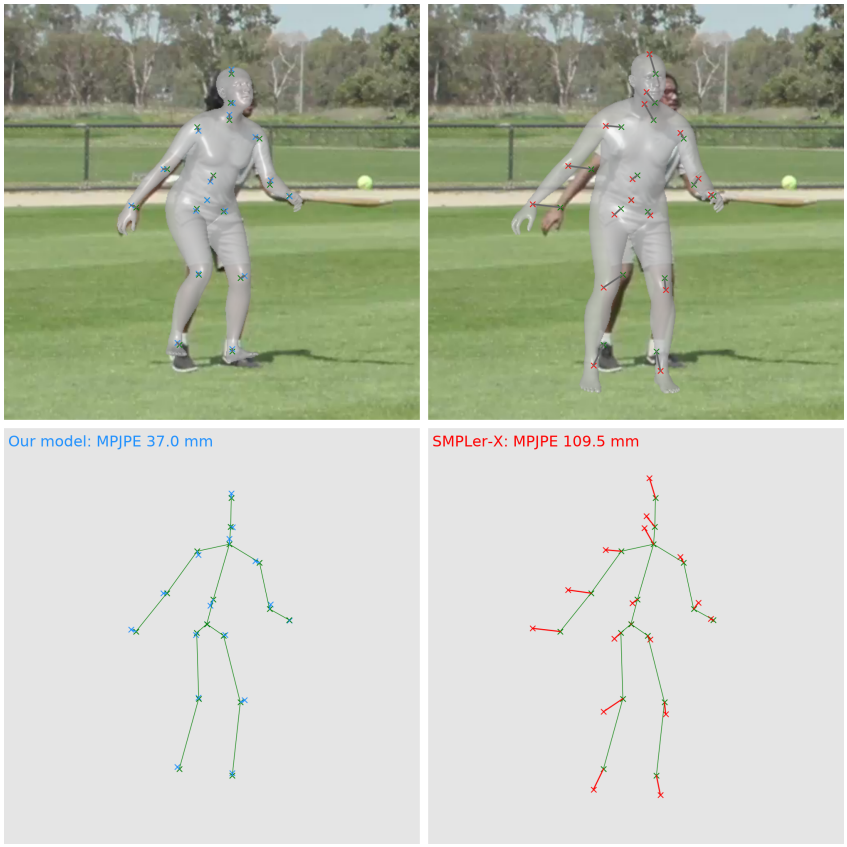


Figure 9.10: Qualitative results of SMPLer-X and our approach for an example frame from ASPset. We display the estimated meshes and the ground truth and estimated joints. Ground truth joints are shown in green, estimated joints from our model in blue and the SMPLer-X joints in red. Corresponding joints are connected. In the lower part, we show the ground truth and estimated joints in the same way, but without the mesh and image to reduce distraction. We further display the MPJPE values.

need to mention that our approach is not directly comparable to the original results of the HME models, since our approach needs the additional information of measurements, but they already exist in our scenario. However, our approach still outperforms the existing HME models even when they use the same measurements and A2B results as our model does. Further, our model provides results for all frames, while the other HME models fail to detect a human in some frames and therefore do not output a mesh at all. See the last column (no r.) in Tables 9.7, 9.8, and 9.9.

We analyze the results of the building blocks of our model in detail. Applying IK to the UU results deteriorates the UU results by nearly 4 mm for ASPset and 5 mm for fit3D (see Tab. 9.9, first and second rows, column *orig.*), but this step is necessary since the UU result is only a stick-figure pose and not sufficient for our purpose. Moreover, these results are still better than the best existing HME model (see last rows in Tab. 9.9).

Next, we replace the inconsistent β parameters with the results from our A2B models. This is especially helpful for our approach since IK produces body shapes with high inconsistencies, as shown by the larger standard deviation of the body height compared to other HME models. For ASPset, using pseudo ground truth measurements results in a large improvement of over 12 mm. Remarkably, this result surpasses even the original UU result by 8 mm. It seems that incorporating a clearly defined mesh helps to fix some typical errors of UU and enhances its result in case of ASPset. In general, the error on fit3D is much lower for UU based approaches. The reason might be that it consists of much more data, such that we can fine-tune UU for a longer time. Further, the videos are recorded in a lab in comparison to the in-the-wild videos of ASPset. The lab environment is very similar to the Human3.6M dataset [38], which serves as a training dataset for most recent HME models. Therefore, the results of ASPset are more relevant for future applications of our approach, where we assume only a few available 3D annotations and in-the-wild recordings. For fit3D, applying the A2B body shapes from pseudo ground truth measurements leads to a slight decrease in performance of 0.2 mm. Inconsistent shapes in the ground truth (see Section 9.3) are likely to cause this behavior. Still, our approach using a 3D HPE model and IK outperforms all existing HME models, no matter if the original inconsistent or the consistent body shapes from A2B are used.

Regarding the gendered meshes, we observe that the performance is slightly better for male than for female subjects. fit3D consists of two female and six male subjects. The best score of 40.2 mm for the male subjects is achieved with the SVR model. For the female subjects, the best score is 42.1 mm with the NN model.

As described in Section 9.5.1, we further evaluate the capabilities of a consistent shape without available ground truth measurements. The naive approach is to use the median of the estimated inconsistent β parameters (Tab. 9.9, column *median*). Another approach is to use the meshes created by IK applied to the UU results, compute the anthropometric measurements with B2A, calculate the median measurements and convert them to β parameters via the A2B models. Results are displayed in Table 9.9, row three. For ASPset, using fixed body shape parameters from A2B models based on the measurements from UU results achieves a slightly better score than the results with inconsistent body shapes. For fit3D, the MPJPE increases by 0.9 mm, but the A2B model results are a slightly better alternative for consistent body shapes compared to the median β parameters.

9.5.3 Fine-Tuning Existing HME Models

Fine-tuning existing HME models on pure 3D joints datasets is not possible, since they need mesh annotations for training. However, we can generate pseudo ground truth meshes by applying IK to the ground truth 3D poses. We exemplary test a fine-tuning of SMPLer-X on ASPset with this approach. Using their fine-tuning script with 1.6M iterations leads to worse results than the results without fine-tuning. We find that the fine-tuning faces heavy overfitting. Therefore, we reduce the number of iterations with early stopping and achieve better results with fine-tuning only for 32K iterations.

The results shown in Table 9.10 prove that fine-tuning on IK generated meshes can lead to a significant improvement of the scores. Replacing the β parameters of the fine-tuned results with the ground truth A2B β parameters boosts the performance even more. These are the best results achieved with any existing HME model throughout this study, although still worse than the results with our approach.

Moreover, we experiment with using the SMPLer-X body shape parameters with the poses estimated by IK applied to the UU results (see last two rows of Table 9.10). Using the β parameters from SMPLer-X leads to a slightly better result than the original 3D joint based result (without IK). This evaluation shows that 3D HPE models are better in precisely locating the joints of humans than HME models, and HME models are better in estimating the shape of humans. We also try to use the β parameters of the fine-tuned variant for the UU IK results like before. However, this resulted in a performance drop compared to the body shape parameters from the original SMPLer-X without fine-tuning. These experiments show that fine-tuning HME models on pseudo ground truth leads to a better performance regarding the keypoints, but the estimated β parameters have worse quality. This can further be proven by replacing the β parameters from the fine-tuned SMPLer-X variant with the β parameters from the not fine-tuned model, which results in a performance gain of over 5 mm compared to the original results from the fine-tuned version (rows 2 and 4 in Table 9.10).

9.5.4 Overview of the Results

We have executed a multitude of experiments with different combinations of pose and shape parameters. Figure 9.11 summarizes the results with their pose and shape origins for ASPset. In general, the poses estimated by IK based on the UU results (red branch in Fig. 9.11) are more precise than the poses estimated by SMPLer-X (light blue branch in Fig. 9.11). Further, the body shape parameters from our A2B models with ground truth anthropometric measurements as they already exist for professional athletes (green boxes

	Model	shape/measure	orig.	median	A2B			
					NN m	SVR m	NN n	SVR n
1	SMPLer-X	SMPLer-X	86.02	86.04	85.89	85.69	86.03	85.99
2	SMPLer-X FT	SMPLer-X FT	79.09	79.44	78.92	78.88	79.44	79.37
3	SMPLer-X FT	ground truth	-	64.79	65.63	65.84	**64.71**	64.76
4	SMPLer-X FT	SMPLer-X	-	73.66	73.41	73.29	73.65	73.63
5	UU IK	UU	67.54	67.16	66.92	66.60	67.25	67.12
6	UU IK	SMPLer-X	-	63.82	63.80	**63.64**	63.81	63.78
7	UU IK	SMPLer-X FT	-	69.69	69.46	69.27	69.70	69.63

Table 9.10: MPJPE results in mm for different methods and β parameters on the test split of ASPset. SMPLer-X FT stands for the best fine-tuned variant of SMPLer-X (fine-tuned with the meshes obtained from IK executed on the ground truth 3D joints). The *orig* column contains the results without replaced β parameters.

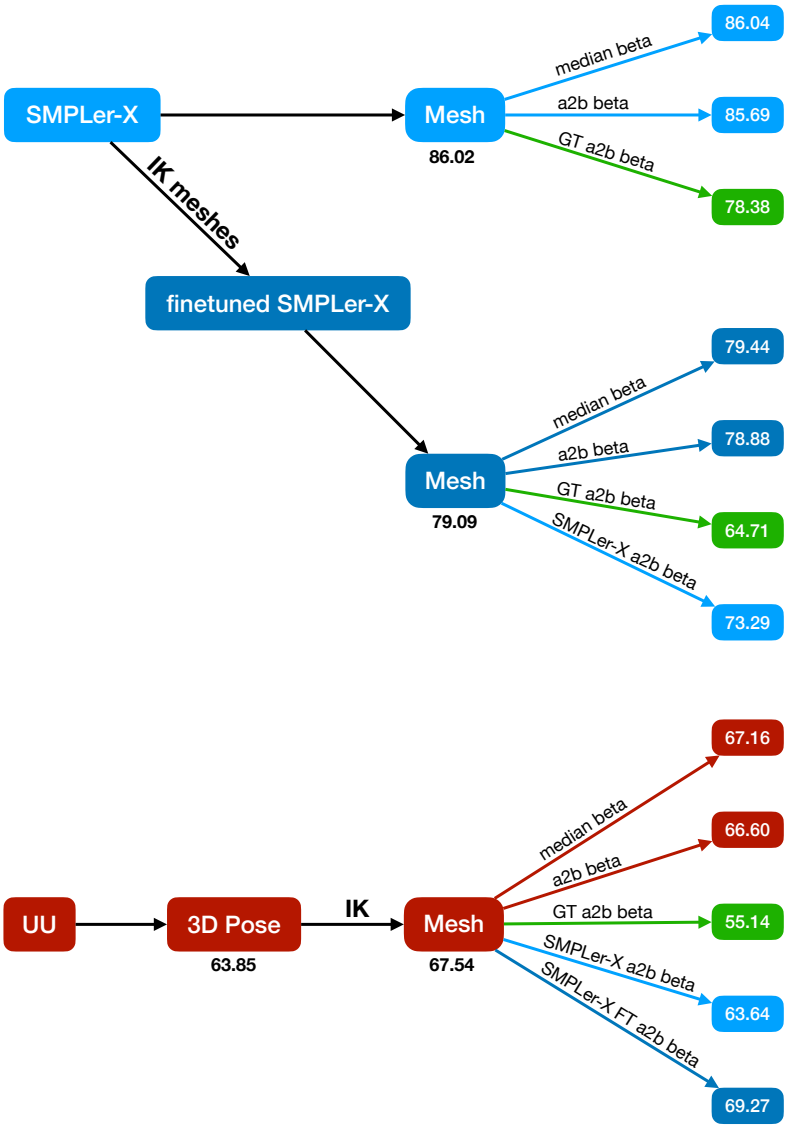


Figure 9.11: Overview of the main results for the ASPset dataset. All results are MPJPE results in mm. Results below "Mesh" boxes show the result with the original β parameters. All results after arrows to the right are results with replaced β parameters. The type of the β parameters is noted on the arrow and is color-coded.

in Fig. 9.11) achieve the best results for all poses. Without access to the ground truth, all models benefit slightly from A2B model results with the median anthropometric measurements from B2A of the estimated meshes by the respective model (boxes with same color for all three branches in Fig. 9.11). Moreover, SMPLer-X A2B body shape parameters perform best when analyzing body shapes without ground truth access (light blue boxes in Fig. 9.11). Fine-tuning SMPLer-X with IK created meshes (dark blue

branch in Fig. 9.11) improves the performance of SMPLer-X, although the quality of the body shape deteriorates. This can be seen as by comparing the shapes from SMPLer-X and fine-tuned SMPLer-X (dark blue and light blue boxes in Fig. 9.11) with fine-tuned and IK poses.

Since fit3D is a larger dataset, fine-tuning UU works better, which further leads to better IK meshes with an MPJPE of 37.02 mm. Enforcing consistent meshes with ground truth or IK A2B shape parameters decreases the performance slightly in this case. However, A2B shape parameters achieve slightly better scores than median values. This also holds for OSX and Multi-HMR. Overall, the approach with UU, IK, and A2B body shape parameters reduces the MPJPE by 33 mm over any (tested) SOTA HME model.

9.6 Summary

We addressed the problem of inconsistent estimated basic body shapes of humans in videos in this chapter. We analyzed the ground truth data of 3D pose and mesh datasets and found inconsistencies in their annotations. We proposed a family of learned *A2B* models to convert 36 anthropometric measurements to SMPL-X β parameters. This can be used to measure a human once (as it is established practice for athletes, our main focus) and use the resulting shape of the A2B model for all following evaluations. With this strategy, the body shape is accurate and consistent per person. Evaluations have shown that using IK on the results of a state-of-the-art 3D HPE model to estimate the mesh pose combined with our A2B model's shape parameters leads to superior and consistent results compared to existing HME models. Moreover, HME models also benefit from our approach. Replacing their estimated shape parameters with the A2B shape parameters leads to an improvement of their score and consistent body shapes. However, our approach based on 3D HPE still outperforms these improved scores.

10 Conclusion and Outlook

We conclude this work with a brief recap of the main aspects and contributions of all parts. Additionally, we relate current and future research areas to the topics of this thesis.

10.1 Conclusion

This thesis is founded on computer vision techniques to ease or enable performance analyses of athletes in various sports disciplines. Many analyses are very time-consuming or just infeasible if executed manually. Therefore, this thesis presents a set of methods to automate these analyses, making them accessible to more athletes and making it possible to introduce new kinds of analyses. Since sports is a domain with very challenging datasets, we focus at first on creating well-performing 2D HPE models in such scenarios, including low image quality or a low number of annotated images. We further extend these models to be able to detect more than just a set of standard keypoints. Our new models are able to detect any point on the human body, including knowledge about the body boundary, bent limbs, etc. However, these points are still in 2D, limiting their usability. Therefore, we propose a method to estimate the 3D mesh of humans which additionally preserves a consistent core body shape over time, since this is crucial for sports analyses.

The first part of this thesis focused on 2D HPE with imperfect data. In Chapter 3, we have introduced a method to robustly estimate relevant flight angles for ski jumpers during their flight phase. To analyze the performance of their athletes, ski jumping coaches place several cameras along the ski jumping hill. For each camera view, the relevant flight angles are calculated based on a small amount of frames which show the athlete, since their speed is high. We automated this process, making this analysis accessible to all ski jumpers using that hill. Our approach incorporates a CNN trained on the relevant keypoints of ski jumpers and their skis. We further apply heuristics to filter our obviously wrong poses and use RANSAC to achieve a robust estimate of the flight angles. This process is especially challenging since the image quality is low due to bad resolution, lighting, weather, motion blur, etc. In cooperation with the Institute of Applied Training Science (IAT) in Leipzig, we improved our model so far that the Olympic Ski Jumping Team of Germany can now use our system in their daily business. The ski jumping coaches already performed such analyses manually in the past, which lead to the advantage that this data could be used as labels to train our model. However, for most sports disciplines, no labels are available at all. Therefore, we investigated in Chapter 4 the usage of semi-supervised learning techniques to train usable 2D HPE models on a small amount of labeled data for sports disciplines. We have shown that

with 50 labeled images, we can achieve a reasonable performance of 83.8% PCK@0.1 compared to 91.9% with the fully supervised training on our main dataset with triple and long jump athletes. Our best approach is an iterative self-training mechanism. At first, pseudo-labels are generated for all unlabeled images with a baseline model solely trained on the few labeled images. We filter the pseudo-labels to train only on the most reliable ones. Then, we train a new model on the pseudo-labeled images and the labeled images until convergence and repeat this process multiple times. Moreover, we evaluate a mean teacher approach, which consists of a single training with alternately using labeled and unlabeled images. Pseudo labels for the unlabeled images are generated on the fly by the teacher, which is an exponential moving average of the student model. This approach works as well, although slightly outperformed by the first approach.

In the second part of this thesis, we gradually improved a 2D HPE model to detect arbitrary keypoints on the human body. In Chapter 6, we introduced a method to detect keypoints on the line between two standard keypoints and in Chapter 7, we extended this approach at first to points on the limbs, then to points on skis of ski jumpers, and eventually to all points on the human body. We could prove in experiments that our approach is able to detect such points. Moreover, we introduced a new metric such that we can evaluate if the model can estimate the right distance from the boundary of the body, which is the case for our approach. Since this method requires segmentation masks for label generation, we additionally proposed methods to deal with only few segmentation masks and with only partly correct masks. Our model is designed based on a Transformer architecture. We leverage the ability of Transformers to deal with sequences of arbitrary length to train our model on an arbitrary number of keypoints. The users can define which keypoints they require for their analyses in a human-readable way and pass all these queries to the model. The model learns how to interpret these queries, and we ensure by using a special attention design that the output of each query is independent of the other queries processed at the same time. This way, our model produces robust results no matter the input.

Having a full overview of the human body empowers coaches to create more sophisticated analyses. However, these points are two-dimensional, which is not sufficient for some analyses. Therefore, we focused in the third part of this thesis on estimating three-dimensional meshes of the human body surface. While analyzing 3D HPE and HME datasets, we found that for most of them, the ground truth data is inconsistent. Human meshes are mostly represented by a separate pose and body shape, and the latter should be consistent for the same person across all frames of a video or multiple videos in a short time frame. However, this is not the case for many datasets. Furthermore, existing well-performing HME models do not estimate consistent body shapes. Therefore, we proposed a novel approach to HME in Chapter 8 that ensures consistent body shapes with marginal overhead for sports applications. At first, we introduced novel small models that can convert between anthropometric measurements and the body shape parameters of the common SMPL-X body model. Next, we leveraged the state-of-the-art 3D HPE model Uplift and Upsample to estimate precise 3D poses from sports videos. Inverse kinematics helped us to extract the full body pose out of the insufficient 3D stick-figure pose. By combining this pose with the body shape estimated from anthropometric mea-

surements using our model, we achieved superior results compared to state-of-the-art HME models and further ensured the consistency of the body shape.

10.2 Outlook

Finally, we highlight currently ongoing and potential future research directions that are related to the main topics of this thesis.

The research in the field of 2D HPE is still evolving, but it has reached a performance level which is hard to improve further. For a long time, CNN based models were the state of the art, but recently, Transformer based models have shown to be superior [98, 28]. The drawback of Transformer networks requiring huge amounts of training data could be addressed by using self-supervised representation learning tasks such as masked image modeling, making it feasible to pretrain large Transformers [98]. With such pretrainings, they can be trained efficiently for 2D HPE tasks, making it unnecessary to use techniques to reduce parameters like token clustering or token pruning [101, 59]. Research further focuses on alternative ways to represent the final keypoint detections. Instead of using the most popular 2D heatmaps, more fine-grained 1D heatmaps [47], direct regression [43], and more recently lookup tables [28] have been explored. Semi-supervised learning techniques are further investigated to train 2D HPE models on few labeled images [36]. Since 2D HPE models serve as an input for 2D to 3D uplifting models, which is the current state of the art for 3D HPE, the quality of the 2D HPE models is crucial for the final 3D pose estimation. Therefore, the research in 2D HPE is still highly relevant.

Focussing on more than just standard keypoints has proven to be effective in recent 3D HPE and HME research [76, 61]. It has shown to be successful to estimate human meshes based on a learned set of intermediate markers [61]. The position of these markers is learned until now. Our approach for estimating arbitrary keypoints enables the extraction of a set of dense 2D keypoints on the human body. These keypoints could potentially be used to train a 2D to 3D mesh uplifting model with these dense keypoints as an input. Uplifting methods have shown to be successful in 3D HPE [21, 82], but are yet not explored for HME. Most HME methods are operating on single images, although sequence based methods show significant advantages. With dense 2D keypoints, the spatial information of the human body is preserved, which could be leveraged by 2D to HME models.

Our work on HME achieves superior results in challenging scenarios with consistent body shapes, but is rather inefficient regarding the runtime. IK is an iterative process which has to be run for each image, which is well-known to be slow. We need IK to extract the full body pose from the 3D stick-figure pose, which is estimated by a 3D HPE model, and discard the body shape that IK estimates. A more efficient approach would be to train a model to estimate the full body pose directly from the 2D pose sequence. In order to achieve a high precision for such a model, the 2D keypoints have to be chosen carefully. Since large motion capture datasets like AMASS [63] contain the full body pose, they can be leveraged to pretrain such a model, similar to the approach Uplift and Upsample [21]. This could also help to remove the small jitter in the estimated 3D

meshes, which is caused by the IK process running per frame. Removing such a jitter by enforcing physical consistency is further not yet explored in the field of HME and could be further investigated for 3D HPE. Recent research has shown promising results by leveraging a physics simulator for motion augmentation [62] and as a building block for 3D HPE [30]. Especially in the field of sports, where human motions contain fast, complex and sometimes unusual movements, such a physics simulator could be beneficial.

The breakthrough of large language models (LLMs) is increasingly influencing human pose and mesh estimation tasks. While less precise than anthropometric measurements, linguistic attributes can effectively describe human shapes and enhance the accuracy of estimated meshes [15]. Additionally, LLMs have demonstrated promising capabilities in estimating keypoint coordinates by relying solely on textual descriptions of these keypoints [90]. This approach enables the retrieval of any visually describable keypoint, aligning with the methodology outlined in the second part of this thesis. Beyond keypoint estimation, LLMs can be utilized to generate meshes in specified poses, identify poses of described individuals in images, interpret poses represented by meshes, compare poses to highlight differences, propose adjustments needed to transition between poses, and edit existing poses based on textual descriptions [25, 60, 46]. These multimodal LLMs integrate language and vision modules, including specialized pose estimation building blocks, which are jointly trained. Further exploration of such approaches holds significant potential for advancing detection accuracy in challenging scenarios such as sports, and for enabling innovative motion analysis applications for athletes.

List of Figures

List of Tables

Bibliography

[1] M. Andriluka, L. Pishchulin, P. Gehler, and B. Schiele. 2d human pose estimation: New benchmark and state of the art analysis. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014.

[2] F. Baradel, M. Armando, S. Galaaoui, R. Brégier, P. Weinzaepfel, G. Rogez, and T. Lucas. Multi-hmr: Multi-person whole-body human mesh recovery in a single shot. In *European Conference on Computer Vision*, pages 202–218. Springer, 2025.

[3] T. Baumgartner and S. Klatt. Monocular 3d human pose estimation for sports broadcasts using partial sports field registration. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5109–5118, 2023.

[4] Y. Bin, X. Cao, X. Chen, Y. Ge, Y. Tai, C. Wang, J. Li, F. Huang, C. Gao, and N. Sang. Adversarial semantic data augmentation for human pose estimation. In *European Conference on Computer Vision*, pages 606–622. Springer, 2020.

[5] D. Bojanic. Smpl-anthropometry. https://github.com/DavidBoja/SMPL-Anthropometry/, 2023.

[6] Y. Cai, Z. Wang, B. Yin, R. Yin, A. Du, Z. Luo, Z. Li, X. Zhou, G. Yu, E. Zhou, X. Zhang, Y. Wei, and J. Sun. Res-steps-net for multi-person pose estimation. *Joint COCO and Mapillary Workshop at ICCV 2019: COCO Keypoint Challenge Track*, 2019.

[7] Z. Cai, W. Yin, A. Zeng, C. Wei, Q. Sun, W. Yanjun, H. E. Pang, H. Mei, M. Zhang, L. Zhang, et al. Smpler-x: Scaling up expressive human pose and shape estimation. *Advances in Neural Information Processing Systems*, 36, 2024.

[8] J. Cha, M. Saqlain, G. Kim, M. Shin, and S. Baek. Multi-person 3d pose and shape estimation via inverse kinematics and refinement. In *European Conference on Computer Vision*, pages 660–677. Springer, 2022.

[9] C.-F. R. Chen, Q. Fan, and R. Panda. Crossvit: Cross-attention multi-scale vision transformer for image classification. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 357–366, 2021.

[10] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*, 2017.

[11] X. Chen, A. Pang, W. Yang, Y. Ma, L. Xu, and J. Yu. Sportscap: Monocular 3d human motion capture and fine-grained understanding in challenging sports videos. *International Journal of Computer Vision*, 129:2846–2864, 2021.

[12] Y. Chen, Z. Wang, Y. Peng, Z. Zhang, G. Yu, and J. Sun. Cascaded pyramid network for multi-person pose estimation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7103–7112, 2018.

[13] B. Cheng, B. Xiao, J. Wang, H. Shi, T. S. Huang, and L. Zhang. Higherhrnet: Scale-aware representation learning for bottom-up human pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5386–5395, 2020.

[14] H. Choi, G. Moon, J. Park, and K. M. Lee. Learning to estimate robust 3d human mesh from in-the-wild crowded scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1475–1484, 2022.

[15] V. Choutas, L. Müller, C.-H. P. Huang, S. Tang, D. Tzionas, and M. J. Black. Accurate 3d body shape regression using metric and semantic attributes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2718–2728, 2022.

[16] V. Choutas, G. Pavlakos, T. Bolkart, D. Tzionas, and M. J. Black. Monocular expressive body regression through body-driven attention. In *European Conference on Computer Vision*, pages 20–40, 2020.

[17] O. Chum, J. Matas, and J. Kittler. Locally optimized ransac. In *Joint Pattern Recognition Symposium*, pages 236–243. Springer, 2003.

[18] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[19] L. Doyon, T. Faure, M. Sanz, J. Daura, L. Cassard, and F. d'Errico. A 39,600-year-old leather punch board from canyars, gavà, spain. *Science Advances*, 9(15):eadg0834, 2023.

[20] M. Einfalt, C. Dampeyrou, D. Zecha, and R. Lienhart. Frame-level event detection in athletics videos with pose-based convolutional sequence networks. In *Proceedings Proceedings of the 2nd International Workshop on Multimedia Content Analysis in Sports*, pages 42–50, 2019.

[21] M. Einfalt, K. Ludwig, and R. Lienhart. Uplift and upsample: Efficient 3d human pose estimation with uplifting transformers. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 2903–2913, 2023.

[22] M. Einfalt, D. Zecha, and R. Lienhart. Activity-conditioned continuous human pose estimation for performance analysis of athletes using the example of swimming. In *IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 446–455. IEEE, 2018.

[23] M. Fastovets, J.-Y. Guillemaut, and A. Hilton. Athlete pose estimation from monocular tv sports footage. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 1048–1054, 2013.

[24] Y. Feng, V. Choutas, T. Bolkart, D. Tzionas, and M. J. Black. Collaborative regression of expressive bodies using moderation. In *2021 International Conference on 3D Vision (3DV)*, pages 792–804. IEEE, 2021.

[25] Y. Feng, J. Lin, S. K. Dwivedi, Y. Sun, P. Patel, and M. J. Black. Chatpose: Chatting about 3d human pose. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2093–2103, 2024.

[26] M. Fieraru, M. Zanfir, S.-C. Pirlea, V. Olaru, and C. Sminchisescu. Aifit: Automatic 3d human-interpretable feedback models for fitness training. In *The IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2021.

[27] M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[28] Z. Geng, C. Wang, Y. Wei, Z. Liu, H. Li, and H. Hu. Human pose as compositional tokens. In *CVPR*, 2023.

[29] S. Goel, G. Pavlakos, J. Rajasegaran, A. Kanazawa, and J. Malik. Humans in 4d: Reconstructing and tracking humans with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 14783–14794, 2023.

[30] K. Gong, B. Li, J. Zhang, T. Wang, J. Huang, M. B. Mi, J. Feng, and X. Wang. Posetriplet: Co-evolving 3d human pose estimation, imitation, and hallucination under self-supervision. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11017–11027, 2022.

[31] C. C. Gordon, C. L. Blackwell, B. Bradtmiller, J. L. Parham, P. Barrientos, S. P. Paquette, B. D. Corner, J. M. Carson, J. C. Venezia, B. M. Rockwell, et al. 2012 anthropometric survey of us army personnel: Methods and summary statistics. *Army Natick Soldier Research Development and Engineering Center MA, Tech. Rep*, 2014.

[32] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[33] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[34] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[35] J. Huang, Z. Shan, Y. Cai, F. Guo, Y. Ye, X. Chen, Z. Zhu, G. Huang, J. Lu, and D. Du. Joint coco and lvis workshop at eccv 2020: Coco keypoint challenge track technical report: Udp+. *European conference on computer vision workshops*, 2020.

[36] L. Huang, Y. Li, H. Tian, Y. Yang, X. Li, W. Deng, and J. Ye. Semi-supervised 2d human pose estimation driven by position inconsistency pseudo label correction module. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 693–703, 2023.

[37] C. K. Ingwersen, C. M. Mikkelstrup, J. N. Jensen, M. R. Hannemose, and A. B. Dahl. Sportspose-a dynamic 3d sports pose dataset. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5219–5228, 2023.

[38] C. Ionescu, D. Papava, V. Olaru, and C. Sminchisescu. Human3.6m: Large scale datasets and predictive methods for 3d human sensing in natural environments. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(7):1325–1339, jul 2014.

[39] A. Kirillov, R. Girshick, K. He, and P. Dollár. Panoptic feature pyramid networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6399–6408, 2019.

[40] D.-H. Lee et al. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *Workshop on challenges in representation learning, ICML*, volume 3, 2013.

[41] J. Li, S. Bian, Q. Liu, J. Tang, F. Wang, and C. Lu. Niki: Neural inverse kinematics with invertible neural networks for 3d human pose and shape estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12933–12942, 2023.

[42] J. Li, S. Bian, C. Xu, Z. Chen, L. Yang, and C. Lu. Hybrik-x: Hybrid analytical-neural inverse kinematics for whole-body mesh recovery. *arXiv preprint arXiv:2304.05690*, 2023.

[43] J. Li, S. Bian, A. Zeng, C. Wang, B. Pang, W. Liu, and C. Lu. Human pose regression with residual log-likelihood estimation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 11025–11034, 2021.

[44] J. Li, C. Xu, Z. Chen, S. Bian, L. Yang, and C. Lu. Hybrik: A hybrid analytical-neural inverse kinematics solution for 3d human pose and shape estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3383–3393, 2021.

[45] W. Li, Z. Wang, B. Yin, Q. Peng, Y. Du, T. Xiao, G. Yu, H. Lu, Y. Wei, and J. Sun. Rethinking on multi-stage networks for human pose estimation. *arXiv preprint arXiv:1901.00148*, 2019.

[46] Y. Li, R. Hou, H. Chang, S. Shan, and X. Chen. Unipose: A unified multimodal framework for human pose comprehension, generation and editing. *arXiv preprint arXiv:2411.16781*, 2024.

[47] Y. Li, S. Yang, S. Zhang, Z. Wang, W. Yang, S.-T. Xia, and E. Zhou. Is 2d heatmap representation even necessary for human pose estimation? *CoRR*, 2021.

[48] Y. Li, S. Zhang, Z. Wang, S. Yang, W. Yang, S.-T. Xia, and E. Zhou. Tokenpose: Learning keypoint tokens for human pose estimation. In *Proceedings of the IEEE/CVF International conference on computer vision*, pages 11313–11322, 2021.

[49] J. Lin, A. Zeng, H. Wang, L. Zhang, and Y. Li. One-stage 3d whole-body mesh recovery with component aware transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21159–21168, 2023.

[50] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[51] M. Loper, N. Mahmood, J. Romero, G. Pons-Moll, and M. J. Black. Smpl: a skinned multi-person linear model. *ACM Transactions on Graphics (TOG)*, 34(6):1–16, 2015.

[52] K. Ludwig, M. Einfalt, and R. Lienhart. Robust estimation of flight parameters for ski jumpers. In *2020 IEEE International Conference on Multimedia & Expo Workshops (ICMEW)*, pages 1–6. IEEE, 2020.

[53] K. Ludwig, P. Harzig, and R. Lienhart. Detecting arbitrary intermediate keypoints for human pose estimation with vision transformers. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 663–671, 2022.

[54] K. Ludwig, D. Kienzle, and R. Lienhart. Recognition of freely selected keypoints on human limbs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3531–3539, 2022.

[55] K. Ludwig, D. Kienzle, J. Lorenz, and R. Lienhart. Detecting arbitrary keypoints on limbs and skis with sparse partly correct segmentation masks. In *Proceedings*

of the IEEE/CVF Winter Conference on Applications of Computer Vision, pages 461–470, 2023.

[56] K. Ludwig, J. Lorenz, D. Kienzle, T. Bui, and R. Lienhart. Leveraging anthropometric measurements to improve human mesh estimation and ensure consistent body shapes. In *under review*, 2025.

[57] K. Ludwig, J. Lorenz, R. Schön, and R. Lienhart. All keypoints you need: Detecting arbitrary keypoints on the body of triple, high, and long jump athletes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5179–5187, 2023.

[58] K. Ludwig, S. Scherer, M. Einfalt, and R. Lienhart. Self-supervised learning for human pose estimation in sports. In *2021 IEEE International Conference on Multimedia & Expo Workshops (ICMEW)*, pages 1–6. IEEE, 2021.

[59] H. Ma, Z. Wang, Y. Chen, D. Kong, L. Chen, X. Liu, X. Yan, H. Tang, and X. Xie. Ppt: token-pruned pose transformer for monocular and multi-view human pose estimation. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part V*, pages 424–442. Springer, 2022.

[60] X. Ma, Y. Li, L. Zhao, C. Zhou, and Y. Xu. Scale-pose: Skeletal correction and language knowledge-assisted for 3d human pose estimation. In *Chinese Conference on Pattern Recognition and Computer Vision (PRCV)*, pages 578–592. Springer, 2024.

[61] X. Ma, J. Su, C. Wang, W. Zhu, and Y. Wang. 3d human mesh estimation from virtual markers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 534–543, 2023.

[62] T. Maeda and N. Ukita. Motionaug: Augmentation with physical correction for human motion prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6427–6436, 2022.

[63] N. Mahmood, N. Ghorbani, N. F. Troje, G. Pons-Moll, and M. J. Black. Amass: Archive of motion capture as surface shapes. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 5442–5451, 2019.

[64] D. Mehta, H. Rhodin, D. Casas, P. Fua, O. Sotnychenko, W. Xu, and C. Theobalt. Monocular 3d human pose estimation in the wild using improved cnn supervision. In *2017 international conference on 3D vision (3DV)*, pages 506–516. IEEE, 2017.

[65] G. Moon, H. Choi, and K. M. Lee. Accurate 3d hand pose estimation for whole-body 3d human mesh estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2308–2317, 2022.

[66] G. Moon, H. Choi, and K. M. Lee. Neuralannot: Neural annotator for 3d human mesh training sets. In *Computer Vision and Pattern Recognition Workshop (CVPRW)*, 2022.

[67] A. Nibali, J. Millward, Z. He, and S. Morgan. Aspset: An outdoor sports pose video dataset with 3d keypoint annotations. *Image and Vision Computing*, 111:104196, 2021.

[68] P. Patel, C.-H. P. Huang, J. Tesch, D. T. Hoffmann, S. Tripathi, and M. J. Black. Agora: Avatars in geography optimized for regression analysis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13468–13478, 2021.

[69] G. Pavlakos, V. Choutas, N. Ghorbani, T. Bolkart, A. A. Osman, D. Tzionas, and M. J. Black. Expressive body capture: 3d hands, face, and body from a single image. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10975–10985, 2019.

[70] K. Pearson. Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):559–572, 1901.

[71] F. Picetti, S. Deshpande, J. Leban, S. Shahtalebi, J. Patel, P. Jing, C. Wang, C. Metze III, C. Sun, C. Laidlaw, et al. Anthronet: Conditional generation of humans via anthropometrics. *arXiv preprint arXiv:2309.03812*, 2023.

[72] Z. Qiu, Q. Yang, J. Wang, H. Feng, J. Han, E. Ding, C. Xu, D. Fu, and J. Wang. Psvt: End-to-end multi-person 3d pose and shape estimation with progressive video transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21254–21263, 2023.

[73] Z. Qiu, Q. Yang, J. Wang, and D. Fu. Dynamic graph reasoning for multi-person 3d pose estimation. In *Proceedings of the 30th ACM International Conference on Multimedia*, pages 3521–3529, 2022.

[74] I. K. Rıza Alp Güler, Natalia Neverova. Densepose: Dense human pose estimation in the wild. *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[75] A. Russo. Domain analysis of end-to-end recovery of human shape and pose. https://github.com/russoale/hmr2.0, 2020.

[76] I. Sárándi, A. Hermans, and B. Leibe. Learning 3d human pose estimation from dozens of datasets using a geometry-aware autoencoder to bridge between skeleton formats. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 2956–2966, 2023.

[77] R. Sarkar, A. Dave, G. Medioni, and B. Biggs. Shape of you: Precise 3d shape estimations for diverse body types. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3520–3524, 2023.

[78] A. Sengupta, I. Budvytis, and R. Cipolla. Probabilistic estimation of 3d human shape and pose with a semantic local parametric model. *arXiv preprint arXiv:2111.15404*, 2021.

[79] K. Shetty, A. Birkhold, S. Jaganathan, N. Strobel, M. Kowarschik, A. Maier, and B. Egger. Pliks: A pseudo-linear inverse kinematic solver for 3d human body estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 574–584, 2023.

[80] D. Shi, X. Wei, L. Li, Y. Ren, and W. Tan. End-to-end multi-person pose estimation with transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11069–11078, 2022.

[81] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[82] B. T. Soroush Mehraban, Vida Adeli. Motionagformer: Enhancing 3d human pose estimation with a transformer-gcnformer network. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2024.

[83] V. Srivastav, A. Gangi, and N. Padoy. Self-supervision on unlabelled or data for multi-person 2d/3d human pose estimation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 761–771. Springer, 2020.

[84] Z. Su, M. Ye, G. Zhang, L. Dai, and J. Sheng. Cascade feature aggregation for human pose estimation. *arXiv preprint arXiv:1902.07837*, 2019.

[85] Y. Sun, Q. Bao, W. Liu, Y. Fu, M. J. Black, and T. Mei. Monocular, one-stage, regression of multiple 3d people. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 11179–11188, 2021.

[86] A. Tarvainen and H. Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[87] A. Tarvainen and H. Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[88] J. E. van Engelen and H. H. Hoos. A survey on semi-supervised learning. *Machine Learning*, 109(2):373–440, 2020.

[89] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[90] D. Wang, S. Xuan, and S. Zhang. Locllm: Exploiting generalizable human keypoint localization via large language model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 614–623, June 2024.

[91] J. Wang, K. Qiu, H. Peng, J. Fu, and J. Zhu. Ai coach: Deep human pose estimation and analysis for personalized athletic training assistance. In *Proceedings of the 27th ACM International Conference on Multimedia*, pages 374–382, 2019.

[92] J. Wang, K. Sun, T. Cheng, B. Jiang, C. Deng, Y. Zhao, D. Liu, Y. Mu, M. Tan, X. Wang, et al. Deep high-resolution representation learning for visual recognition. *IEEE transactions on pattern analysis and machine intelligence*, 2020.

[93] S.-E. Wei, V. Ramakrishna, T. Kanade, and Y. Sheikh. Convolutional pose machines. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 4724–4732, 2016.

[94] X. Wei, L. Sha, P. Lucey, P. Carr, S. Sridharan, and I. Matthews. Predicting ball ownership in basketball from a monocular view using only player trajectories. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 63–70, 2015.

[95] Y. Wu, A. Kirillov, F. Massa, W.-Y. Lo, and R. Girshick. Detectron2. `https://github.com/facebookresearch/detectron2`, 2019.

[96] Q. Xie, M.-T. Luong, E. Hovy, and Q. V. Le. Self-training with noisy student improves imagenet classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10687–10698, 2020.

[97] R. Xie, C. Wang, W. Zeng, and Y. Wang. Humble teacher and eager student: Dual network learning for semi-supervised 2d human pose estimation. *arXiv preprint arXiv:2011.12498*, 2020.

[98] Y. Xu, J. Zhang, Q. Zhang, and D. Tao. Vitpose: Simple vision transformer baselines for human pose estimation. *Advances in Neural Information Processing Systems*, 35:38571–38584, 2022.

[99] Y. Yuan, R. Fu, L. Huang, W. Lin, C. Zhang, X. Chen, and J. Wang. Hrformer: high-resolution transformer for dense prediction. In *Proceedings of the 35th International Conference on Neural Information Processing Systems*, pages 7281–7293, 2021.

[100] D. Zecha, C. Eggert, M. Einfalt, S. Brehm, and R. Lienhart. A convolutional sequence to sequence model for multimodal dynamics prediction in ski jumps. In *Proceedings of the 1st International Workshop on Multimedia Content Analysis in Sports*, pages 11–19, 2018.

[101] W. Zeng, S. Jin, W. Liu, C. Qian, P. Luo, W. Ouyang, and X. Wang. Not all tokens are equal: Human-centric visual analysis via token clustering transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11101–11111, 2022.

[102] F. Zhang, X. Zhu, H. Dai, M. Ye, and C. Zhu. Distribution-aware coordinate representation for human pose estimation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 7093–7102, 2020.

[103] H. Zhang, Y. Tian, X. Zhou, W. Ouyang, Y. Liu, L. Wang, and Z. Sun. Pymaf: 3d human pose and shape regression with pyramidal mesh alignment feedback loop. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 11446–11456, 2021.

[104] N. Zioulis and J. F. O'Brien. Kbody: Towards general, robust, and aligned monocular whole-body estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6215–6225, 2023.