Master Thesis

Learning in-the-wild Temporal 3D Pose Estimation from MoCap Data

Author(s):
Cetin, Doruk

Publication Date:
2020

Permanent Link:
https://doi.org/10.3929/ethz-b-000449664

Rights / License:
In Copyright - Non-Commercial Use Permitted
Learning in-the-wild
Temporal 3D Pose Estimation
from MoCap Data

Doruk Çetin

Master Thesis
June 2020

Supervisors:
Emre Aksan
Xu Chen
Prof. Dr. Otmar Hilliges
Abstract

Recovering 3D poses from 2D representations has been a challenging task due to its inherent ambiguity and complexities of the human motion. Lack of annotated data makes it even harder to train neural networks for 3D pose estimation task. In this work, we propose a temporal two-stage architecture to estimate sequences of 3D poses from 2D joint detections, for which we use only 3D motion capture data without paired images for training. More specifically, we generate paired examples by projecting augmented 3D poses to 2D, on-the-fly. We modify our inputs during training through noise and masking to obtain models robust to 2D detection errors. Our approaches utilize the relative simplicity of augmenting 2D and 3D poses rather than images that lie in higher dimensions. Resulting framework employs a simple model, trained on poses without paired images, that can achieve competitive performance on common evaluation scenarios. We present our work as a baseline for the task at hand, discussing further directions for building upon the proposed methodology.
# Table of Contents

List of Figures ........................................ II
List of Tables .......................................... III

1 Introduction ........................................... 1

2 Background ........................................... 3
   2.1 Task definition .................................. 3
   2.2 Representing the human body .................. 4
   2.3 Evaluation metrics ............................... 10
   2.4 Datasets for human pose estimation .......... 11

3 Related work ......................................... 13
   3.1 2D-to-3D pose lifting ......................... 14
   3.2 Data augmentation .............................. 14
   3.3 Temporal human pose estimation ............. 15

4 Methodology .......................................... 17
   4.1 Model input and output ....................... 17
   4.2 Spatio-temporal masking ...................... 20
   4.3 Augmentation on 3D data ...................... 21
   4.4 Regularization by noisy inputs ............... 22

5 Experiments .......................................... 24
   5.1 Experimental setup ............................. 24
   5.2 Results .......................................... 26
   5.3 Comparison with state-of-the-art .......... 30
   5.4 Unsuccessful methods ......................... 31

6 Conclusion ........................................... 34
   6.1 Future work ..................................... 34

A Appendix ............................................. 36
   A.1 Comparison of skeletons ....................... 36
   A.2 Sample estimations ............................. 37
   A.3 Hyperparameters ............................... 40

Bibliography ............................................ 41
List of Figures

1  Comparison of body models  ........................................  5
2  SMPL model  ......................................................  6
3  Rotation representations  ...........................................  8
4  Sample frames from evaluation datasets  ..........................  12
5  Model architecture  ..................................................  17
6  Normalization on 2D inputs  .......................................  18
7  Forward kinematics pipeline  .......................................  19
8  Joint masking on 2D inputs  .......................................  20
9  Data augmentation by rotation  ...................................  21
10 Data augmentation by mirroring  ..................................  22
11 Random noise on 2D inputs  .....................................  22
12 Hybrid architecture  ...............................................  32
13 Failure mode for temporal prediction  ............................  37
14 Prediction of self-occluded joints  .................................  38
15 Successful temporal prediction  ....................................  39
List of Tables

1. Evaluation of rotation formats on 3DPW ........................................ 26
2. Evaluation of temporal modeling on 3DPW ....................................... 27
3. Evaluation of rotation augmentation on Human3.6M and AMASS .......... 27
4. Evaluation of data augmentation methods on 3DPW ............................. 28
5. Evaluation of random noise on Human3.6M and AMASS ........................ 29
6. Evaluation of random noise and joint masking on 3DPW ....................... 29
7. State-of-the-art on Human3.6M ......................................................... 30
8. State-of-the-art on 3DPW ............................................................... 31
9. Comparison of skeletons ............................................................... 36
10. Model settings and hyperparameters ................................................. 40
1 Introduction

Human biological motion is deemed important for communication and research has shown that changes in pose carry a significant amount of information on human motion (e.g. Johansson [31]’s point light display experiments). Therefore, a thorough understanding of human pose serves a similarly crucial role in understanding human behaviour for the machines that perceive and interact with moving people. In this context, human pose estimation has many practical applications in areas such as human-computer interaction, activity recognition and augmented/virtual reality.

Human pose estimation has seen a growing interest over the years, but the task is still an open research problem due to its various natural challenges including variations in lighting and background conditions, occlusions and physical constraints. Moreover, 3D pose estimation is even a harder task than traditional 2D pose estimation as it is inherently ambiguous due to loss of information in perspective projection.

Recently, deep learning based methods achieved significant improvements in 3D pose estimation benchmarks, after becoming viable with the advent of large scale motion-capture datasets. However, in-the-wild settings are still problematic for 3D pose estimation since available datasets are limited in size and collected mostly on restricted environments, such as indoors with motion capture (mocap) suits. Thus, an effective solution to the 3D pose estimation task should utilize the available datasets extensively and also do this in a way that explicitly addresses the challenges of the in-the-wild settings. We believe many methods do not fully utilize the rich amount of information that exists in 3D datasets and there is a potential in the exploration of techniques that better utilize such intrinsic, structural information.

The most common framework for 3D pose estimation is to directly estimate 3D poses from 2D images. Existing annotated 3D datasets are insufficient to train models for this task, so one approach to alleviate the challenges of 3D setting is to exploit the well-studied problem of 2D pose estimation, where many of the aforementioned challenges also exist. It is possible to train fairly robust models for 2D pose estimation through data-driven approaches, such as convolutional neural networks (CNNs), as collection of diverse annotated datasets is comparably much easier for the 2D setting. Therefore, many methods for 3D pose estimation employ 2D datasets in their approaches for additional supervision. Such methods either predict poses both in 2D and 3D [80, 50, 10] or they obtain 2D poses by reprojecting their 3D pose estimations onto images [87, 34, 84]. As successful they are, these approaches typically utilize the 3D datasets only through their paired images, which limits the potential gains.

Similarly, many competitive approaches do not utilize the temporal information at all and operate on individual frames [56, 39, 24], even though models that use the the temporal information in an end-to-end manner [60, 35, 38] or as a post-processing step [6, 15] report that incorporating temporal knowledge improves 3D pose estimations by generating smoother and more realistic body movements.

To address these issues, we propose a methodology to train models on mocap datasets
augmented to reflect the challenges of in-the-wild settings. We adopt the so-called *two-stage approach* for 3D pose estimation, where the problem is split into two consecutive parts: 2D pose estimation from images and 3D pose estimation from 2D joint detections. This not only allows the usage of 2D pose estimation models, but it also brings the much simpler problem of 2D-to-3D pose estimation into the focus. The main advantage of this formulation is the simplicity of data augmentation, relative to the common setting with 2D images. Generating realistic synthetic datasets is hard and they currently do not reach the variability of in-the-wild images [88]. This is not the case with our setting as 2D poses exhibit less variation than that of images and they can be generated simply through perspective projection when 3D poses are available.

More specifically, our approach is as follows. We train our models using AMASS, a recently introduced large-scale mocap dataset [48]. We first augment the 3D pose sequences by mirroring the body models and globally rotating them. We then obtain paired 2D pose instances through projection and also augment them. To simulate the inaccuracies of the 2D pose detections we utilize, we inject noise on projected 2D joints and spatiotemporally mask them. We then train temporal models for 3D pose estimation on the 2D-3D pose sequence pairs we generate.

We tackle a restricted setting where we are provided only with 2D pose detections of another model at test time. Our aim is to train robust, domain-agnostic models —especially for the more challenging in-the-wild scenario. We only use mocap data to train our models and obtain competitive performance without utilizing any image data during training. We illustrate how basic but useful approaches can be brought together to build up a pose estimation system that addresses deficits of some current models. We perform several experiments to investigate the inner workings of our model both qualitatively and quantitatively. We also explore shortcomings of our model and identify prospective directions for research.

In summary, our main contributions include:

- We build a temporal model for 3D pose estimation that achieves competitive results by leveraging AMASS through extensive data augmentation.
- We conduct an experimental analysis on the methodologies we employ and propose potential solutions to address a number of problems we identify.
- We evaluate our approaches on two widely-used benchmarks for the task at hand and present our results as a simple yet effective baseline.

The thesis is structured as follows. In Chapter 2, we delineate the task at hand and provide the necessary background. In Chapter 3, we summarize the previous work related to ours, discussing motivations behind each line of work. In Chapter 4, we detail our methodology, individually examining the different approaches we employ. In Chapter 5, we present the setup and results of our experimental analysis. In Chapter 6, we summarize the findings of this project and propose future work. In Appendix, we provide supplementary material.


2 Background

Here, we go over the basics of human pose estimation, mostly focusing on elements related to our problem definition as well as our experimental methodology. Section 2.1 places the particular task we work on in the broader context of human pose estimation. Section 2.2 introduces parametric body models and discuss different modeling choices. Section 2.3 details common evaluation metrics for the task of human pose estimation. Section 2.4 outlines the evolution and current state of the 3D pose estimation datasets.

2.1 Task definition

Human pose estimation is the task of predicting the human body configurations in images—or sequences of images (i.e. videos). In practice, it is most generally defined through the localization of human joints (also called keypoints), although it has been commonly coupled with the human shape estimation task as well.

2.1.1 Taxonomy of approaches

Input for pose estimation models is most often RGB images, but there are also other classes of methods that operate on different data formats, such as RGB-D images which include depth information [74, 64].

Majority of the work in literature operate in per-frame basis [8, 56, 34, 24], i.e. they estimate poses in sequences of images separately. Conversely, some methods work exclusively on videos and not singular images, in order to learn and exploit the temporal dynamics of human motion in image sequences [35, 38, 60]. We refer to these two classes respectively as non-temporal and temporal methods.

One major division in the field concerns the differences in 2D and 3D pose estimation tasks—former involves estimating the \((x, y)\) coordinates of joints in images, whereas the output of the latter is joint locations in 3D space. However, these two are not disjoint tasks as models and datasets for 2D pose estimation have been utilized for the more difficult task of 3D pose estimation. There are 3D pose estimation models that jointly predict keypoints both in 2D and 3D, to utilize annotated 2D datasets as well [80, 79]. To that end, some works reproject their 3D pose estimations onto images to obtain 2D joint locations [87, 34, 84]. Similarly, there is a family of approaches that first estimate 2D poses and then lift them to 3D by also estimating depth information [50, 53, 10].

3D human pose estimation is further divided into two common settings, namely single-view (monocular) and multi-view scenarios. Monocular 3D pose estimation setting exhibits depth ambiguity and it is therefore inherently more challenging than the latter scenario. Multi-view setting assumes the availability of multiple synchronized and calibrated cameras, which can be useful when resolving depth ambiguities [33, 29].

In this thesis, we work on the particular task of monocular 3D human pose estimation and we comparatively evaluate our approaches in this setting. We train and test our models in both non-temporal and temporal scenarios. In the rest of this thesis, we
interchangeably use the term “pose estimation” to refer to the aforementioned problem setting, unless explicitly stated otherwise.

### 2.1.2 Related tasks and applications

Human pose estimation is closely related to many other tasks, primarily in the fields of computer vision, graphics and animation. For examples, human activity detection and recognition tasks aim to identify different actions in videos and naturally involves interpretation of human poses and their changes over time. Similarly, motion modeling requires estimation of valid and temporally consistent poses. Numerous other tasks such as scene understanding, motion retargeting or human tracking share challenges common in pose estimation literature and they consequently benefit from approaches that address such challenges in a reciprocal manner. Therefore, human pose estimation is a key component of many application areas not only in itself, but also with regards to its close relation to many other problems.

Applications of human pose estimation can be found in both scientific and consumer domains [72]. For example, Human-Computer Interaction (HCI) can utilize human pose estimation to build natural interfaces that recognize human actions and gestures. Similarly, Human-Robot Interaction (HRI) requires a thorough perception of moving people. Other applications of human pose estimation range from gaming and virtual reality (VR) to video surveillance and sport performance analysis, to name a few more examples.

### 2.2 Representing the human body

The human body is a very complex system that can exhibit many types of deformation through its highly articulated and nonrigid nature. Furthermore, body shapes can also vary greatly across different individuals. Utilizing prior knowledge can be helpful when dealing with such systems with high degrees of freedom. Model-based approaches offer a convenient way to introduce a pose estimation system with prior knowledge about the articulation and shape of body parts.

As larger datasets that provide motion capture and 3D body scans became available [5, 76, 28], parameterized body models have become more widely used in modern approaches to human pose estimation. This is not only the case with human pose estimation as body models are commonly used in aforementioned related tasks as well.

#### 2.2.1 Parametric body models

**Skeletons**, also called stick figures or kinematic models, are the most common body models in the literature. They define the structure and kinematic properties of the human body through a predetermined set of joints. Usage of skeletons makes it easy to quantify errors in human pose estimation. These errors can be useful when comparatively evaluating different pose estimation models, but they can be also used as loss functions while training such models.
Figure 1: Comparison of the 3D body models on one sample frame from AMASS [48] dataset. H36M and COCO skeletons share the same body joints. We do not utilize the illustrated joint hierarchy in our experiments and it is used only for visualization purposes. We refer to OpenPose [9] for the joint hierarchy of H36M and COCO skeletons.

Skeletons also allow the parameterization of the body poses through joint angles, which better encapsulates the motion dynamics. Joint positions can still be directly calculated via forward kinematics, given either absolute or relative joint angles. This calculation of the joint positions too can be done while training a neural network model as forward kinematics is differentiable with respect to joint rotations. This allows a model that regresses joint rotations to be trained using position-based loss functions [61, 85, 21]. This way, a model can predict joint rotations on a fixed 3D skeleton, avoiding errors due to estimating non-constant bone-lengths or motions outside an articulation range [34, 60].

In Figure 1 we comparatively illustrate the three different skeleton models we employ in this study, alongside the corresponding body mesh for the respective sample frame.

**H36M skeleton** we use, illustrated in Figure 1a, is a derivative of the skeleton format introduced by Ionescu et al. [28]. Human3.6M dataset originally annotated 3D body poses with a skeleton composed of 32 joints. Only 17 of those joints move and the remaining joints can be directly calculated from those moving joints. As it is common in literature to estimate this skeleton of 17 joints [50, 60], some approaches further take a subset of 14 joints to train and evaluate their models [35]. This subset of joints actually corresponds to skeleton format presented in the Leeds Sports Pose (LSP) Dataset [32]. In the rest of this thesis we will also be referring to this skeleton with 14 joints when we discuss H36M skeleton format.
COCO skeleton in Figure 1b similarly takes its name from the COCO Keypoints Challenge dataset [45] where it is first introduced. COCO and H36M skeletons share the same set of 13 joints to denote the body pose. However, COCO skeleton annotates the head pose using 5 joints that corresponds to nose, ears and eyes —in contrast to H36M skeleton using just one keypoint to denote the top of the head. Reader can refer to Table 9 in Appendix A.1 to compare COCO skeleton format with two aforementioned variations of H36M skeleton.

Skeletons can also be useful in the sense that body shape models can be constructed on top of them. Although having a separate structure model is not a necessary part of modeling the body shape (e.g. SCAPE [5]), having a skeleton underneath allows the model to decouple body shape and pose through separate parameters. Last body model we discuss is an example of this.

Figure 2: SMPL model: (a) Following the standard practice, skinning starts with model template at zero pose —also called T-pose. Blend weights are indicated by color and joints are denoted as white dots. (b) Blend shapes corresponding to subject identity are added, vertex and joint locations are transformed linearly with respect to shape parameters $\beta$. (c) Another set of blendshapes (calculated using pose parameters $\theta$) are applied to account for pose-dependent deformations. (d) Lastly, a standard blend skinning function —-dual-quaternion function for this model —is applied to repose deformed vertices for the split pose.

Skinned Multi-Person Linear model (SMPL) [47] is a skinned vertex-based human body model. As discussed above, SMPL models the body pose and shape through separate parameters, namely $\theta \in \mathbb{R}^{72}$ and $\beta \in \mathbb{R}^{10}$. Body pose is defined using a skeleton with 23 joints, illustrated in Figure 1c, by specifying relative rotations in axis-angle format. Global body rotation is also specified by a rotation vector, which is commonly viewed as the rotation of the hip joint (red in the respective figure), resulting in a total of 24 joints. Body shape is parameterized by the first 10 coefficients of the PCA shape space the authors have constructed. SMPL is then the differentiable function, $M(\theta, \beta) \in \mathbb{R}^{6890 \times 3}$, that outputs a triangulated mesh shaped based on the forward kinematics using parameters $\theta$ and $\beta$. Figure 2 illustrates and briefly explains this process. Reader should refer to the original paper by Loper et al. [47] for a more in-depth discussion.
Similarly to discussion above about utilizing skeletons, explicitly parameterizing body shape and poses through SMPL has been reported to be useful for 3D human pose estimation by several studies [8, 58, 39, 34]. Main advantage of using such a body model is that the neural networks can be trained to produce very detailed mesh results while being trained only to estimate a small number of parameters. Referenced studies also demonstrate how these parameters can be reliably predicted from images or 2D keypoints and how the neural networks can be further supervised using estimated 3D meshes.

Authors have originally provided two SMPL models, one each for male and female body shapes, trained appropriately to learn different shape spaces. Bogo et al. [8] further expanded their methodology by providing a gender-neutral model trained on both male and female subjects. Such a model is helpful for estimation tasks where the gender of the subject is unclear or completely unknown. We note that all three models produce meshes with same topology.

SMPL has two variants, namely SMPL-H [70] and SMPL-X [57]. SMPL-H replaces two hand joints in the SMPL model with MANO hand pose model, which provides more detailed articulation through 15 joints per hand. SMPL-X further extends the SMPL-H model with the addition of 3 additional joints for face keypoints. SMPL-X model can therefore express articulation in jaw and eyes alongside fingers on both hands using 54 joints in total, excluding the global rotation.

We continue our discussion on body models and their specifics on following sections. More specifically, Section 4.1 describes where we integrate the body models in our estimation pipeline and Section 5.1 details how exactly we use each body model in our experiments.

2.2.2 Rotation representations

In the previous section, we have discussed how parametric body models such as skeletons can be used to represent the body pose through joint rotations. Another modeling choice regarding such body models is how we choose to represent the rotations themselves. Rotations in three dimensions can always be uniquely described by a minimum of three parameters and different rotation formalisms will always have three degrees of freedom, as Euler’s rotation theorem dictates, even when they use more than three parameters. However, these rotation representations have different characteristics and can therefore be expected to behave differently in the context of neural networks.

Euler angles split any given rotation into three successive rotations around the axes of the coordinate system. Unlike other representations such as rotation matrices or quaternions, they are not expressed in terms of external frame. Their definitions vary as rotations are not commutative, i.e. choice of axes and their successive ordering do matter. There exists a total of 12 different possibilities to specify Euler angles and they are divided into two groups as Tait-Bryan ordering ($xyz$, $zxz$, $yxz$, $zxy$, $zyx$, $yzx$) and proper ordering ($xyz$, $xzx$, $yyx$, $zyx$, $zxy$, $zyz$). Tait-Bryan angles following extrinsic $xyz$ ordering are also called $yaw$, $pitch$, $roll$ in the fields of aerospace and aviation, which can be seen in
Figure 3a. One drawback of Euler angles is that they suffer from singularities as all other \(\mathbb{R}^3\) parameterizations. This issue is referred to as *gimbal lock*, which makes these rotation representations unsuitable for some applications: the problem arises when two of the three rotation axes align and causes a rotational degree of freedom to be lost.

**Figure 3: Rotation representations:** (a) Euler angles in extrinsic \(xyz\) ordering refer to heading, elevation and bank (i.e. yaw, pitch and roll) of an aircraft. (b) An arbitrary rotation can be defined using an axis of rotation and a rotation angle (both shown in green) using axis-angles. (c) A function \(g\) maps the original Euclidean space to a representation space, to be used by neural networks. Zhou et al. [97] define a representation as discontinuous if the function that maps the original space to the representation space is a discontinuous function. This is the case with the example shown here, as a set of continuous angles become discontinuous after the mapping is applied, consequently making the mapping itself a discontinuous representation per the definition above.

**Axis-angles** (also called exponential maps) parameterize rotations by two quantities: a unit vector to specify the axis of rotation and an angle to specify the magnitude of rotation around that axis. A sample rotation in axis-angle format is illustrated in Figure 3b. Axis of rotation is the one that remains unchanged by the rotation. Since the axis is normalized, it only has two degrees of freedom and not three —resulting in axis-angles using a total of three parameters to represent rotations. One drawback of axis-angles is that they lack a composition operator as they do not satisfy the law of vector addition. In practice, axis-angles are converted to another representation (e.g. rotation matrices or quaternions, where composition is straightforward), product is calculated and result is converted back to axis-angle formulation.

**Rotation matrices** represent rotations through \(3 \times 3\) matrices, whose columns (or rows depending on the definition) form vector bases. For a specific rotation, the three unit vectors that form the basis specify the position of the rotated coordinate frame in terms of the reference (non-rotated) coordinate axes. Reader should note that elements of the...
matrix are not independent as the matrix has only three degrees of freedom, as discussed above. Although they require relatively more parameters than other representations, they are advantageous in the sense that successive rotations can be combined very easily by matrix multiplication.

**Quaternions** form a four-dimensional vector space that extends the complex numbers. Rotations in 3D are represented through normalized quaternions, which are also called versors, unit-quaternions or more specifically rotation quaternions in the context of 3D rotations. Extension to four dimension avoids gimbal lock and discontinuous jumps that are inherent to three-dimensional parameterizations. Interpolating rotation quaternions can be done easily via spherical linear interpolation (Slerp) [73] and also result in smooth transitions. Rotation composition is simply defined through quaternion multiplication function, as seen in Equation 1. One other advantage of using rotation quaternions is that they are less susceptible to round-off errors as an erroneous quaternion is still a valid rotation after normalisation. This is not the case for rotation matrices as orthogonality could be hard to ensure for such scenarios. Similarly, quaternions involves no trigonometric functions.

\[
\tilde{q} \otimes q = \begin{bmatrix}
q_r & q_k & -q_j & q_i \\
-q_k & q_r & q_i & q_j \\
q_j & -q_i & q_r & q_k \\
-q_i & -q_j & -q_k & q_r \\
\end{bmatrix}
\begin{bmatrix}
\tilde{q}_i \\
\tilde{q}_j \\
\tilde{q}_k \\
\tilde{q}_r \\
\end{bmatrix}
= \begin{bmatrix}
\tilde{q}_r & -\tilde{q}_k & \tilde{q}_j & \tilde{q}_i \\
\tilde{q}_k & q_r & -\tilde{q}_i & \tilde{q}_j \\
-\tilde{q}_j & \tilde{q}_i & q_r & \tilde{q}_k \\
\tilde{q}_i & \tilde{q}_j & -\tilde{q}_k & q_r \\
\end{bmatrix}
\begin{bmatrix}
q_i \\
q_j \\
q_k \\
q_r \\
\end{bmatrix}
\tag{1}
\]

**5D and 6D continuous representations** are rotation formulations presented by Zhou et al. [97] in their comparative study on 3D rotation formulations for deep neural networks. They put forward a definition for representation continuity in neural networks (see Figure 3c) and show that all representations of 3D rotations are discontinuous in four or fewer dimensions, such as Euler angles and quaternions. They also present a method to continuously represent rotations in fewer dimension and obtain continuous representations for 3D rotations in 5D and 6D. Equation 2 shows the mapping from $SO(3)$ (i.e. rotation matrices) to their 6D representation. Similarly, Equation 3 illustrate how these 6D rotations can be converted back to rotation matrices, where $N(v) = v/||v||$ denote vector normalization. Throughout our study, we briefly refer to this 6D continuous representation as “rot6d”.

\[
g_{GS}\left(\begin{bmatrix}
a_1 & a_2 & a_3 \\
\end{bmatrix}\right) = \begin{bmatrix}
a_1 \\
a_2 \\
\end{bmatrix}
\tag{2}
\]

\[
f_{GS}\left(\begin{bmatrix}
a_1 & a_2 \\
\end{bmatrix}\right) = \begin{bmatrix}
b_1 \\
b_2 \\
b_3 \\
\end{bmatrix},
\tag{3}
\]

where $b_i = \begin{cases}
N(a_1) & \text{if } i = 1 \\
N(a_2 - (b_1 \cdot a_2) b_1) & \text{if } i = 2 \\
b_1 \times b_2 & \text{if } i = n
\end{cases}$
A number of works experimented with alternative rotation formalisms and analysed their effects on learning. Pavllo et al. [59] compare Euler angles, axis-angles and quaternions, both empirically and theoretically, on the motion prediction problem. They also present methods to tackle drawbacks of representations they examine. Villegas et al. [89] also regress quaternions and obtain positions via forward kinematics for the motion retargetting task they address. Kolotouros et al. [39] report faster convergence times using rot6d, whose advantages over axis-angles are later corroborated by Kocabas et al. [38].

2.3 Evaluation metrics

Mean Per Joint Position Error (MPJPE) is perhaps the most common evaluation metric for 3D Human Pose Estimation. Joint-wise errors are calculated as the Euclidean distances between corresponding ground-truth and prediction joint positions. MPJPE is then simply the mean of those per joint position errors calculated. It is important to note that MPJPE is most generally calculated after the aligning of the root joint (typically the pelvis) of ground-truth and prediction body poses.

Equation 4 shows the MPJPE calculation for one frame after root joint alignment, where \( N \) is the number of joints, \( J_i \) is the 3D position of the \( i^{th} \) ground-truth joint and \( \hat{J}_i \) is that of the \( i^{th} \) predicted joint.

\[
E_{MPJPE} = \frac{1}{N} \sum_{i=1}^{N} \left\| J_i - \hat{J}_i \right\|_2
\]  

MPJPE is averaged over all frames in a dataset when comparing different models across multiple samples. Equation 5 shows a more complete definition of the MPJPE metric that includes the root alignment and averaging over frames, where \( T \) is the total number of frames in the dataset.

\[
E_{MPJPE} = \frac{1}{T} \frac{1}{N} \sum_{t=1}^{T} \sum_{i=1}^{N} \left\| (J_i^{(t)} - J_{\text{root}}^{(t)}) - (\hat{J}_i^{(t)} - \hat{J}_{\text{root}}^{(t)}) \right\|_2
\]  

The metric has been first introduced by Sigal et al. [76] as 3D Error on HumanEva dataset and later referred by Ionescu et al. [28] as MPJPE. Numerous works [78, 8, 53] have also reported variants of the MPJPE metric that incorporated a frame-wise alignment step.

Procrustes Alignment MPJPE (PA-MPJPE) is one such popular variant where MPJPE is calculated after applying a similarity transform to predicted 3D poses, in order to frame-wise align each pose optimally in least-square sense. This method of shape analysis is known as Procrustes method [22], hence the name of the modified metric. Procrustes Alignment MPJPE has been referred by different names in human pose estimation literature —namely, PA Joint Error [79], P-MPJPE [67], reconstruction error [34, 58] or simply protocol #2 [50] when referring evaluation on Human3.6M. In this thesis, we use the names PA-MPJPE and reconstruction error when referring to the metric.
2.4 Datasets for human pose estimation

Human pose estimation has been studied for years, but it is still an active problem due to its inherent challenges. Such challenges can include natural variability of the human shape and pose, as well as variations in external conditions such as occlusions or changes in lighting. In order to compensate the said variability, human pose estimation datasets accordingly saw an increase in number and complexity. This trend has gained even more speed since the current paradigm is dominated by data-driven deep learning methods [16]. In this subsection, we try to give the reader an overview of the progression of 3D pose estimation datasets. However, we focus on the three datasets utilized in our experimental analysis and follow a more cursory discussion on the rest.

Earlier datasets in the human pose estimation literature were aimed at the 2D pose estimation task and provided images with paired 2D annotations. Some more recent and well-known examples include LSP [32], MSCOCO [45] and MPII Human Pose [4] datasets. First major 3D pose dataset was HumanEva-I&II [76], aimed for establishing a quantitative evaluation setting for 3D human pose estimation. It was later followed by Human3.6M [28], which also took its place in the 3D pose estimation literature as a standard evaluation scenario.

Human3.6M is an indoor dataset collected using a marker-based motion capture system. The dataset contains 11 professional actors (5 female and 6 male) in everyday scenarios such as walking, smoking or talking on the phone. Corresponding videos are recorded from 4 different viewpoints, amounting to a total of 3.6 million frame-wise poses (at a framerate of 50Hz), giving the dataset its name. A sample frame from Human3.6M can be viewed in Figure 4a.

Such datasets as HumanEva and Human3.6M have contributed to advances in the field, but they have been shown to present restricted scenarios. These datasets are captured in constrained environments with limited variability on clothing, background and lighting as well as bounded number of people and actions. Therefore, models trained on these datasets fail to account for the complexity of in-the-wild images —ones that pertain to real world, rather than controlled settings. This called for the need of a novel evaluation dataset that better reflects the real world variability.

3D Poses in the Wild (3DPW) [90] is the current standard benchmark for in-the-wild 3D pose estimation task. The dataset comprises of videos taken outdoors with a moving phone camera and their pose annotations. A sample frame from 3DPW is shown in Figure 4b. 3D poses in the dataset are estimated through joint optimization of SMPL body models on 2D pose detections and recordings from inertial measurement units (IMUs). 3DPW is comparably smaller than Human3.6M with around 51k frames (at a framerate of 30Hz) and it similarly focuses on everyday actions such as walking, drinking or climbing stairs. However, 3DPW presents a more realistic and consequently a more challenging evaluation setting than that of Human3.6M: the camera is moving, subject clothing is more realistic, background is more cluttered and there are occlusions due to environment.
2 BACKGROUND

(a) Sample frame from Human3.6M
(b) Sample frame from 3DPW

Figure 4: Sample frames from evaluation datasets: Two evaluation datasets we use, exemplified side-by-side. (a) Human3.6M is recorded through four cameras, two of which can be viewed in the example image. 3D annotation is done via mocap markers placed around the body of the actor. (b) 3DPW consists of videos taken with a phone camera, almost always outdoors. Some samples include multiple moving actors, as one illustrated here.

However, there is still a lack of labeled real-world datasets large enough to train estimation models. 3D data collected in controlled environments have found to be insufficient for 3D pose estimation in-the-wild. Some recent works underline the differences in Human3.6M and 3DPW test scenarios, illustrating the discrepancy of training on mocap datasets for pose estimation in-the-wild [35, 38]. Researchers tried to overcome this challenge through different methods, such as synthetic dataset generation or weak supervision from 2D detections (see Section 3.2). One such direction is not only utilizing image-annotation pairs, but also exploiting any available data —such as images without pose annotations or pose sequences without matching images.

Archive of Motion Capture as Surface Shapes (AMASS) [48] is a recent meta-dataset that unifies 17 different optical marker-based mocap datasets under a single framework and it is the largest unified mocap dataset at the date. The dataset does not include any 2D data, such as images and it is completely made up of 3D pose sequences. AMASS uses an extension of SMPL [47] called SMPL-H [70] to represent the collection of 3D human meshes. Authors introduce a new method called MoSh++ to obtain the respective SMPL parameters by fitting the surface of the SMPL model to observed mocap markers.
3 Related work

Numerous previous work proposed methods to alleviate the difficulties of the 3D pose estimation task in alternative ways. Some recent examples include usage of parametric body models [8, 39], adversarial learning [34, 38, 13] and incorporation of additional image representations as inputs [18, 56]. In the rest of this section, we focus our attention on literature related to the methodologies we employ. We first briefly present the motivations in each line of work and follow this by detailed discussions on respective approaches. We discuss how specific methods relate to the methodologies employed in our approach in Section 4.

- **2D-to-3D pose lifting:** Collecting data with 2D pose annotations is easier and more applicable to in-the-wild scenarios than 3D pose datasets, which usually require lab environments with advanced equipments. Furthermore, pose estimation in 2D is a well-studied problem that is comparably simpler than its 3D counterpart. It naturally follows that 2D pose estimators have been quite successful in achieving invariancy with respect to factors such as occlusions or background clutter. Therefore, some studies have found it useful to separate 3D pose estimation into two consecutive parts: 2D pose estimation from images and 2D-to-3D pose lifting. This not only divides the problem in two well-defined sub-tasks, but it also provides the opportunity to exploit robust 2D pose estimator models. We discuss related works on 2D-to-3D pose lifting in Section 3.1.

- **Data augmentation:** Neural network architectures, particularly ones utilizing CNNs, currently dominate the state-of-the-art for both 2D and 3D pose estimation tasks. Deep learning methods are infamously known to be data-hungry and the lack of large annotated datasets particularly exacerbates this problem for 3D pose estimation. One way of dealing with insufficient data is data augmentation, i.e. transforming or modifying the available samples to increase the diversity of datasets. Similarly, numerous approaches create synthetic data samples (e.g. through generative models) to increase the multiplicity of training datasets. Similarly, there have been approaches that utilize 3D data without corresponding 2D representations in order to help models learn useful priors. We discuss related works that aim to explicitly address the lack of annotated datasets in Section 3.2.

- **Temporal human pose estimation:** Pose estimation in 3D is an ill-posed problem and human poses can exhibit much variations. However, this variety is typically constrained in ways we can understand, through strong characteristics the human body incorporates. Examples include the symmetry of bone lengths or limits of joint angles, among others. These phenomena can then act as priors for the estimator models, either implicitly or through explicit modeling. One such recent approach is utilizing the motion dynamics in pose sequences to acquire temporally consistent 3D pose estimations, which can directly help with the ambiguity in 3D-to-2D mapping. We discuss related work on temporal pose estimation in Section 3.3.
3 RELATED WORK

3.1 2D-to-3D pose lifting

Lee et al. [42] present the earliest example of the two-stage architecture, which lays out the idea behind many approaches to come. Some later works have assumed access to ground truth 2D joints [1, 20] or their manual real-time annotation [23], although most of the subsequent work focuses on fully automated approaches. While numerous methods choose to rely on 2D pose estimations of external models, there exists a number of works that implement both parts of the two-stage architecture [94, 44, 96]. This additionally allows for the joint training of both stages, where supervision from 2D samples is used to train the first stage (i.e. 2D pose estimator model) directly, alongside paired 3D data supervising either the entire network or just the lifting model.

That being said, a large body of work that estimates 3D poses from 2D assumes that the 2D joints are provided by an off-the-shelf 2D pose detector, allowing them to focus only on the task of 2D-to-3D lifting [77, 65, 78, 92, 8, 50, 53, 96, 10, 55]. This methodology became increasingly viable following the availability of many successful 2D pose estimators [62, 54, 93], but it is important to note that performance of the 2D pose estimator becomes a limiting factor for works that follow this methodology.

Martinez et al. [50] study a simple and fast neural network architecture that directly lifts 2D pose into 3D coordinates, which works well with 2D ground truth as well as 2D joint detections. Moreno-Noguer [53] formulates the lifting problem as a 2D-to-3D distance matrix regression. Zhou et al. [96] integrate a pretrained 2D pose estimator into their model and train it using supervision from both 2D and 3D data. Chen et al. [10] project a library of 3D poses on virtual cameras to obtain a training set of paired 2D and 3D pose instances. Then, they match the 2D detections to their nearest neighbors in the training set and report the 3D pose associated with the closest matching 2D sample. Nie et al. [55] estimate depths of 3D joints using a multi-layered long short-term memory (LSTM) architecture that uses 2D joint detections, as well as local image patches around them.

3.2 Data augmentation

Most straightforward way to generate additional data samples for pose estimation is augmenting images through basic image manipulations. Such manipulations include randomly rotating, scaling, flipping, cropping or translating images and accordingly modifying their corresponding annotations. These simple manipulations have been useful for many computer vision tasks and numerous approaches mentioned throughout this thesis also employ a combination of such manipulations [95, 54, 93, 79, 44, 58, 60, 81]. However, affine transformations are not the only way to modify RGB images and some methods have utilized green screens and segmentation masks to modify both foreground and background textures to augment their training datasets [66, 51].

More elaborate methods rely on exploiting 3D models and their projections to generate synthetic data samples. Pishchulin et al. [63] present an early example of such augmentation where they randomly sample parameters of a 3D body model, project them onto images and embed them into backgrounds by morphing the segmentations and applying linear blend skinning. Rogez et al. [68] create synthetic images by stitching
image patches that locally match the projected joint positions of a 3D pose and blending them. Chen et al. [11] and Varol et al. [88] randomly sample pose, shape and texture for 3D body models and render them on real backgrounds under various viewpoints and lighting conditions. Similarly, there are methods that utilize game engines coupled with procedural generation [17] and adversarial learning [26] to generate synthetic videos.

There also exist more specific approaches for data augmentation that align more closely with our methodology. Similarly to our approach, some two-stage estimator models generate paired data samples by projecting 3D data to 2D via virtual cameras [94, 10, 58]. [71] train an inpainting model for motion modeling using spatiotemporal occlusion masks on 3D joints during training. [12] simulate occlusion conditions on 2D data by randomly dropping joints, frames or groups of joints. [36] copy background patches from the images and paste them over keypoints to mask them by rendering them invisible.

It is also possible to harness datasets where there are not input-target pairs, but only a collection of inputs or targets. For the task of 3D human pose estimation, inputs are most generally assumed to be 2D images and targets are 3D pose annotations, usually in the form of body model parameters. There exists a number of works that utilize datasets made up of only 3D poses without matching images or only 2D images without pose annotations.

A number of works utilize 3D-only mocap data to train to adversarially train a generator-discriminator pair to obtain more realistic pose estimations [86, 34, 35, 38]. On the other hand, Kanazawa et al. [35] annotate unlabeled videos with pseudo-ground truth 2D poses using an off-the-shelf pose estimator to generate paired data instances for weak supervision.

### 3.3 Temporal human pose estimation

Most of the previous work on human pose estimation employed frame-wise approaches, although there is a tendency in recent studies to exploit temporal information in videos. We focus our review on monocular setting (i.e. single camera), but we also note the existence of temporal approaches for pose estimation that operates in multi-view settings.

Numerous approaches utilize temporal information as a post-processing step: they first obtain 3D pose estimations in frame-wise settings and then solve constrained optimization problems to temporally smoothen sequences of poses. Such constraints include priors on rotations and pose coefficients [95], bone lengths [91], 3D joint locations [27] and joint angles [52].

Conversely, some more recent approaches employ either convolutional or recurrent architectures to obtain temporal coherency in 3D pose estimations in an end-to-end manner. Lin et al. [44] sequentially estimate 3D poses from images and iteratively refine them. Coskun et al. [14] train an LSTM model to learn the parameters of a Kalman filter, which they use to regularize the outputs of an off-the-shelf 3D pose estimator. Hossain
et al. [25] use a sequence to sequence model [82] to lift 2D detections to 3D. Tung et al. [87] train their model completely on synthetic data [88] and fine-tune it in test time using self-supervision through 2D keypoints, silhouettes and optical flow. Dabral et al. [15] present a two-stage approach where they frame-wise estimate 3D poses from images (SAP-net) and temporally refine the estimated poses via CNNs over a finite context of past frames.

Most recent works for temporal pose estimation also follow the aforementioned line of work — i.e. they integrate temporal components in the end-to-end training procedure. Kanazawa et al. [35] temporally encode frames and further exploit motion dynamics by making the model predict into nearby past and future timesteps. Pavllo et al. [60] use dilated temporal convolutions to lift 2D pose sequences to 3D while exploiting long-term information. Kocabas et al. [38] utilize sequence-level adversarial learning on pose sequences in AMASS via a temporal generator-discriminator couple — that uses bidirectional gated recurrent units (GRU) and temporal attention, respectively.
4 Methodology

Figure 5: Model architecture: Our model uses 2D joint locations as input and estimates 3D joint angles as output. At test time we use 2D joint predictions of pretrained pose estimator in lieu of ground truth joint locations. We train our model in an end-to-end manner using paired 2D-3D sequences. In the diagram, letters F and R correspond to fully connected feedforward and recurrent layers, respectively. Dashed line denotes residual connection.

Figure 5 illustrates our proposed framework, including the architecture of our model. Our model takes 2D joint locations as input and estimates corresponding 3D joint angles—pose parameters $\theta$ of the SMPL model. We train our models by projecting 3D pose sequences onto 2D to create paired samples to be used for supervision. At test time, we assume that we are only provided with 2D joint estimations of an off-the-shelf pose estimator and nothing else. This particular assumption presents some additional challenges in addition to the inaccuracies of the utilized 2D estimator as well as the inherent challenges of the task (e.g. lack of access to silhouettes or joint detection heatmaps). Methods we employ to address these challenges are detailed in the following sections.

4.1 Model input and output

We aim to train dataset-agnostic models and to that end, we evaluate our models in cross-dataset settings. That means the models we train ought to accommodate 2D joint position annotations on different datasets. This is an issue we have to take into consideration as we assume no bounding boxes are presented. Thus, 2D joints can be found anywhere in an image, for images of any size, which results in a scale ambiguity on our inputs. To tackle this problem, we uniformly scale every 2D input skeleton to fit a unit-square, in a frame-wise manner. Figure 6 illustrates two examples of this process.

A similar preprocessing on 2D joints has been explored before by Yasin et al. [94]. They always normalize joints so that joint coordinates in $y$-axis fit $[-1, 1]$ range. Note that we use scaling and normalization interchangeably when referring to this preprocessing step on 2D inputs.
Our assumption on 2D input format presents two more additional handicaps for our models. Firstly, we do not know the gender of the subject and cannot predict it using only the skeleton input. Thus, we always use a gender-neutral model, following the earlier works [34, 39]. Secondly, we are not provided with the images themselves or any segmentation masks, which means much of the information on shape is also lost. Therefore, we do not estimate shape parameters $\beta$ while predicting and always use the default body shape.

Our training data is 3D-only, i.e. we only have 3D meshes but no corresponding 2D representations, so we obtain 2D joints by projection\(^1\). In training time we place the 3D skeleton in origin (i.e. aligned by its pelvis), project the 3D joints onto a perspective camera with a fixed depth, and normalize the projected 2D joints as described above. We use a perspective projection on a camera positioned in $(0,0,6)$ with focal lengths 256 in both $x$ and $y$ to obtain 2D poses from 3D.

Similarly, we obtain 3D joint positions manually from 3D meshes. We follow the previous line of work [8, 41, 34] and use pretrained linear regressors to obtain the desired set of 3D joints, given an input SMPL mesh. Different skeleton formats correspond to different joint regressors $W \in \mathbb{R}^{k \times 6890}$ that linearly weights 6890 3D vertices of an SMPL mesh in order to estimate 3D locations of $k$ joints. We use these regressors not only to produce training data but also to estimate 3D joint locations from predicted 3D meshes, again in line with the previous work.

To summarize, our model predicts 3D joint angles from normalized 2D joint positions. Then, we calculate the SMPL mesh corresponding to the joint angles and estimate 3D joint positions on the predicted mesh via a pretrained joint regressor. This pipeline is illustrated in Figure 7.

\(^1\)This discussion is mainly aimed towards utilizing AMASS, but we also use the same methodology when training using MoSh [46] version Human3.6M data. See also Section 5.1 for more discussion on this topic.
Figure 7: **Forward kinematics pipeline:** We illustrate how each representation is obtained from the previous one, in a consecutive manner. We use this pipeline to obtain paired data instances from 3D mocap data, as well as obtaining estimated joint positions for loss calculation and later for evaluation. We note that all three steps of this pipeline is differentiable and each representation can be used for loss calculation during training phase. If the 2D skeleton format is different than that of the 3D, as it is the case with the illustrated sample, we also calculate an intermediary 3D skeleton that matches the 2D for the projection.

Additionally, we use alternative rotation formulations (see Section 2.2.2). SMPL body model parameterizes joint rotations through axis-angles, which might be problematic for neural networks. Therefore, we optionally predict rotations in other formulations and convert them back to axis-angles to calculate forward kinematics. This gives the model the opportunity to calculate losses on rotation values using different formulations.

We follow the work of Aksan et al. [3] on motion prediction and calculate our losses in a joint-wise manner before summing them over complete sequences. Equation 6 shows how the loss function we use decomposes into three terms, respectively as losses on 3D joint angles $L_{rot}$, 3D joint positions $L_{3D}$ and 2D joint positions $L_{2D}$ through reprojection. We weight these loss terms by empirically chosen loss coefficients $\lambda_{rot}$, $\lambda_{3D}$ and $\lambda_{2D}$.

$$L = \lambda_{rot}L_{rot} + \lambda_{3D}L_{3D} + \lambda_{2D}L_{2D}$$

$$= \lambda_{rot}L_{base}(R, \hat{R}) + \lambda_{3D}L_{base}(J, \hat{J}) + \lambda_{2D}L_{base}(j, \hat{j})$$  \hspace{1cm} (6)

We further illustrate how we compute this joint-wise loss in Equation 7. We use the same convention as before: $R^{(t)}_i$ denotes rotation of the $i^{th}$ joint in frame $t$ in the ground truth sequence and $\hat{R}^{(t)}_i$ is that of the estimated sequence. The example shows how a loss function $f$ is calculated on individual joints separately, before being summed up separate frames. In this work, we always use mean squared error (MSE) as the loss function $f$.

$$L_{base}(R, \hat{R}) = \frac{1}{T} \sum_{t=1}^{T} \sum_{n=1}^{N} f(R^{(t)}_i, \hat{R}^{(t)}_i)$$  \hspace{1cm} (7)
### 4.2 Spatio-temporal masking

![Joint masking on 2D inputs](image)

**Figure 8: Joint masking on 2D inputs:** We illustrate masked joints on four consecutive 2D poses from AMASS. Frames denote a unit box in $xy$-coordinates, points denote valid (i.e. unmasked) joints and limbs are drawn only if both ends are valid joints. We mask joints in both spatial and temporal dimensions uniformly random.

Our aim is to train a model that can utilize predictions of an off-the-shelf 2D pose estimator as input. Such models can account for the uncertainties in their predictions in different ways (e.g. confidence values, heatmaps) or they might not express any uncertainty at all. Similarly, as an extreme case of uncertainty, the model might not be able to estimate positions of certain joints and present a skeleton with missing joints. Therefore, we take these model differences into account and train our networks according to the type of model to be used in test time.

If the model is expected to use the outputs of an estimator that only reports joint locations, we also provide only the joint locations during training. On the other hand, if the 2D estimator model provides us with confidence values for individual joint estimations we also append those values alongside the 2D joint coordinates in input. For our model to learn how to utilize these confidence values and address missing joints at test time, we modify our input during training.

We use a simplified approach to generate input poses with uncertainty and use binarized confidence values. We generate random confidence arrays during training — that is we drop out random joints and set their confidences to 0, where confidence values of other (valid) joints remaining at 1. Such spatiotemporal masking of joints has previously
been explored and found beneficial for pose estimation [12] as well as motion modeling [71].

Figure 8 illustrates our joint masking approach on pose samples. At test time we also accommodate this approach by first setting a threshold and then binarizing the joints according to the set threshold. This essentially means discarding low fidelity joints and fullycommitting to the rest.

Additionally, we experiment with masking entire frames, alongside individual joints. The existence of missing frames at test time is not a problem for frame-wise models as they make their predictions separately for each frame. However, this is not the case for temporal models as they do their predictions over sequences and they should be able to accommodate occasional missing frame occurrences.

### 4.3 Augmentation on 3D data

![Figure 9: Data augmentation by rotation](image)

**Figure 9:** Data augmentation by rotation: Top row shows ten consecutive frames from the original sequence and two bottom rows illustrate augmented samples obtained by random rotations around y-axis.

One major advantage of using 2D poses rather than RGB images as input is the relative simplicity of data augmentation. This is because 2D poses exist in a feature space of much smaller dimensions than that of the images.

In training time, we obtain paired data by projecting 3D poses to 2D. That means we can augment only the 3D data to get novel paired instances. The easiest way to that would be to rotate the body around the vertical axis. Figure 9 illustrates how we augment data by random rotations. Since the relative rotations of the joints stay the same, we only change the global rotation vector (i.e. rotation of the hip joint) in each training batch. We sample a rotation angle for each sequence in the batch from a uniform random distribution.
This particular approach for generating 2D-3D pose pairs has been explored many times in previous works [94, 10, 58]. The only difference of our augmentation is in terms of its practical implementation: preceding works generate virtual cameras around a stationary body model whereas we globally rotate the centered body models in front of a fixed camera.

![Figure 10: Data augmentation by mirroring](image)

**Figure 10: Data augmentation by mirroring**: Top row shows ten consecutive frames from the original sequence and two bottom rows illustrate augmented samples obtained by mirroring the left and right parts of the body.

Similarly, another way for realistically altering the 3D pose is mirroring its left and right halves. Here, we preserve the global rotation but change the joint rotations. Specifically, we first swap all the corresponding rotations of the left and right joints, then mirror them. Mirroring is quite straightforward as the model uses relative rotations instead of global ones. Result of such data augmentation by mirroring can be seen in Figure 10.

### 4.4 Regularization by noisy inputs

![Figure 11: Random noise on 2D inputs](image)

**Figure 11: Random noise on 2D inputs**: We illustrate the effects of additive noise on one example frame from AMASS, where frames denote a unit box in \( xy \)-coordinates. Perturbations of greater scale distorts both pose and shape of the 2D skeleton.
Alternatively, we can augment data pairs by changing only 2D data itself, after obtaining it by projecting 3D pose onto the camera plane. This is especially useful as our aim is not to generate perfect 2D/3D data pairs, but to generate samples that multiply our training data realistically—with respect to our test conditions. At test time, we obtain 2D inputs through off-the-shelf pose detectors. Their outputs can be unreliable in various ways and we should account for these deviations from ground truth joint locations.

Injecting noise directly on 2D joint locations during training can be used to simulate the inaccuracies of the used 2D pose estimator. A fast and straightforward way to do is to employ additive random noise sampled from a Gaussian distribution. We sample a noise vector for every frame of each sequence in a batch, apply such random noise on input poses and renormalize them for 2D joints to fit the unit square. Figure 11 illustrates the 2D joint locations after applying random noise of different scales.

Random noise injection is known to be a useful tool to control bias-variance trade-off. The earliest examples of analysing the effects of noise injection go back to the work of Sietsma et al. [75]. Authors found experimentally that using noisy inputs had improved the generalization capability of the networks with fully connected layers.
5 Experiments

We evaluate our methodology on two widely used evaluation scenarios, namely Human3.6M and 3DPW datasets. Section 5.1 details the specifics of our training and evaluation procedures. Section 5.2 compares the effects of different complementary approaches we utilize. Section 5.3 compares our approaches to state-of-the-art 3D pose estimation methods. Section 5.4 discusses some of the alternative approaches we pursued that did not result in successful results.

5.1 Experimental setup

Here, we go over the details of our preprocessing procedure, training routine and evaluation scenarios. We detail how and where we utilize each dataset in our framework. Reader may refer to Sections 2.4 and 2.2.1 for the preliminary discussions on the datasets and body models we use, respectively.

- **Converting SMPL-H to SMPL:** We always use the original SMPL parameterization instead of its variants throughout our experiments. Since AMASS uses SMPL-H, we convert both shape and pose parameters accordingly for SMPL, in a straightforward manner. SMPL-H and SMPL have equal shape spaces, so first 10 parameters of the 16 SMPL-H shape space corresponds to shape parameters $\beta$ of the SMPL model. Similarly, SMPL-H and SMPL share the same body joints and only differences are the hand effectors. Following the work of Kocabas et al. [38], we copy the parameters for the body pose and set the angles of the hand effectors as rotations of the first joints on respective index fingers. This process is illustrated in the snippet in Listing 1.

```python
betas = betas[:10]
thetas = np.concatenate([thetas[:, :66], thetas[:, 66:69], thetas[:, 111:114]], axis=1)
```

**Listing 1:** Converting SMPL-H to SMPL

- **Training and evaluation on Human3.6M:** Following the work of Kanazawa et al. [34], we use the MoSh [46] version of the Human3.6M dataset for training, which provides SMPL parameters obtained from raw 3D mocap markers of the original dataset. Having access to SMPL parameters allows us to augment data during training. That being said, evaluation is done accordingly on the publicly available 3D joint locations of the test split. We obtain 2D detections for Human3.6M test sequences using the hourglass model of Newell et al. [54], trained again on Human3.6M itself.

We test our models on Human3.6M under two evaluation scenarios, named as protocol 1 and protocol 2. There is not an unanimity on the exact definition of these two protocols, so we adhere to their recent definitions by Kanazawa et al. [35] for compatibility. For both protocols, they train their models on five subjects (S1, S5, S6, S7 and S8) and evaluate on subjects S9 and S11. In protocol 1, they use pose
sequences from all cameras in evaluation both they only use the frontal camera (i.e. camera 3) when evaluating under protocol 2. Consistently with the previous work, we report MPJPE for protocol 1 and PA-MPJPE for protocol 2.

- **Evaluation on 3DPW:** For 3DPW, we use the 2D detections provided in the dataset files, which were obtained using OpenPose [9] detector. 2D detections are provided alongside confidence values for each joint detection. Zero confidence values indicates missing joints and authors only provide detections if at least 6 joints were detected correctly for a frame —otherwise they provide a frame without any joint detections. When evaluating our models, we follow the previous line of work [35] and aggregate metrics only on valid frames, i.e. we discard frames with less than 6 joints. We use a confidence threshold of 0.1 to separate valid (reliable) joint detections from invalid (unreliable) ones.

3DPW dataset provides 3D joint positions in SMPL skeleton format. Kanazawa et al. [35] compute a new set of 3D joints in H36M skeleton format and evaluate their approaches against these set of joints. They do this computation via a joint regressor (see Section 4) on gendered ground truth SMPL meshes of the 3DPW dataset. We follow their approach regarding this as well.

- **Downsampling and window extraction:** We downsample all datasets to the same framerate —10 frames per second —to be able to provide settings comparable to recent work on temporal human pose estimation [35, 38]. We both train and evaluate our models on batched inputs, so we split the pose sequences into smaller “windows” of fixed lengths —sequences of 30 frames. During training, we sample windows randomly from sequences of poses in each pass. For validation, we only use the windows from the middle of the each sequence. At test time, we exhaustively slice the sequences into non-overlapping, consecutive windows.

- **Dataset splits:** For Human3.6M, we use subjects S1, S5, S6, S7 and S8 as the training split and subjects S9 and S11 as the test split, as discussed above. We split AMASS into three as training, validation and test subsets. We randomly shuffle the dataset once during preprocessing and allocate 90% of it for training purposes. Remaining pose sequences are then split into two as validation and test sets. We use validation subset of AMASS to validate the models we train for Human3.6M evaluation. Similarly, we also evaluate those models on test split of AMASS to compare the trends in these two datasets. For 3DPW, we only report metrics on the test split. We train our models for 3DPW on the entirety of AMASS plus the training split of Human3.6M and we use its respective validation split for checkpointing our models.

- **Training procedure:** We use mini-batches of size 32 while training on both datasets. We periodically validate our models and save the best performing checkpoints with respect to PA-MPJPE error on validation samples. We then evaluate our models on test datasets using these model checkpoints. We use Adam optimizer [37] with a learning rate of $1 \times 10^{-3}$. We train different models for evaluation on 3DPW and Human3.6M, as they respectively use COCO and H36M skeleton formats. We calculate loss on reprojected 2D joints $L_{2D}$ before normalizing the joints. We use pelvis-aligned 3D joints (as in both MPJPE and PA-MPJPE) metrics before calculating $L_{3D}$. We do
not employ dropout anywhere in our model. Unless stated otherwise, we use the respective settings listed in Appendix A.3 for our experiments on Human3.6M and 3DPW datasets. To augment the data by mirroring, we append all the mirrored sequences to the original training dataset and cut the number of epochs in half.

5.2 Results

In this subsection, we evaluate the performance of our models both qualitatively and quantitatively. We evaluate the effects of different methodologies we employ under controlled settings. We should note that groups of experimental results we report are consistent within themselves but not directly comparable to each other, due to technical difficulties. That means, an identical setting might be attributed with different results throughout this subsection as those groups of experiments were conducted on different machines. Appendix A.2 include some sample predictions of our network on sequences from 3DPW test split.

5.2.1 Temporal modeling and rotations

<table>
<thead>
<tr>
<th>Rot. Format</th>
<th>PA-MPJPE</th>
<th>MPJPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Axis-Angles</td>
<td>85.05</td>
<td>144.7</td>
</tr>
<tr>
<td>Quaternion</td>
<td>74.09</td>
<td>129.2</td>
</tr>
<tr>
<td>Rotation Matrix</td>
<td><strong>64.35</strong></td>
<td><strong>111.1</strong></td>
</tr>
<tr>
<td>rot6d [97]</td>
<td>65.08</td>
<td>112.2</td>
</tr>
</tbody>
</table>

Table 1: Evaluation of rotation formats on 3DPW: We compare our methods on four rotation formats. We employ transformation functions that convert the predicted rotations into axis-angles for the calculation of joint positions. For quaternions, we also employ normalization before conversion, which helps the performance.

We experimented with different formalisms to represent rotations. A summary of our evaluation can be viewed in Table 1. Although this exact set of results might not be seen as representative of the each format's potential (as different formats may prefer different hyperparameters, loss functions etc.), we believe it satisfactorily reflects our findings in our preliminary experiments.

We consistently obtained our best results with rotation matrices and 6D continuous representations [97] (dubbed rot6d). Compared to axis-angles and quaternions, these two representations were both faster and also more stable during training.

We also evaluate the effect of having a recurrent layer in our architecture —which was illustrated in Figure 5. Table 2 shows the comparison of a temporal model with a non-temporal one on 3DPW dataset. Former has a GRU layer between feedforward layers and the latter does not.
5 EXPERIMENTS

<table>
<thead>
<tr>
<th>Random noise</th>
<th>Temporal</th>
<th>PA-MPJPE</th>
<th>MPJPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>No</td>
<td>71.36</td>
<td>114.4</td>
</tr>
<tr>
<td></td>
<td>Yes</td>
<td>69.46</td>
<td>111.3</td>
</tr>
<tr>
<td>0.01</td>
<td>No</td>
<td>69.62</td>
<td>110.5</td>
</tr>
<tr>
<td></td>
<td>Yes</td>
<td>68.35</td>
<td>108.8</td>
</tr>
<tr>
<td>0.02</td>
<td>No</td>
<td>64.90</td>
<td>108.9</td>
</tr>
<tr>
<td></td>
<td>Yes</td>
<td>65.56</td>
<td>102.3</td>
</tr>
</tbody>
</table>

Table 2: Evaluation of temporal modeling on 3DPW: Inclusion of the recurrent layer is generally useful, but this increase in performance is not consistent throughout different settings.

We see that while it is generally beneficial for us to utilize a recurrent layer, performance gain is only marginal. Yet, we believe there is much space for improvement regarding temporal modeling and our qualitative analysis also confirms the shortcomings of our current temporal models. We illustrate one such failure case in Figure 13, in Appendix A.2. We note that we did not see any advantage in increasing the number of recurrent layers in our current methodology. Similarly, our experiments with bidirectional recurrent layers were unfruitful.

5.2.2 Data augmentation

<table>
<thead>
<tr>
<th>Setting</th>
<th>Augmentation</th>
<th>PA-MPJPE</th>
<th>MPJPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Evaluation on Human3.6M test</td>
<td>None Rotation</td>
<td>51.38</td>
<td>70.41</td>
</tr>
<tr>
<td></td>
<td>Rotation</td>
<td>50.83</td>
<td>71.95</td>
</tr>
<tr>
<td>Evaluation on AMASS test</td>
<td>None Rotation</td>
<td>39.33</td>
<td>64.17</td>
</tr>
<tr>
<td></td>
<td>Rotation</td>
<td>33.07</td>
<td>44.05</td>
</tr>
</tbody>
</table>

Table 3: Evaluation of rotation augmentation on Human3.6M and AMASS: Two compared models are identical except one augments input by rotation and the other does not. Our models can overfit to subset of AMASS with relative ease, rotation augmentation is visibly helpful on this.

Now, we evaluate our methods for data augmentation. Table 3 examines the effects of data augmentation through random rotations on Human3.6M and AMASS test splits, whereas Table 4 compares the results obtained with different augmentation methods on 3DPW.

Looking at Table 3, we can see that augmentation clearly helps the performance on AMASS, while the difference is only marginal on Human3.6M. Similarly, in Table 4, we can see the effect of rotation augmentation is again inconsistent on 3DPW —rotation augmentation decreases the PA-MPJPE at the expense of MPJPE. On the other hand, our
5 EXPERIMENTS

<table>
<thead>
<tr>
<th>Setting</th>
<th>Augmentation</th>
<th>PA-MPJPE</th>
<th>MPJPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>No random noise</td>
<td>None</td>
<td>70.13</td>
<td>115.0</td>
</tr>
<tr>
<td>Joint masking 0.2</td>
<td>Rotation</td>
<td>71.94</td>
<td>118.2</td>
</tr>
<tr>
<td></td>
<td>Mirroring</td>
<td>69.64</td>
<td>111.8</td>
</tr>
<tr>
<td></td>
<td>Mirror. &amp; Rot.</td>
<td>69.46</td>
<td>111.3</td>
</tr>
<tr>
<td>Random noise 0.04</td>
<td>None</td>
<td>63.29</td>
<td>108.2</td>
</tr>
<tr>
<td>Joint masking 0.2</td>
<td>Rotation</td>
<td>61.95</td>
<td>111.3</td>
</tr>
<tr>
<td></td>
<td>Mirroring</td>
<td>62.95</td>
<td>105.9</td>
</tr>
<tr>
<td></td>
<td>Mirror. &amp; Rot.</td>
<td>62.18</td>
<td>106.4</td>
</tr>
<tr>
<td>Random noise 0.04</td>
<td>None</td>
<td>64.3</td>
<td>108.6</td>
</tr>
<tr>
<td>Joint masking 0.1</td>
<td>Rotation</td>
<td>63.05</td>
<td>116.0</td>
</tr>
<tr>
<td></td>
<td>Mirroring</td>
<td>63.68</td>
<td><strong>105.4</strong></td>
</tr>
<tr>
<td></td>
<td>Mirror. &amp; Rot.</td>
<td>62.38</td>
<td>108.2</td>
</tr>
</tbody>
</table>

Table 4: Evaluation of data augmentation methods on 3DPW: All models are trained on AMASS and Human3.6M. Rotation augmentation is done randomly on-the-fly, whereas for mirroring we expand the datasets by concatenating all the mirrored versions. We cut the number of epochs in half when using mirroring augmentation.

experiments show that mirroring results in consistent improvements in both metrics, under different settings.

Rotation augmentation only results in variability on one root “joint” (i.e. global rotation), whereas mirroring completely modifies skeleton by changing the rotations of all but one joint —although both augmentation techniques generate novel 2D poses. We believe this is why mirroring improves performance more consistently as it generates realistic and also more diverse sets of pose sequences than rotation. Considering the results of our experimentation in this section, we always apply both rotation and mirroring augmentations in the following experiments.

5.2.3 Noise and masking

Here, we evaluate the effects of regularization by random noise on the inputs. Table 5 shows results on Human3.6M and AMASS test splits, whereas Table 6 illustrates 3DPW—where we also employ joint masking.

Spatiotemporal masking is necessary for us in the 3DPW evaluation scenario, as there exists missing joints in 2D detections at test time. We evaluate dropping joints with different probabilities and find 0.2 is what works best for us. Random noise seems to help on both Human3.6M on 3DPW, but models for different evaluation scenarios benefit from different amount of noise. More specifically, we obtain best results for Human3.6M and 3DPW respectively using coefficients 0.01 and 0.04 to scale the noise vectors.

This is partly due to detection qualities, as we need larger perturbations to better simulate more challenging test conditions of 3DPW. Another reason that we obtain best results on 3DPW with considerably large perturbations (cf. Figure 11) is that we independently
5 EXPERIMENTS

### Table 5: Evaluation of random noise on Human3.6M and AMASS:

Injecting random noise on small scales is useful for evaluation on Human3.6M, where we rely on inaccurate joint detections as input. This is not the case for AMASS as 2D inputs are always in perfect condition and any noise injection during training hinders the performance.

<table>
<thead>
<tr>
<th>Setting</th>
<th>Random noise</th>
<th>PA-MPJPE</th>
<th>MPJPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Evaluation on</td>
<td>None</td>
<td>53.31</td>
<td>76.97</td>
</tr>
<tr>
<td>Human3.6M test</td>
<td>0.01</td>
<td><strong>51.04</strong></td>
<td><strong>72.05</strong></td>
</tr>
<tr>
<td>Evaluation on</td>
<td>None</td>
<td><strong>29.18</strong></td>
<td><strong>38.10</strong></td>
</tr>
<tr>
<td>AMASS test</td>
<td>0.01</td>
<td>33.14</td>
<td>44.38</td>
</tr>
</tbody>
</table>

### Table 6: Evaluation of random noise and joint masking on 3DPW:

Models overfit to training AMASS data without any random noise, so our models always benefit from regularization. Higher amounts of noise can help with the PA-MPJPE but this can be at the expense of performance with respect to MPJPE metric.

<table>
<thead>
<tr>
<th>Joint masking</th>
<th>Random noise</th>
<th>PA-MPJPE</th>
<th>MPJPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.3</td>
<td>0.04</td>
<td><strong>62.15</strong></td>
<td>108.3</td>
</tr>
<tr>
<td></td>
<td>0.02</td>
<td>67.85</td>
<td>105.6</td>
</tr>
<tr>
<td></td>
<td>None</td>
<td>72.03</td>
<td>109.9</td>
</tr>
<tr>
<td>0.2</td>
<td>0.08</td>
<td>64.77</td>
<td>120.8</td>
</tr>
<tr>
<td></td>
<td>0.06</td>
<td>62.99</td>
<td>115.2</td>
</tr>
<tr>
<td></td>
<td>0.04</td>
<td>62.18</td>
<td>106.4</td>
</tr>
<tr>
<td></td>
<td>0.02</td>
<td>65.56</td>
<td><strong>102.3</strong></td>
</tr>
<tr>
<td></td>
<td>None</td>
<td>69.46</td>
<td>111.3</td>
</tr>
<tr>
<td>0.1</td>
<td>0.04</td>
<td>62.38</td>
<td>108.2</td>
</tr>
<tr>
<td></td>
<td>0.02</td>
<td>65.77</td>
<td>104.0</td>
</tr>
<tr>
<td></td>
<td>None</td>
<td>69.43</td>
<td>120.2</td>
</tr>
</tbody>
</table>

generate random noise vectors and joints masks, effectively hindering the regularization effects of perturbations on 3DPW. This is a limitation of our simplistic assumption that missing and inaccurate joints are distributed independently through spatial and temporal dimensions, which is not the case with the actual 2D joint detections.

Our preliminary experiments with masking complete frames in addition to masking individual joints did not bring any improvement in performance, so we did not employ frame masking in any of our later experiments. We experimented with dropping random individual frames, dropping contiguous frames of random sizes and combination of those. Neither option was fruitful in the presence of joint masking. We believe more powerful temporal models can better utilize aforementioned frame masking methods.
### 5.3 Comparison with state-of-the-art

<table>
<thead>
<tr>
<th>Models</th>
<th>PA-MPJPE ↓</th>
<th>MPJPE ↓</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kanazawa et al. [34]</td>
<td>56.8</td>
<td>88</td>
</tr>
<tr>
<td>Omran et al. [56]</td>
<td>59.9</td>
<td>-</td>
</tr>
<tr>
<td>Kolotouros et al. [40]</td>
<td>50.1</td>
<td>-</td>
</tr>
<tr>
<td>Arnab et al. [6]</td>
<td>54.3</td>
<td>77.8</td>
</tr>
<tr>
<td>Kolotouros et al. [39]</td>
<td><strong>41.1</strong></td>
<td>-</td>
</tr>
<tr>
<td>Kanazawa et al. [35]</td>
<td>56.9</td>
<td>-</td>
</tr>
<tr>
<td>Doersch et al. [18]</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Sun et al. [81]</td>
<td>42.4</td>
<td><strong>59.1</strong></td>
</tr>
<tr>
<td>Kocabas et al. [38] (d.c.)</td>
<td>41.5</td>
<td>65.9</td>
</tr>
<tr>
<td>Kocabas et al. [38]</td>
<td><strong>41.4</strong></td>
<td>65.6</td>
</tr>
<tr>
<td>Ours, 2D detections</td>
<td>50.83</td>
<td>71.95</td>
</tr>
<tr>
<td>Ours, ground truth 2D joints</td>
<td>44.09</td>
<td>73.60</td>
</tr>
<tr>
<td>Ours, 2D detections (A)</td>
<td>60.00</td>
<td>90.06</td>
</tr>
<tr>
<td>Ours, ground truth 2D joints (A)</td>
<td>51.06</td>
<td>92.85</td>
</tr>
</tbody>
</table>

**Table 7: Benchmark of the state-of-the-art models on H36M:** We train our models on the training split of the Human3.6M, ones marked with an (A) on the side were trained exclusively on AMASS training split. PA-MPJPE is reported on protocol 2 and MPJPE is reported on protocol 1. We obtain 2D detections from Stacked Hourglass model [54].

Here, we compare our approaches to state-of-the-art methods on Human3.6M and 3DPW benchmarks, respectively in Tables 7 and 8. In both tables, reported results are grouped into two as non-temporal and temporal methods. First thing to notice is that we obtain similar positions in both evaluation scenarios with respect to compared approaches. We obtain competitive results, yet we cannot improve the current state-of-the-art in either benchmark. We note that our model is the only two-stage model among the compared works and our performance is always constrained by the quality of provided 2D joint predictions.

Evaluating our model on Human3.6M using ground truth 2D joints instead of detections moves us closer to reference models, but still fails to overperform them. We also report our results where we only train on AMASS. Training or pretraining models using AMASS only results in worse performance on Human3.6M, although AMASS contains much greater scale and variety. We note that similar findings have been noted in recent works [38, 35] and this result is in line with our expectations.

Our focus is the more challenging in-the-wild scenario, 3DPW, where we manage...
 Models | PA-MPJPE ↓ | MPJPE ↓ |
--- | --- | --- |
Kanazawa et al. [34] | 76.7 | 130.0 |
Omran et al. [56] | - | - |
Kolotouros et al. [40] | 70.2 | - |
Arnab et al. [6] | 72.2 | - |
Kolotouros et al. [39] | **59.2** | **96.9** |
Kanazawa et al. [35] | 72.6 | 116.5 |
Doersch et al. [18] | 74.7 | - |
Sun et al. [81] | 69.5 | - |
Kocabas et al. [38] (d.c.) | 56.5 | 93.5 |
Kocabas et al. [38] | **51.9** | **82.9** |
Ours, 2D detections | 61.03 | 107.5 |
Ours, ground truth 2D joints | 45.10 | 100.1 |
Ours, ground truth 2D joints and shape (β) | 44.81 | 98.08 |

Table 8: Benchmark of the state-of-the-art models on 3DPW: We train our models on entirety AMASS and the training split of the Human3.6M. We obtain 2D detections from OpenPose [9].

The final set of results reported here uses a number of different modeling choices compared to the rest of our experiments on 3DPW. Namely we use random noise of 0.03 instead of 0.04 and a recurrent cell size of 2048 instead of 1024. Lastly, we replace ReLU activation function with Randomized Leaky ReLU (RReLU).

5.4 Unsuccessful methods

In this subsection, we briefly talk about some of our failed approaches that aimed to address the shortcomings of our models. We believe similar approaches can benefit from our investigations discussed here.

3 The final set of results reported here uses a number of different modeling choices compared to the rest of our experiments on 3DPW. Namely we use random noise of 0.03 instead of 0.04 and a recurrent cell size of 2048 instead of 1024. Lastly, we replace ReLU activation function with Randomized Leaky ReLU (RReLU).
5.4.1 Input rescaling

We have discussed earlier why we have needed to normalize 2D joint position inputs and how this process results in loss of scale information. The distortion might become greater when there are some missing joints, especially extremities, which is illustrated in Figure 6 on two consecutive frames of a pose sequence.

Inspired by Jaderberg et al. [30], we aimed to use a part of our network to spatially transform to inputs, that is, scale and translate the provided 2D joints. Our aim was to make the model learn how to properly scale the provided 2D pose sequences, with respect to a reference scale. Such reference scale for us was the scale of 2D joint locations obtained by the projection of pelvis-centered 3D pose.

We provided inputs to this transformation module in different formats during our experiments. Apart from the normalized inputs (cf. Figure 6), we experimented with inputs centered by the mean of the valid joints and ones that were mean-centered and then normalized in y-axis. Our experiments on learning the transformation parameters (either implicitly or through explicit supervision) were unfruitful as the model tended to disregard our concept of rescaling even when directly supervised.

5.4.2 Hybrid architecture

Some recent works on pose estimation have shown that incorporating temporal knowledge improves 3D pose estimations by generating smoother and more realistic body movements (see Section 3.3). Thus, one of our aims in this project was to exploit motion dynamics for a pose estimation system using AMASS—as recent work has shown that AMASS is not only larger, but also more diverse than preceding datasets for motion modeling [3].

![Diagram of Hybrid Architecture](image)

**Figure 12: Hybrid architecture for pose estimation:** Our hybrid architecture (b) combines two streams: a standard 2D-to-3D lifting model (a) and a simple motion model with a residual connection [49].

We tried to unify a simple motion modeling approach with our pose estimation model in a single network. We call this a hybrid architecture, against our standard architecture—the two are comparatively illustrated in Figure 12. Our intuition was to force temporal
consistency through auto-regressive modeling. At test time, hybrid model uses its own predictions in an auto-regressive manner estimate a 3D pose sequence. We do not have access to ground truth at test time, but we can employ different sampling schemes during training. We experiment with four different scenarios for training:

1. Always using own predictions.
2. Always using ground truth 3D poses instead of own predictions, known as teacher forcing in literature.
3. Randomly sampling from ground truth or estimated poses at each step, a method known as scheduled sampling [7].
4. Alternating between using ground truth and estimated poses at regular intervals, a method named auto-conditioned recurrent neural network (acRNN) by its authors [43].

We consistently obtained the best results with always using own predictions and worse results with teacher forcing. Using own predictions requires step-wise estimation that slows down the training, so the acRNN approach can be used to speed it up while trading performance. As it is the case with the standard model, we obtained better results using rotation matrices and rot6d to represent rotations on hybrid architecture. However, hybrid models we trained always underperformed standard models in comparable settings. We believe explicitly modeling motion dynamics on the network's own 3D estimations rather than proper sequences that exhibit temporal consistency is a difficult task. We believe with modeling modifications and a careful training procedure, the hybrid model can encode temporal dynamics more successfully.
6 Conclusion

In this thesis, we proposed a novel framework that incorporates basic compatible methods to address the challenges of in-the-wild 3D pose estimation. Methodologies we employ made it possible for our models to further utilize the information in mocap datasets. Through our approach, we have demonstrated that 3D mocap data alone can suffice to train effective pose estimation models. We have conducted an empirical study to assess the strengths and shortcomings of our methodology.

6.1 Future work

In the previous sections of the thesis, we have discussed the results of our experimental analysis on the different approaches we employ. Based on our findings, we suggest following lines of work as future directions for research.

- **Temporal modeling:** Although our temporal models have produced satisfactory results, we believe there is still much room for improvement and motion modeling literature could be the most appropriate place to look for inspiration. Some recent examples include powerful models that utilize spatiotemporal context through generative adversarial networks [71] or transformers with decoupled self-attention mechanisms [2]. Introducing such architectures to our framework could help our models obtain temporally smooth and consistent pose estimations and also make them more robust to the shortcomings of the 2D pose estimators.

- **Synthetic data:** It is cheap to generate synthetic data in very large scales [17, 19] and creating only 3D meshes an even simpler task. Datasets of much greater size and variety would surely be beneficial, but an expected obstacle would be the problem of domain adaptation—which is an issue even on mocap datasets. We believe that we can use mocap and synthetic 3D data together through ideas from transfer learning literature, such as using adversarial learning for domain adaptation [83]. Thus, reproducing our experimental analysis on synthetic datasets could be insightful for a better understanding of the aforementioned setting and its prospective usage.

- **Training both stages:** AMASS offers a great prospect for 3D pose estimation through its currently incomparable size. Its potential has only been explored via adversarial training [38], but not through direct supervision. Training both parts of the two-stage network together and utilizing AMASS to additionally train the second part through our methodology would be a natural extension of this work. Such end-to-end training could also be beneficial for the first stage of the network. Temporal 3D pose estimations can be used to fill the missing 2D joint detections (through lifting and reprojection) to provide weak supervision to the 2D pose estimator model. Ideally, we would like to exploit weakly labeled internet-scale datasets [35] together with AMASS to train more powerful two-stage models.

- **Better augmentation:** Our approaches to data augmentation rely on simplistic assumptions and use straightforward techniques. These can be done more eloquently to realistically adapt our models to test conditions. Currently, we disregard the spa-
tiotemporal structure when masking or injecting noise, whereas 2D pose estimators we utilize oftentimes provide erroneous estimations for joints in spatiotemporally close regions. Similarly, we sample masks and noise vectors separately and we do not attempt to simulate confidence values. We believe simulation of 2D estimation errors could be unified for a more realistic and effective augmentation strategy. For example, this could be done in a data-driven manner by analysing the predictions of estimator models on large-scale 2D datasets. Lastly, paying attention to self occlusions of 3D models during training is another simple idea for obtaining more realistic samples.
## A Appendix

### A.1 Comparison of skeletons

<table>
<thead>
<tr>
<th>Joint</th>
<th>H36M_17</th>
<th>H36M_14</th>
<th>COCO</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hip</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>LRHip</td>
<td>✓✓</td>
<td>✓✓</td>
<td>✓✓</td>
</tr>
<tr>
<td>LRKnee</td>
<td>✓✓</td>
<td>✓✓</td>
<td>✓✓</td>
</tr>
<tr>
<td>LRFoot</td>
<td>✓✓</td>
<td>✓✓</td>
<td>✓✓</td>
</tr>
<tr>
<td>LRShoulder</td>
<td>✓✓</td>
<td>✓✓</td>
<td>✓✓</td>
</tr>
<tr>
<td>LRElbow</td>
<td>✓✓</td>
<td>✓✓</td>
<td>✓✓</td>
</tr>
<tr>
<td>LRWrist</td>
<td>✓✓</td>
<td>✓✓</td>
<td>✓✓</td>
</tr>
<tr>
<td>Spine</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Neck</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Nose</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Head</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>LREye</td>
<td>xx</td>
<td>xx</td>
<td>✓✓</td>
</tr>
<tr>
<td>LREar</td>
<td>xx</td>
<td>xx</td>
<td>✓✓</td>
</tr>
</tbody>
</table>

**Table 9: Comparison of skeletons:** We compare the joint that compose three skeletons, namely the common 17-joint H36M skeleton with only moving joints out of original 32, reduced 14-joint H36M skeleton we use that corresponds to LSP format and the COCO skeleton that shares the same body joints with the 14-joint H36M skeleton. LR prefix before a joint name indicates that joint exist in left-right pairs.
A.2 Sample estimations

Figure 13: Failure mode for temporal prediction: Last 12 consecutive frames from a 30-frame temporally inconsistent sequence prediction for a 3DPW test sample. Bottom row shows estimated 3D poses overlaid on respective ground truth skeletons. Timesteps with missing frames are denoted with black borders. See how the predictions jump back-and-forth—between static poses on missing frames and proper estimates on valid frames.
Figure 14: Prediction of self-occluded joints: Last 12 consecutive frames from a 30-frame sequence prediction for a 3DPW test sample. Bottom row shows estimated 3D poses overlaid on respective ground truth skeletons. Our model manages to approximately remember the position of the joint before occlusion and also recover its new pose when the occlusion ends.
Figure 15: Successful temporal prediction: Last 12 consecutive frames from a 30-frame temporally consistent sequence prediction for a 3DPW test sample. Bottom row shows estimated 3D poses overlaid on respective ground truth skeletons. Timesteps with missing frames are denoted with black borders. Although not completely accurate, our model manages to exhibit strong motion cues under occlusion and missing frames. Model can recover the motion sequence by building upon the earlier half of the pose sequence, which presents unoccluded lower body (not shown here).
### A.3 Hyperparameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Human3.6M</th>
<th>3DPW</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\lambda_{rot}$</td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td>$\lambda_{3D}$</td>
<td>5.0</td>
<td>5.0</td>
</tr>
<tr>
<td>$\lambda_{2D}$</td>
<td>0.2</td>
<td>0</td>
</tr>
</tbody>
</table>

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Recurrent cell</td>
<td>LSTM</td>
<td>GRU</td>
</tr>
<tr>
<td>Input hidden layer size</td>
<td>256</td>
<td>512</td>
</tr>
<tr>
<td>Recurrent layer size</td>
<td>2048</td>
<td>1024</td>
</tr>
<tr>
<td>Output hidden layer size</td>
<td>1024</td>
<td>1024</td>
</tr>
<tr>
<td>Activation functions</td>
<td>ReLU</td>
<td>ReLU</td>
</tr>
</tbody>
</table>

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Rotation format</td>
<td>rot6d</td>
<td>Rotation Matrix</td>
</tr>
<tr>
<td>Random noise scale</td>
<td>0.01</td>
<td>0.04</td>
</tr>
<tr>
<td>Joint mask ratio</td>
<td>None</td>
<td>0.2</td>
</tr>
<tr>
<td>Confidence threshold</td>
<td>N/A</td>
<td>0.1</td>
</tr>
</tbody>
</table>

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of epochs (std.)</td>
<td>190</td>
<td>10</td>
</tr>
<tr>
<td>Number of epochs (decay)</td>
<td>250</td>
<td>8</td>
</tr>
<tr>
<td>Gradient clipping</td>
<td>0.5</td>
<td>None</td>
</tr>
<tr>
<td>Checkpointing freq. (epochs)</td>
<td>25</td>
<td>1</td>
</tr>
</tbody>
</table>

**Table 10: Model settings and hyperparameters:** Number of epochs (std.) denote the updates where we keep the learning as constant and Number of epochs (decay) denote the updates where we linearly decay the learning rate to zero over time. We do the gradient clipping after the backward pass is done for all layers of the network and the gradients were calculated, not while backpropagating the gradients.
Bibliography


Declaration of originality

The signed declaration of originality is a component of every semester paper, Bachelor's thesis, Master's thesis and any other degree paper undertaken during the course of studies, including the respective electronic versions.

Lecturers may also require a declaration of originality for other written papers compiled for their courses.

I hereby confirm that I am the sole author of the written work here enclosed and that I have compiled it in my own words. Parts excepted are corrections of form and content by the supervisor.

Title of work (in block letters):

LEARNING IN-THE-WILD TEMPORAL 3D POSE ESTIMATION FROM MOCAP DATA

Authored by (in block letters):

For papers written by groups the names of all authors are required.

Name(s): CETIN

First name(s): DORUK

With my signature I confirm that
- I have committed none of the forms of plagiarism described in the 'Citation etiquette' information sheet.
- I have documented all methods, data and processes truthfully.
- I have not manipulated any data.
- I have mentioned all persons who were significant facilitators of the work.

I am aware that the work may be screened electronically for plagiarism.

Place, date
Zurich, 01 June 2020

Signature(s)

For papers written by groups the names of all authors are required. Their signatures collectively guarantee the entire content of the written paper.