THE UNIVERSITY *of* EDINBURGH

Global Human Motion given Monocular Camera Assumptions: From Known, Static to Unknown and Moving

Julian Habekost

Doctor of Philosophy
Institute of Perception, Action and Behaviour
School of Informatics
University of Edinburgh
2024

Abstract

In this doctoral thesis, I present a body of work around estimating 3D global human motion from monocular videos under different camera assumptions by learning from motion capture data. The classical monocular 3D human pose estimation task is only concerned with root-relative poses, here called local poses. Local human poses do not traverse in space and are only of limited use for motion-capture-like applications, e.g. for a character in a game or animated movie. The relationship between local and global human poses is conceptually connected to the camera projection and its position or motion. Chapter 3 proposes a generative model based on adversarial learning that learns the projection of human motion of a known but unseen camera. We are the first to introduce a differentiable egocentrisation in order to embed global human motion into a neural prior. We show that this approach exceeds the performance of other camera domain adaptation methods by comparing them in the local pose space. We are the first to show that the model's knowledge of the ground plane and the projection plane also improves the local 3D pose quality. In chapter 4 we learn a supervised model based on synthetically rendered humans in sequences of arbitrary length. If we can assume that the subject's motion is on an unknown ground plane and we know that the camera is static but unknown, we show that we can infer human motion and even camera intrinsics and extrinsics. In chapter 5 we adapt a generative model based on a conditional variational autoencoder (cVAE) to enable the subject traversing terrain under an unknown moving camera. The moving camera estimation is supported with a classic feature matching visual optometry approach. We are the first to show that a neural model of global motion on terrain can enable and enhance the performance of simple feature matching based visual optometry. We record a large dataset of two subjects moving over obstacles and on the flat ground while being filmed with a handheld camera with different field of views. This allows us to analyse under which circumstances the model performs best, specifically with respect to the estimation of camera intrinsics and motion.

Acknowledgements

Thanks to Facebook Reality Labs for supporting this research. Specific thanks to Takaaki Shiratori for continuous input and useful critique. Thanks to Bob Fisher for co-eyeing my PhD journey and taking over the administrative supervision on the last mile; and last but not least thanks to my professor Taku Komura for all of the above and additionally up-keeping the high ambitions and standards and for steering the direction. Finally, thanks to Robin Julia Trute, who not only helped me mentally through tough years but also with capturing ground truth data in exhausting night shifts; without you I would not have made it.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Julian Habekost*)

Table of Contents

Chapter 1

Introduction

Estimating human motion from videos has applications in augmented reality, virtual reality, computer animated movies, computer games and even sports. The canonical 3D human pose estimation task, facilitated through milestone datasets like Human3.6M (Ionescu et al., 2014) and MPI-INF-3DHP (Mehta et al., 2017a), is only concerned with a local, root-relative poses (e.g. Martinez et al. (2017); Kanazawa et al. (2018); Kocabas et al. (2019); Kanazawa et al. (2019)). A local pose is rotated relative to the camera and is centered to the origin with the root of the kinematic skeleton tree, usually the hip. This means that the orientation of the local pose is a superposition of the the orientation of the camera and the human and regardless of translational motion of the camera or the human, the root of the skeleton never moves translationally.

Global human motion means that the skeleton translates and rotates in space defined by the actual human motion and independent of the camera. Two global human motion sequences are seen as equivalent in context of this work if there is a single fixed rotation, translation and scale (i.e. an affine transformation without skew) that maps one into the other. If global motion instead of local motion is desired, a simple solution is to obtain a calibrated camera and estimate the depth of the local pose by placing it in the camera space while minimizing the reprojection error between 3D pose and 2D detections. This approach has two issues: First, by solving global motion through two independent steps, one implicitly assumes that global motion does not yield exploitable information for local poses. In contrast, we argue that world contacts, or in most cases foot contacts, help stabilizing the local pose and can only be exploited in models that directly estimate global motion. Second, there are a lot of videos in the wild that contain human motion for which we cannot get a camera calibration. Therefore we jointly estimate global motion and camera calibration in chapter 4 and chapter 5.

One issue that our work faces is that the mentioned common benchmark datasets for human pose estimation are not designed to investigate questions about global human motion and the camera. For example the popular Human3.6M (Ionescu et al., 2014) benchmark has plenty of variation in subjects and acting topics but almost none with respect to the cameras. Even worse, the camera setup is the same for training and test split of the dataset. So the bulk of the methods, which rightfully all train on the training set, overfit to this camera setup and hence abstract estimating the camera away by taking it as an implicit input to their systems through aforementioned training material. In this work we are much more explicit with cameras, their estimation and their connection to global human motion. We have to disregard training procedures that would lead to implicit camera overfitting. This gives us a disadvantage when comparing our performance results to existing methods on typical benchmarks. But on the other hand there is a strong argument that our methods are consequentially more generalizable to different camera setups because we only need to adjust our explicit camera assumptions (if at all) instead of recording and retraining a new dataset with this new camera setup. Note that the training material in these benchmark datasets is large and rerecording it is very expensive. Therefore we introduce new benchmark datasets but also still report on the popular Human3.6M (Ionescu et al., 2014) and argue that our results should be seen as a lower bound on what can be achieved without exploiting ground truth cameras.

1.1 Thesis Structure

This thesis is structured into this introduction, a background chapter (chapter 2), three main chapters (chapters 3, 4 and 5) and an overall conclusion (chapter 6). The background chapter details the underlying concepts, e.g. human motion, camera models and neural networks for generative and regression models. The three main chapters focus on three different methods. Their motivations, connections and contrasts are laid out in the following subsections.

1.1.1 Chapter 3: Adversarial Learning for new Cameras

We start in chapter 3 with the setting of a known and calibrated static camera. We invent the differentiable egocentrization to efficiently learn an adversarial motion model and use a differentiable reprojection to make sure the human subject follows the ob-

servations supplied by an off the shelf 2D keypoint estimator. This regression model is learned in a conditional generative-adversarial setting with a motion discriminator and a pose discriminator both looking at a large dataset of feasible human motions. The regression in the form of the adversarial generator needs to satisfy the 2D observations via a differentiable projection and both discriminators' demand for 3D human motion *realness*, the likelihood of the generated motion coming from the real human motion dataset. Our construction hypothesis is that realistic motion that fits the 2d observations is strongly correlated if not unambiguously identical with the actual subject's motion recorded in the video.

Our experimental demonstrations are based on the well established Human3.6M dataset (Ionescu et al., 2014). The publicly available subset of the dataset features seven of eleven actors (called subjects). The subjects all independently act the same 17 scenarios (e.g. eating, talking on the telephone, walking a dog) while being recorded with four calibrated and synchronised cameras from four corners of the room. The room is also set up with a marker based motion capture system, synchronised alongside the video cameras, and the subjects wear generic outfits with markers taped to them. This allows to match the four video recordings with a 3D skeleton or 3D joint keypoints in world space as a ground truth. We choose one camera and its known intrinsics and extrinsics to learn the 3D global motion of all subjects available to us. Our method assumes 2D observations from that camera and lifts them into 3D global motion by learning a unified but specific model across all subjects acting in the identical scene and recorded with the identical camera.

Since state-of-the-art methods only estimate local human poses trained with Human3.6M 3D human pose ground truth (Kanazawa et al., 2018, 2019; Martinez et al., 2017), we can only compare to these methods and show that we come reasonably close in local human pose accuracy. We achieve this without using Human3.6M 3D human pose ground truth, which makes our approach much more versatile. With our approach we can learn from videos without the need of a motion capture system to capture specific training 3D human motion or pose ground truth for the target camera and scene setup. We only use ground truth for evaluation. Instead we can show that our system learns the motion from the generic human motion dataset and correctly places it in the scene and behind the camera.

We also show that when no 3D ground truth is available, directly estimating local poses with a generative-adversarial approach yields worse local pose accuracy than our generative-adversarial approach of estimating global motion. We can also visually see

that without the learned global motion the model struggles to keep the individual poses aligned with implicit reference points, like foot contacts with the floor and body contacts with chairs (see hypothesis **H1**). Note that while the floor ground plane is given through the camera calibration, foot contacts and chair contacts are implicit features learnt only from global human motion. That this works to a reasonable degree can be seen qualitatively and in our ablation experiments.

1.1.2 Chapter 4: Synthesizing Simple Sequences

In chapter 4 we assume a static camera with unknown calibration. This means we know that the camera stays at its unknown position and orientation for the entire sequence and we also do not know the focal length (i.e. field-of-view).

The motivation is that it is easy to assess whether a video recording of human motion has been filmed with a static camera and we want to be able to exploit this knowledge even when there is no calibration available. The method in chapter 3 needs a more controlled environment and more video data. Even if we are only interested in one subject acting one sequence we need the video recordings of multiple subjects acting multiple different scenarios each in the same scene. In contrast the method in chapter 4 is a one-shot regression, which can handle individual sequences of single subjects while exploiting the unknown static camera.

In this setting there are two issues inherent to the problem to overcome: the scale ambiguity and the field-of-view ambiguity. The scale of objects (or subjects) in a monocular video cannot be estimated without a given reference. Note that in chapter 3 this reference is given through the known camera position: the resulting distance between the ground plane and the camera is effectively a scaling reference.

We create a synthetic dataset with rendered videos of human motion. We generate random body shapes, motion samples, camera field-of-views, orientations and positions to learn a regressor that jointly estimates human motion and the camera field-of-view, position and orientation.

We solve the scale ambiguity with our height-normalized SMPL (HN-SMPL). This is the SMPL (Loper et al., 2015) model but with the new latent space projected into a SMPL latent subspace that only ever represents a constant body height. This constant body height is arbitrary since we cannot solve for it and the real height does not give useful information beyond the scale. If we optionally know the subject's height, then we can simply multiply the result by the quotient of known height and assumed height.

Hence we choose the average of the body height in the original SMPL model simply because we expect the best support around its mode.

We show that accounting for the field-of-view ambiguity can be trained into the neural network regression with our approach. The intuition is here that a person walking towards the camera will have a growth rate in the 2D image projection plane depending on the field-of-view of the camera and the walking speed. The model implicitly learns to estimate the walking speed by foot contacts, human posture and walking characteristics and can attribute the rest of the 2D growth and shrinking differential to the field-of-view.

Note that the estimated field-of-view, camera position and camera orientation are not used in the model as an optimization target or any other input to the human motion estimation problem. The estimated camera's characteristics are just the neural network's regression result alongside the human motion estimate. The main claim is here, that a model that aims at estimating human motion within the laid-out camera setting needs to be able to estimate camera field-of-view, orientation and position to optimally leverage the setting's assumptions.

Additionally to a Human3.6M evaluation like in chapter 3, we show experimental evidence specifically for the claim of estimating the camera field-of-view, orientation and position by recording our own video test dataset. We record videos of different subjects with different camera field-of-views, orientations and positions to show our method's ability to estimate said randomized camera characteristics alongside human motion. The human pose estimation performance is comparable to methods that need more data or are supervisedly trained on the target dataset while our method is not. Other aspects can only be estimated by our method, which defies a comparison.

1.1.3 Chapter 5: A cVAE for Terrain and Visual Odometry

In chapter 5 we extend our task and estimate global human motion with a handheld moving camera with unknown focal length both on flat and non-flat terrain with obstacles. For this we capture a new dataset with two subjects traversing both flat and obstacle terrain and recorded with a handheld camera with varying focal length modes. Both the subject's human motion and the handheld camera are tracked with a marker based motion capture system giving a ground truth for the evaluation of our approach.

If the camera is moving, the movement of a projected human can be explained alone through the superposition of human global motion of the subject and the motion

of the camera (hypothesis **H2**). A conditional generative cVAE model of valid human motion that is optimized in test time to target 3D and 2D body features from off-the-shelf detectors helps us to disentangle and differentiate between both by simply exploiting this and attributing all residual human motion in the projected image space to the camera. Note that including obstacles in the task setup makes this harder because the space of valid motion is not constrained to the ground plane.

We mitigate some of the conceptual difficulty of this task by combining a test time optimization approach based on a cVAE human model with classic camera pose estimation via feature matching and RANSAC. We invent a new approach to combine the gradient based cVAE optimisation with the random sampling consensus based method. We include both camera path solutions crosswise in each other's optimization terms with iteratively increasing biases to enforce converging to a joint solution. We investigate different bias scheduling schemes in context of the iterative process and show which work well to arrive at a combined camera path estimate that beats each method's separate individual calculations. Note that this means that we also improve global human motion quality because human and camera motion are interlocked as discussed above and shown in the evaluation (hypothesis **H1**).

We also improve the stability of the height traversal by addressing drift in the height of the human motion by predicting whether sequences of footsteps are made on the same ground plane to then pull the heights of all those footsteps towards a common ground plane height level in the cVAE test time optimization. We demonstrate the improvement of human motion estimation performance in an ablation study for this so-called *step plane estimation*.

We show that our approach outperforms a state-of-the-art approach for flat terrain motion and a handheld camera. A state-of-the-art approach that handles motion with obstacles and a handheld camera is not available for comparison.

1.2 Hypotheses

The main hypotheses of this work are:

H1 Estimating global motion improves the local pose quality by exploiting implicit or explicit knowledge about world contacts.

H2 A human's global motion can act as an implicit camera calibration reference by jointly estimating the global motion and the camera's intrinsics and extrinsics.

We show evidence for **H1** in all three chapters. Evidence for **H2** is presented in chapter 4 and chapter 5.

1.3 Contributions

The contributions of this thesis can be summarized as following:

- Two novel, from ground up original methods (chapter 3, 4) that beat or equalize state of the art competing methods in fair comparisons while delivering evidence for **H1** and **H2**.
- One novel, heavily extended and adapted method (chapter 5) that beats a state of the art competing method in a fair comparison while delivering evidence for **H1** and **H2**.
- Many novel sub-modules, like differentiable egocentrization (section 3.3.3), HN-SMPL (section 4.3.1), iterative refinement via bias scheduling (section 5.5.2) and step plane estimation (section 5.4), that are either functional requirements or are validated in an ablation study.
- Two newly recorded datasets (section 4.5.1, 5.3) for evaluating and validating our methods.

See the individual chapters for a more detailed discussion on the chapter's contributions.

Chapter 2

Background

This chapter introduces the concepts necessary to understand the thesis and which the methods that were used are based on.

2.1 Machine Learning

Machine Learning is the discipline of leveraging data to model functions instead of solely handcrafting them. A set of parameters is calculated to find the function in a parameterized family of functions that explains the observed data best with a given criteria. These parameters are called the model weights. The function family, fit criteria and parameter search method are the main subjects of variability and research. One of the simplest example for this is a linear regression where the slope and the intercept are the weights, which can be calculated with a closed form solution. See Bishop (2006) for further details on classic machine learning methods. Also Bishop (2006) is used for reference in this chapter.

In the most general setting the data consists of input features x and and labels y. They are assumed to be sampled from a distribution $p_{data}(x,y)$ (or sometimes expressed as $x,y \sim p_{data}$) that we can not calculate. If it is easy to observe x but hard or expensive yet desirable to observe y it is a good idea to collect N samples $x_i, y_i, 0 < i \leq N$, the so-called ground-truth and learn a model f_W with weights W to predict y from x given a loss function L:

$$\hat{y} = f_W(x) \tag{2.1}$$

$$\text{e.g. } L(y,\hat{y}) = \|y-\hat{y}\|^2 \tag{2.2}$$

$$J_{data} = \mathbb{E}_{p_{data}}(L(y, f_W(x))) \tag{2.3}$$

$$J_{data} \simeq \sum_{\{x_i,y_i\}_{test}}^{N_{test}} \frac{L(y_i, f_W(x_i))}{N_{test}} \tag{2.4}$$

This example loss function here is a squared distance between the predicted and the ground truth feature vector label. Note that we are actually interested in hypothetically minimizing J_{data}, the true expectancy of L under the true distribution p_{data}. As p_{data} is unknown, this is impossible, instead we can only help us by using a surrogate, a so called test subset. The test subset $\{x_i,y_i\}_{test}$ is not used for training (i.e. finding the model parameters) but only for evaluation. The assumption is that it is a good stand-in for the true distribution of the data. In practice, a big part of applied machine learning research is often about arguing whether a test dataset fulfills equation 2.4 well enough and about proposing better, larger or more diverse test sets.

2.1.1 Neural Networks

Neural network is a metaphorical name for class of function families that have huge parameter spaces (in most cases a lot of affine transformations), that are structured in layers and that are optimized with stochastic gradient descent (SGD) or a related method (Schmidhuber, 2015). Each layer is an established or proposed family of functions. Different layers can be arranged freely to a neural network as long as the dimensionalities of output and input of consecutive layers are the same.

The neural network is the resulting function graph. After random initialization its weights are optimized iteratively in the so-called training loop. For each iteration the network is evaluated for a random subset of data point inputs at its current weights. The result is contrasted to the corresponding set of data labels through a loss function. The loss function is minimized by moving all of the network's weights a small step in the direction opposite of the loss's gradient for each random batch of data points at once. This is repeated at most until the loss converges.

This so-called error gradient is calculated through back-propagation (Rumelhart et al., 1986). In back-propagation the gradient of a parent vectored value in a function graph is calculated through multiplying the child's gradient with the Jacobian of the

operation between them. We start at the end of the function graph, the scalar loss, and go backwards until we reach all weights. Anytime a node is reached multiple times the gradients are added. This back-propagation procedure follows directly from the chain rule of derivation. See Goodfellow et al. (2016) for a more in-depth explanation.

An important aspect of back-propagation is that it applies to any sub-function in a function graph, parameterized or not. This not only makes it easy to arrange common layers into new neural network architectures – it also makes it easy to construct a new layer. In modern neural network libraries most atomic mathematical functions come with predefined backpropagation if possible. This is why e.g. in chapter 3 we can simply construct a new camera projection layer and use it in a neural network without analytically deriving any derivatives. It is enough to know that each atomic mathematical operation necessary for the projection is differentiable and is fully implemented with back-propagation in the neural network library. This well supported abstraction level is also why this author has not seen any applied machine learning researcher implement any back-propagation or function derivatives on their own.

As a big caveat to this, modern neural network libraries are based on floating-point arithmetic with fixed precision. Very large or very small gradients in combination with floating point errors can let the gradient *explode* to NaN or *vanish* to zero. The first makes the network unable to calculate anything and the latter makes it stop learning anything. Hence it is still useful to have an understanding of the individual math operation's derivative. Sometimes it is necessary to replace a combination of functions that have large and small sub-gradients with a mathematical equivalent combination with more moderate sub-gradients, less likely to explode or vanish. This was necessary for the egocentrization layer in chapter 3.

2.1.2 Feed Forward

A feed forward network (FFN), also called multi-layer perception, is a concatenation of K feed forward or linear layers, with $K \geq 2$. A linear layer is an affine transformation and, if the linear layer is a hidden layer, followed by a so-called activation function. All layers except the final output layer are hidden layers and activation functions are element-wise non-linear functions. A classic activation function is the sigmoid function, but more recently the piece-wise linear ReLU is used more widely: ReLU is a function that is constant zero for values below and equal to zero and the identity for everything above zero.

In the simplest case of $K = 2$ the feed forward neural network consists of two affine transformations with an element-wise non-linearity in-between them. The result of a hidden layer is called hidden space, latent space or latent code. The simplest case has a single hidden space.

A formal definition can be given as

$$ReLU(\begin{bmatrix} x_0 \\ x_1 \\ \dots \end{bmatrix}) \coloneqq \begin{bmatrix} y_0 \\ y_1 \\ \dots \end{bmatrix}, \text{ with } y_i \coloneqq \begin{cases} 0, & x_i \leq 0 \\ x_i, & x_i > 0 \end{cases} \tag{2.5}$$

$$h_k(x; W_k, \dots W_1) \coloneqq \begin{cases} ReLU(W_1 \cdot x), & k = 1 \\ ReLU(W_k \cdot h_{k-1}(x; W_{k-1}, \dots W_1)), & k > 1 \end{cases} \tag{2.6}$$

$$f(x; W_K, \dots W_1) \coloneqq W_K \cdot h_{K-1}(x; W_{K-1}, \dots, W_1) \tag{2.7}$$

$$x \in \mathbb{R}^{M_0} \tag{2.8}$$

$$W_k \in \mathbb{R}^{M_k \times M_{k-1}} \tag{2.9}$$

$$h_k(x; W_k, \dots, W_1) \in \mathbb{R}^{M_k} \tag{2.10}$$

$$f(x; W_K, \dots, W_1) \in \mathbb{R}^{M_K} \tag{2.11}$$

The input vector x has an input dimensionality of M_0. The kth hidden layer is a vector function h_k with dimensionality M_k, called the number of hidden units (or hidden dimensionality, latent dimensionality), and an associated weight matrix W_k. The last layer's output f is a vector function with M_K dimensionality (the network's output dimensionality), and an associated weight matrix W_K. It follows that W_k is an $M_k \times M_{k-1}$-matrix, which accounts for the fact that a layer's output dimensionality needs to equal the following layer's input dimensionality.

In theory a wide but shallow feed forward network (i.e. one with a huge latent space and only few or just one hidden layer) is a function approximator with the same expressiveness as a narrow, deep feed forward network. But experimental evidence shows that training deeper networks is more effective, at least with the training methods currently available.

2.1.3 Temporal Neural Networks

Human motion is a temporal sequence (also called time series data), both in 2D and in 3D. A feature vectors x_i can be further split into temporal steps that can be indexed: $x_{i,t}$, usually just x_t. The same applies to the labels. There are two main concepts to differentiate here: if y_t is predicted for a specific t by only looking at $x_0, ..., x_t$, i.e. the past til current temporal steps of the input feature x, it is called filtering, online, causal or unidirectional filtering. This is useful for real-time applications, where one can not see into the future. This work is concerned with offline methods, where you have the whole sequence available. We simply predict the whole of y from x or y_t from x, which is equivalent because you can always sweep over all y_t's. This is called sequence to sequence learning, sometimes bidirectional sequence to sequence or non-causal sequence to sequence learning.

Feed forward networks are very capable and in theory it would be possible to train a temporal model simply by feeding concatenated temporal feature vectors (i.e. all of x) to a feed forward network. But this would mean that the network has to learn the temporal nature of the data based on pattern in the data. Usually we want the network to learn pattern from the data that we do not already know and also cannot already model well by hand. Instead we can incorporate the temporal assumption into the network architecture, which has also been shown experimentally to be much more effective (Hochreiter and Schmidhuber, 1997; van den Oord et al., 2016).

A simple approach is an auto-regressive feed forward network that is originally made for a simple look-ahead prediction task, i.e. to predict x_{t+1} from x_t (there is no y here in this specific look-ahead prediction setting):

$$\hat{x_{t+1}} = f_W(x_t) \tag{2.12}$$

$$\hat{x_{t+3}} = f_W(f_W(f_W(x_t))) \tag{2.13}$$

As seen in equation 2.13 the model can be applied multiple times to get a prediction further into the future at the cost of certainty. We can also adjust this slightly so it can be used for causal filtering with a very short past window, e.g. predicting y_t from x_t and x_{t-1}. It is not strictly auto-regressive, lets call it a pseudo-auto-regressive approach. In chapter 5 we use a pseudo-auto-regressive approach even for a bidirectional model. This is possible in the specific setting of optimizing within the latent space of a VAE, also see subsection 2.1.5.

A convolutional neural network (Lecun et al., 1998) is a neural network in which at least for some layers the learned parameters are convolutional filters instead of matrices of affine transformations (Goodfellow et al., 2016). They are very popular in image based tasks with 2D filters but they can also be applied to time series tasks with 1D filters. There are popular papers (e.g. Pandey and Wang (2019); Lea et al. (2017); van den Oord et al. (2016)) which specifically look at convolutional neural networks in the context of time series tasks and propose specific architectures. We do not work with their proposed architectures but we instead adopt the U-Net (Ronneberger et al., 2015a) 2D convolutional neural network that is renowned for image segmentation tasks to 1D and use it throughout this work. This architecture is explained in more detail in subsection 3.3.2. Note that there are many variables in designing neural network architectures that are first of all experimentally validated. Conceptual arguments for choosing a temporal U-Net are the relative low architectural complexity through high repetitiveness, the resulting flexibility, and the relative wide receptive field without requiring too much network layer depth. The receptive field is the distance between the first and last temporal step of the input that the model can look at to infer an output step. The receptive field increases with pooling operations, stride sizes and layer depth.

Auto-regressive and pseudo-auto-regressive models are useful if the temporal coherence is strong but with zero to none correlation between temporally distant steps. Temporal convolutional neural networks relax this a bit but are still bound to a local temporal receptive field. But there are tasks where it is required to correlate information from distant steps, e.g. in complex language models that aim at understanding long texts. It is also necessary in chapter 5, where evidence for information about the camera can appear in one specific moment but is relevant for the entire sequence. LSTM's were a solution to this introduced by Hochreiter and Schmidhuber (1997). Among other design decisions, they integrate an attention mechanism that can correlate features with an arbitrary temporal distance within a set limit. Later Vaswani et al. (2017) figured out that *attention is all you need* to build large temporal models with distant temporal correlation, which they call transformers. We tried attention or transformer based architectures in chapter 5 but achieved equal or better performance by applying global maxpooling along the temporal domain, a much simpler operation than attention. Global maxpooling applies element-wise maximum along the feature dimensions over hidden features and as a result each step sees the element-wise maximum of all steps for some hidden space. This approach has also been used for global aggregation in the image and point cloud (Qi et al., 2016) domain.

2.1.4 Generative Adversarial Networks (GANs)

Given a random variable x that follows an unknown distribution p_{data}, i.e. $x \sim p_{data}$, generative modeling is the task of modeling the distribution as $\hat{x} = G(z)$ such that we approximate sampling from p_{data} by transforming a random variable z based on a distribution that we can easily sample from, usually a normal distribution $z \sim \mathcal{N}$. A classic example is the distribution of pictures of human faces.

Goodfellow et al. (2014) propose a two player minimax game between a generator G and a discriminator D with the following value function:

$$\hat{x} = G(z) \tag{2.14}$$

$$\min_G \max_D V(D,G) = \mathbb{E}_{x \sim p_{data}}[log(D(x))] + \mathbb{E}_{\hat{x} \sim p_{\hat{x}}}[log(1 - D(\hat{x}))] \tag{2.15}$$

$$\implies \min_G \max_D V(D,G) = \mathbb{E}_{x \sim p_{data}}[log(D(x))] + \mathbb{E}_{z \sim \mathcal{N}}[log(1 - D(G(z)))] \tag{2.16}$$

The generator G is a neural network that transforms samples from a normal distribution ($z \sim \mathcal{N}$) into samples from a distribution $p_{\hat{x}}$ that tries to mimic p_{data}. The discriminator is a neural network D that sees both samples from p_{data} and $p_{\hat{x}}$. Its job is to learn whether a sample comes from the real or fake distribution. Hence, $D(x)$ represents the probability of $x \sim p_{data}$ and the counter-probability $1 - D(x)$ is associated to $x \sim p_{\hat{x}}$. The value function V is alternatingly maximized with stochastic gradient ascend on the weights of D with a fixed G and then minimized with a stochastic gradient descend on the weights of G with a fixed D.

An analogy is a minimax game between money counterfeiters and manufacturers of banknote validator devices. With better banknote validators, the counterfeiters need to produce money that is closer to resembling real money to again fool the devices. Then in turn the manufacturer improves their devices, and so on. In a world with an absent criminal justice system it is imaginable that we would end up with money counterfeiters that have learned to produce fake banknotes that are indistinguishable from real ones and with a validator device that has been perfected for detecting any deviation yet useless because there is none. This is the goal for our generator and discriminator minimax game, because it means our generator can mimic the data distribution perfectly.

This game is very delicate and fragile. If the validator manufacturer's resources are capped we will end up with an inferior device and lazy counterfeiters that fool the device with bad counterfeits. The same will be true if the generator is stronger than the

discriminator, i.e. in terms of number of weights and layer depth. Note that there is no measure of the counterfeits' quality because otherwise we could directly optimize this measure. On the other hand if the device is perfect or even too strong from the start the counterfeiters are forced to make a too large quality jump in a single iteration and will run out of money or give up. The analogy here is less obvious but it still holds: The generator improves its quality by ascending its gradient through the gradient of the discriminator per chain rule. If the discriminator perfectly identifies both samples from G and also the vast space around samples from G as equally fake the generator will be lost because the discriminator's gradient will be close to zero for G's current samples, which will also make the generator's gradient close to zero or zero.

While this can be addressed through weeks of relentless fine-tuning, Arjovsky et al. (2017) introduce the greedy solution of constraining the norm of the gradient of D. They first simplify the value function to a combination of simple linear scoring functions:

$$\min_G \max_D W(D,G) = \mathbb{E}_{x\sim p_{data}}[D(x)] + \mathbb{E}_{z\sim\mathcal{N}}[-D(G(z))] \tag{2.17}$$

Originally in V we optimize log-probabilities. The simple, unbounded linear scores here in W can be monotonically mapped onto the log-probabilities up to undefined areas. These also do not change the nature of the minimax game, the biggest cost of this simplification is interpretability. They prove and empirically show that the upside is that, in this setting, restricting the discriminator's gradient's norm to one works well to regulate the discriminator's strength. In follow-up work Gulrajani et al. (2017) introduce the gradient penalty R_{pen}, a regularization scheme that is applied alongside maximizing W:

$$\tilde{x}(x,\hat{x}) = ax + (1-a)\hat{x}, \quad a \sim U(0,1) \tag{2.18}$$

$$R_{pen}(D,x,\hat{x}) = (\|\nabla_{\tilde{x}} D(\tilde{x}(x,\hat{x}))\|_2 - 1)^2 \tag{2.19}$$

$$\text{(for the max step:)} \ \max_D [W(D,G) - \lambda_{pen} R_{pen}(D,x,\hat{x})] \tag{2.20}$$

They regularize the discriminator's gradient's norm to be one on the line between real and fake examples. They sample from that line by construction of $\tilde{x}(x,\hat{x})$, the uniformly random linear interpolation between a real sample x and a fake one $\hat{x}$. The gradient $\nabla_{\tilde{x}} D(\tilde{x})$ is regularized to an L2-norm of one with a squared error weighted by a regularization coefficient λ_{pen}. This exactly addresses the previously described issue

that a too strong discriminator causes: a flat discriminator gradient as per chain rule resulting in a flat generator gradient. Now we can simply choose a strong discriminator network design and restrict its strength adequately with the gradient penalty.

The framework described until now aims at modeling a distribution without the possibility to condition it (except for changing the underlying dataset). But it can also be adapted to conditional distributions, sometimes the framework is then called cGAN (Mirza and Osindero, 2014). To achieve conditionality one can trivially give the known ground truth condition y as an input to both the Generator and the Discriminator. The latter sees the known ground truth condition both for real as well as for fake samples. This helps the generator to approximate $p_{data}(x|y)$. An example for a condition could be the living space size of a building associated to its picture. While a given living space restricts the distribution of buildings that could possibly be associated, it does not unambiguously define the building. If we have an unambiguously defining relationship between a condition and its dependent variable and a ground truth dataset, we can simply go back to non-generative regression learning. But a regression with only typical L2 or L1 losses can only learn to predict some linear interpolation (e.g. the mean) of a dependent distribution, which might not even be part of that dependent distribution. I.e. the mean of pictures of 3000 square-feet buildings is just a grey mash. We use this to ensure that our generated human motion lies on the metaphorical manifold of feasible motion.

2.1.5 Variational Autoencoders (VAEs)

Another method of learning a distribution through generative modeling is possible with variational autoencoders (VAEs) (Kingma and Welling, 2014). VAEs extend the general concepts of autoencoders. A general autoencoder consists of two functions, the encoder f and the decoder g (following the notation in Goodfellow et al. (2016)):

$$h = f(x) \tag{2.21}$$

$$r = g(h) \tag{2.22}$$

$$\text{e.g. } L(x) = \|x - g(f(x))\|_2^2 \tag{2.23}$$

$$x, r \in \mathbb{R}^N \implies h \in \mathbb{R}^{M<N} \tag{2.24}$$

The core principle of the autoencoder is that its output is learned to equal its input, for example with an L2 loss between the input x and output r, also called the

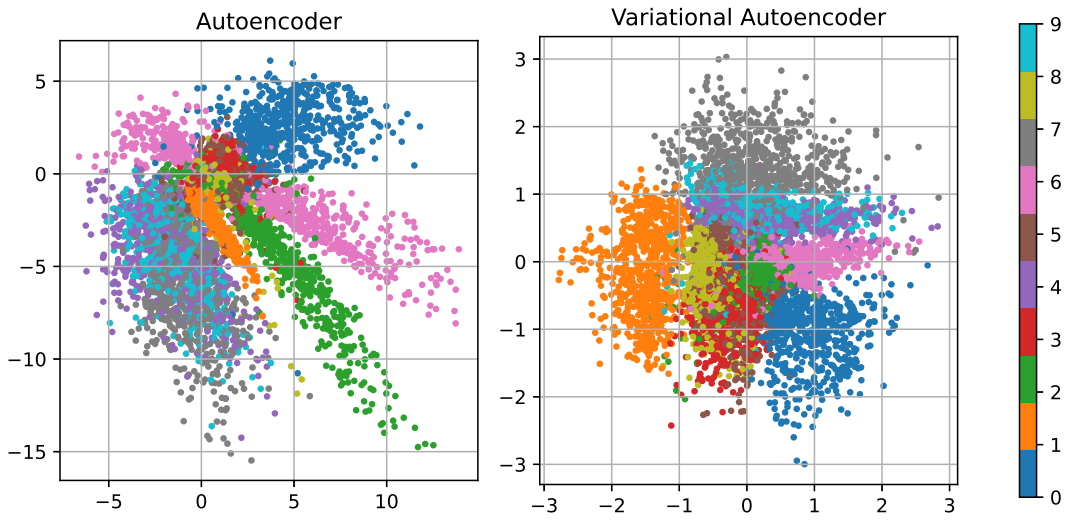


Figure 2.1: Comparison of the $\mathbb{R}^2$ latent spaces of a vanilla autoencoder and a variational one on the MNIST handwritten digits dataset (Lecun et al., 1998). The dataset consists of thousands handwritten digits, each isolated and scaled to a 28x28px grayscale image. Note that the label is not an input to the autoencoders, they are only an input to this visualization. Each point represents a ground truth MNIST digit image encoded into the two dimensional latent space, and color coded by its label. The clusters of the variational autoencoder are more coherent and the latent space as a whole is more densely packed. It follows a normal distribution more closely, which is the prior used in this VAE. This makes the VAE's latent space more useful for sampling and optimization, because as long as we are somewhat close to 0 in the latent space, we can expect reasonable sample quality. Note that where this latent space visualization is empty, we do not expect the decoder to produce good quality digit images. This makes the larger gaps in the vanilla autoencoder concerning for sampling and optimization.

reconstruction. In-between is a lower dimensional hidden or latent space or code h. The encoder is forced to learn to map the input into the hidden space. The decoder then transforms the latent space back into the original space, which gives the reconstruction. The idea is that the hidden space will capture a lower dimensional variable that determines the input as good-as-possible. If f and g are simple linear transformations, the autoencoder has been shown to be equivalent to a PCA (Goodfellow et al., 2016), hence this framework is much more interesting if f and g are multi layer neural networks.

The variational autoencoder framework introduces Bayesian and variational inference to autoencoders. Now the encoder and decoder are expressed as conditional

probability densities $q_\phi(z|x)$ and $p_\theta(x|z)$, with ϕ and θ being the underlying neural network weights. $p(z)$ is the prior on the latent space, for which the most common choice is gaussian, which we also exclusively use. In practice that means the decoder is constructed with a neural network to predict an M-dimensional mean and an M-dimensional standard deviation of an M-dimensional uncorrelated gaussian in order to span an M-dimensional latent space:

$$\begin{bmatrix}\mu(x)\\ \sigma(x)\end{bmatrix} = f_\phi(x) \tag{2.25}$$

$$q_\phi(z|x) = \mathcal{N}(\mu(x), diag(\sigma(x))) \tag{2.26}$$

$$\mu(x), \sigma(x), z \in \mathbb{R}^M \implies f_\phi(x) \in \mathbb{R}^{2M} \tag{2.27}$$

The gaussian prior on the latent space z is enforced by optimizing the evidence lower bound, the *ELBO*, via a Kullback–Leibler criterion. Please see Kingma and Welling (2019) for an in-depth tutorial on Bayesian statistics, variational inference, and how to derive a VAE loss. For the common gaussian prior case this results in the addition of the following regularization term R_{KL} to the VAE's loss:

$$L_{VAE}(x) = L_R(x) + \lambda_{KL} \cdot R_{KL}(x) \tag{2.28}$$

$$R_{KL} = \sum_{i=1}^{M} (\mu_i^2(x) + \sigma_i^2(x) - log(\sigma_i^2(x))) \tag{2.29}$$

Here L_R is the autoencoder's reconstruction loss from above. The regularization coefficient λ_{KL} trade-offs reconstruction quality versus adherence of the latent space to the prior. The effect of the prior on the latent space can be seen in figure 2.1. The main advantage of a VAE that we can rely on the latent space to produce good quality samples if we are sampling according to the prior. We also gain a quality metric for samples: the likelihood of its latent code vector under the prior distribution. In the case of an uncorrelated gaussian prior this quality metric can easily be used in an optimization criterion, since maximizing the likelihood simply means moving the latent code towards the mean. We use this to build a closed loop optimization for human motion. In this scheme a VAE gives an optimization criterion that constrains the motion to feasible motion.

(a) MPII (b) COCO (c) COCO with feet

Figure 2.2: Landmarks of popular datasets MPII-HP (Andriluka et al., 2014) and COCO (Lin et al., 2014). Note that the connecting lines are only for visual reference, they are simply drawn by connecting the joint landmarks. Pictures are taken from Cao et al. (2018).

2.2 2D Human Pose Estimation

2D human pose estimation is a landmark detection task. An image of a person is associated to a set of semantically defined landmarks, e.g. the right and left elbow, right and left knee, nose, eye, etc. Each landmark has its own $\mathbb{R}^2$ pixel coordinate on the given image. If there are multiple people on the same image the landmark sets are coherent so that they can be associated uniquely with a person. Two well known datasets are MPII-HP (Andriluka et al., 2014) and COCO (Lin et al., 2014). It can be observed in figure 2.2 that the landmark definitions differ slightly between both datasets, both in terms of numbers and also slightly in semantic positioning.

2.2.1 Open Pose

One notable work predating OpenPose is the stacked hourglass model Newell et al. (2016). As one of the first deep learning based approaches to 2D human pose landmark detection, they represent the landmarks as small gaussian heat maps and apply a scheme of spatial compression by down-sampling the latent space. OpenPose (Cao et al., 2018) improves upon this in the following ways: First and most notably, they develop the body part affinity field, a vector field that corresponds to the joint hierarchy within a body and opens up the possibility of multi person estimation. Predicting and

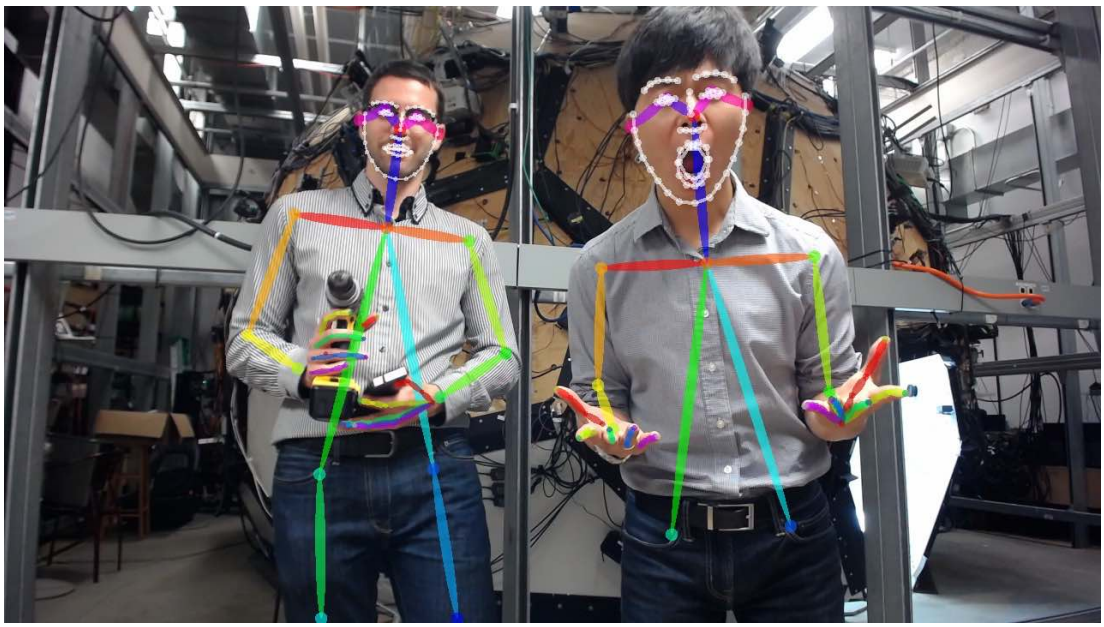
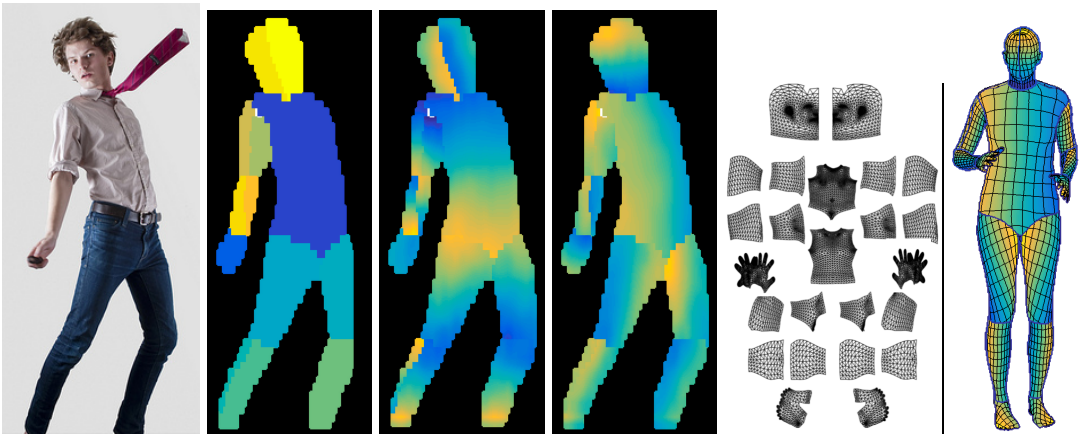


Figure 2.3: This is an almost iconic example result headlining OpenPose's GitHub and paper (Cao et al., 2018), featuring the authors in their CMU lab.

subsequently following the part affinity vector field's direction from one landmark to the next one, body parts that belong to the same person can be linked together which allows to differentiate different people on the same image. Second, multiple datasets even with slightly different semantic keypoint definitions for similar keypoints, but also simply datasets expanding the keypoints to feet or facial landmarks, are unified under the same model. And third, a confidence is calculated for each predicted keypoint.

2.2.2 Dense Pose

DensePose is the name of both a task and the method proposed by Riza Alp Guler (2018). It is a dense superset of the landmarks from 2D human pose estimation. In contrast to just a few landmarks, DensePose associates every pixel that is part of a person with a 3D template mesh. Note that this is still a 2D task, there is no depth estimation involved. They use the SMPL (Loper et al., 2015) model's mesh as the human template mesh. The dense map of a person is expressed as a three channel *IUV* image. I stands for the part index of the 3D body template mesh. They split the template mesh into different parts with each having its individual UV-space. Hence I is an integer $\mathbb{N}^{H\times W}$ matrix and U and V stand for the $\mathbb{R}^{H\times W\times 2}$ UV texture coordinates on the part with index I. Figure 2.4 shows IUV maps on an example image. We use



(a) Image (b) I-Map (c) U-Map (d) V-Map (e) Mesh Parts (f) Mesh

Figure 2.4: DensePose estimates a dense, three channel IUV map (b, c, d). The index I identifies a part (e) of the body mesh (f) for each pixel on the image. The UV coordinates link the pixel to texture coordinates on the body mesh part.

DensePose as an off-the-shelf detector for 2D human features to subsequently regress 3D human motion.

2.3 3D Human Motion

3D human motion (or 3D human motion sequences, or simply 3D human poses) is a time series of 3D human poses. 3D human poses can be expressed via keypoints or via a skeleton with bone orientations. 3D keypoints are simply the 3D coordinates of semantically defined 3D landmarks or joints, e.g. of the knees, elbows, wrists, etc. Just as for the 2D task, these definitions vary in numbers and semantic details, specifically for inner-torso joints.

A skeleton or kinematic skeleton is a tree of bones connected via joints, or equivalently a tree of joints connected via bones, that roughly resembles a human skeleton. Each joint is assigned an orientation that is also applied to all of its children down the kinematic tree. The concept of a bone hereby stands for the fixed distance between two joints for any given skeleton. This kinematic skeleton can then be used to drive a mesh, e,g, with a technique called dual quaternion skinning Kavan et al. (2007), but this is outside the scope of this work. Note that the skeleton constraint can be violated, specifically when representing a human pose only through a number of independent 3D joint keypoints.

Figure 2.5: Subjects and Poses in Human3.6M (Ionescu et al., 2014). Note that these are crops around the actor from one of the four static cameras. Picture is taken from Ionescu et al. (2014).

Local poses, local motion, local 3D human poses, local 3D human motion are all terms in this work to describe sequences of 3D human poses that do not have a translation associated to them. Their skeleton or root joint, usually the hip, is anchored at the origin during the sequence. In contrast global poses, global motion, global 3D human poses and global 3D human motion are terms used to describe sequences of 3D human poses, represented as skeletal bone orientations or keypoint coordinates, with a translation and orientation to an arbitrary origin at the ground plan.

2.3.1 Motion Capture and Human3.6M

Motion capture, in our context short for human motion capture, is any sensor-based method for obtaining or calculating human motion sequences to mirror a human actor's real world motion. The actor usually acts specifically for the purpose of capturing the motion. Some systems place sensors like inertial measurement units on the body of the acting subject and other use video cameras as sensors to observe the actor with or without a set of markers on their body.

The Human3.6M (Ionescu et al., 2014) dataset synchronizes motion capture with RGB video sequences. They set up a Vicon motion capture system in a room, the now iconic room with the red carpet (see figure 2.5). Vicon offers a marker based motion capture system with infrared reflector markers and infrared cameras that also have infrared emitting diodes around them. The markers are usually velcroed onto a

full body black motion capture suit. This is avoided for Human3.6M by instead using casual clothing and double sided tape for sticking the markers to the subjects. This makes the subject's appearance on the video more natural and realistic. The markers are masked out with IR filters, re-identified and tracked across different cameras and lastly triangulated into 3D marker coordinates. Based on a proprietary heuristic, the markers are semantically identified within the context of the human body. For this the marker setup on the actor's body has to follow a predefined semantic marker setup. Then a kinematic skeleton is fitted into the 3d marker coordinates of the actor, accounting for size, symmetry and bone consistency. The fitting procedure and skeleton design are specific and proprietary to Vicon. Note that we use the same Vicon system in our dataset in chapter 5.

Additionally, four static RGB cameras are video-recording the subjects. The four cameras have similar intrinsics (identical up to manufacturing variance) and are placed in the four corners of the room. The system records motion and videos at 50hz. There are eleven actors, of which seven are publicly available. They independently from each other all act the following sequences: *directions*, *discussion*, *eating*, *activities while seated*, *greeting*, *taking photo*, *posing*, *making purchases*, *smoking*, *waiting*, *walking*, *sitting on chair*, *talking on the phone*, *walking dog*, *walking together*.

The proposed validation and test metric, mean per joint position error (MPJPE, typically in mm), and its variants are not skeleton based but keypoint based. The ground truth 3D keypoints are the joints connecting the Vicon skeleton. They are treated as an ordered set x and one-to-one compared with the ordered set $\hat{x}$ of proposed or estimated 3D keypoint coordinates per frame:

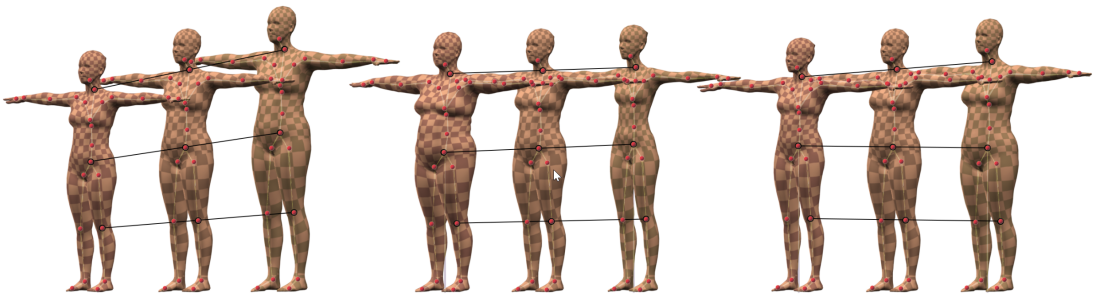
$$E^{\text{test}}_{\text{MPJPE}}(x,\hat{x}) = \sum_{t}^{1\ldots T} \sum_{j}^{1\ldots J} \frac{\|\hat{x}_{j,t} - x_{j,t}\|_2}{J \cdot T}$$

The MPJPE is simply the mean euclidean distance between the keypoints averaged over the number of frames or time steps (here T) and the number of keypoints or joints (here J).

Different evaluation procedures or protocols are used for emphasising different aspects of the evaluation. Each protocol defines an alignment of ground truth and estimates before the calculation of the MPJPE. Note that any motion sequence can always be rotated around the up-axis and translated in parallel to the ground plane. The resulting motion sequence would have an arbitrary large MPJPE but would conceptually be the same motion sequence. Therefore a global MPJPE is rarely used. One protocol is

the root-relative or local pose MPJPE: The estimate and ground truth are aligned at the root joint. The root joints are taken out of the set of evaluation keypoints, as their distances are now zero by design. Another common protocol is called procrustes-aligned MPJPE or PA-MPJPE: A procrustes analysis gives the optimal (under L2-distance metric) translation and rotation between estimated keypoints and ground truth keypoints on a frame by frame basis. The transformation is applied to the estimate before calculating the MPJPE. This effectively also ignores scale and rotation for the evaluation, additionally to ignoring the global placement similar to the root-relative MPJPE.

2.3.2 The Skinned Multi-Person Linear Model (SMPL)



(a) PC 1 (b) PC 2 (c) PC 3

Figure 2.6: He the variations of the template body mesh caused by the variation of the body shape parameter β is shown. The first two principal components of the SMPL body shape are varied from -2 to +2 standard deviations in (a) and (b). The third shape principal component (c) is varied from -3 to +3. Figure is taken from Loper et al. (2015)

The skinned multi-person linear model (*SMPL*, Loper et al. (2015)) is a model expressing a human's body mesh, skeleton and keypoints as a function of a body shape $\beta \in \mathbb{R}^{10}$ as a 10-dimensional PCA projection and the pose angles $\theta \in \mathbb{R}^{24\times 3}$ as 3D rotation vectors for 24 joints. The linear model is based on a PCA over the 3D body scans of hundreds of different subjects with different body shapes in even more different poses.

$$V_T(\beta) \in \mathbb{R}^{6890\times 3} \tag{2.30}$$

$$J_T(V_T(\beta)) \in \mathbb{R}^{24\times 3} \tag{2.31}$$

$$V(\beta,\theta) = \mathrm{LBS}(V_T(\beta), J_T(V_T(\beta)), \theta) \in \mathbb{R}^{6890\times 3} \tag{2.32}$$

$$J_{\mathrm{any}}(V(\beta,\theta)) \in \mathbb{R}^{N_{\mathrm{any}}\times 3} \tag{2.33}$$

The first three principal components of the body shape template mesh can be seen in figure 2.6. V_T linearly projects the first ten principal components into the body shape mesh space. The template body mesh has 6890 vertices. From the T-pose template mesh with the shape β a linear joint regressor J_T gives the 3D coordinates of the 24 joints of a predefined kinematic skeleton. Linear blend skinning (LBS), a way of deforming the vertices via a linear association with the rotating joints, is used to give the final posed body mesh V. Then, based on the final posed mesh, a joint regressor can be trained for any target skeleton definition with any number of joints. For this a motion capture dataset has to be converted into SMPL motion sequences and then the regression has to be trained with the original dataset's skeleton definition. This is useful for converting motion sequences between different datasets that have been recorded with different skeletons.

Note that for simplicity, we ignore the global transformation here. If global poses or vertices are required, a global transformation vector can be added to the keypoints or vertices at any stage. We use the SMPL model to represent and unify human motion from different sources throughout this work.

2.3.3 Marker Motion Capture to SMPL with MoSh for AMASS

To convert an arbitrary motion sequence from marker based motion capture to SMPL sequences, Loper et al. (2014) develop a method they call *MoSh*. MoSh, short for *motion and shape capture from sparse markers*, is done by first manually placing virtual marker positions on the SMPL template skeleton according to the marker set definition of the target motion dataset. The virtual marker can then be expressed as a barycentric combination of the three closest vertices, which also holds for the final mesh when the model is shaped and posed. They further assume that the marker position is not directly on the skin but on an orthogonal line pointing outwards from the mesh directly above the barycenter with some distance. Then for each frame a set of $\{\beta, \theta, \mathbf{t}\}$ (with $\mathbf{t} \in \mathbb{R}^3$ being the translation added to the SMPL vertices and keypoints) are optimized via Gauss-Newton to fit the virtual SMPL markers to the captured markers. Additionally some sequence-unique and heavily regularized wiggle parameters added to the barycentric coordinates and the distance are also optimized to allow for slight semantic misalignment of the manual marker definitions.

AMASS, the *archive of motion capture as surface shapes*, is a dataset of SMPL sequences unifying and based on 24 different marker based motion capture datasets. The dataset has been produced using the MoSh method.

2.4 Camera Projection

In this work we always assume the simple, undistorted perspective pinhole model. In this model the intrinsic parameters or *intrinsics* define how wide or narrow and how centered the projection is. The *extrinsics* define the position and rotation of the camera. We follow the OpenCV (Bradski, 2000) convention for cameras: A right hand coordinate system with positive Z being the camera look-at axis and Y pointing downwards.

2.4.1 Extrinsic Parameters

The extrinsic parameters (or *extrinsics*) define the translation and rotation of the camera. The translation is a 3D vector $\mathbf{t} \in \mathbb{R}^3$. The rotation can be expressed as an orthonormal matrix $\mathbf{R}$:

$$\mathbf{R} \in \mathbb{R}^{3\times 3} \tag{2.34}$$

$$det(\mathbf{R}) = 1 \implies \mathbf{R}^{-1} = \mathbf{R}^T \tag{2.35}$$

$\mathbf{R}$ can be crafted by chaining individual rotations around each of the axes X, Y and Z (Taubin, 2011):

$$\mathbf{R}(\alpha,\beta,\gamma) = \mathbf{R}_Z(\alpha)\cdot\mathbf{R}_Y(\beta)\cdot\mathbf{R}_X(\gamma) \tag{2.36}$$

$$\mathbf{R}_X(\gamma) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & cos(\gamma) & -sin(\gamma) \\ 0 & sin(\gamma) & cos(\gamma) \end{pmatrix}, \quad \mathbf{R}_Y(\beta) = \begin{pmatrix} cos(\beta) & 0 & sin(\beta) \\ 0 & 1 & 0 \\ -sin(\beta) & 0 & cos(\beta) \end{pmatrix} \tag{2.37}$$

$$\mathbf{R}_Z(\alpha) = \begin{pmatrix} cos(\alpha) & -sin(\alpha) & 0 \\ sin(\alpha) & cos(\alpha) & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{2.38}$$

Note that in chapter 4 and chapter 5 we assume a simplified model due to the arbitrariness of motion with respect to rotations around the up-axis and translations

along the ground plane. Hence there our camera can only be rotated around the X-axis and Y-axis and only be placed along the Z-axis:

$$\mathbf{R}(\beta,\gamma) = \mathbf{R}_Y(\beta) \cdot \mathbf{R}_X(\gamma) \tag{2.39}$$

$$\mathbf{t} = \begin{bmatrix} 0 \\ 0 \\ h \end{bmatrix}, \quad h \in \mathbb{R} \tag{2.40}$$

To obtain a point in camera space $p_c \in \mathbb{R}^3$ from a point in world space $p_w \in \mathbb{R}^3$ we simply apply the inverse camera transformation to the point:

$$p_c = \mathbf{R}^T \cdot (p_w - \mathbf{t})$$

2.4.2 Intrinsic Parameters

The intrinsic parameters of the undistorted perspective pinhole model are

- the x-axis focal length $f_x \in \mathbb{R}$
- the y-axis focal length $f_y \in \mathbb{R}$
- the x-axis center $c_x \in \mathbb{R}$
- the y-axis center $c_y \in \mathbb{R}$
- the x-axis image dimensions or width $w \in \mathbb{N}$
- the y-axis image dimensions or height $h \in \mathbb{N}$

They can be assembled into a projection or intrinsic matrix K:

$$K = \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix} \tag{2.41}$$

K transforms 3D points in camera space into 2D homogeneous coordinates in the image space. After normalizing the 2D homogeneous coordinates, the resulting 2D coordinates can be cut off by w and h, which gives the image set I:

$$p_{hi} = K p_c \tag{2.42}$$

$$p_i = \begin{bmatrix} \frac{p_{hi,x}}{p_{hi,z}} & \frac{p_{hi,y}}{p_{hi,z}} \end{bmatrix}^T \tag{2.43}$$

$$p_i \in I \quad if \quad \begin{cases} p_{hi,z} > 0 \wedge \\ \frac{-w}{2} < p_{i,x} < \frac{w}{2} \wedge \\ \frac{-h}{2} < p_{i,y} < \frac{h}{2} \end{cases} \tag{2.44}$$

This model can be further simplified with additional assumptions: The focal lengths can be assumed to be identical ($f = f_x = f_y$) or the center point can be assumed to be the center of the image's with and height ($c_x = \frac{w}{2}, c_y = \frac{h}{2}$). The simplification used is detailed in the context of the method.

Chapter 3

Known, Static Camera: Adversarial Learning for new Cameras

This chapter is based on work published as

Julian Habekost, Takaaki Shiratori, Yuting Ye, Taku Komura: Learning 3D Global Human Motion Estimation from Unpaired, Disjoint Datasets. British Machine Vision Conference (BMVC) 2020.

The work has been proposed and conducted mainly by myself, exclusively implemented by myself, with original ideas mostly from myself. Co-contributors assumed consulting roles, contributed ideas and helped researching related work.

3.1 Introduction

There is a growing demand in 3D human pose estimation from monocular videos for 3D motion capture, surveillance, autonomous driving, and motion analysis. By predicting the 3D pose of the subject, it will be easier to understand the context and predict the future status.

One of the main challenges of predicting 3D motion from videos using machine learning is that they require ground-truth training data, where the 2D monocular videos and the corresponding ground-truth 3D motion data are given. This significantly limits the availability of the training data as most high-quality motion capture data do not come with corresponding 2D monocular videos. It is specifically difficult to find large datasets with both modalities. If the data is available in a natural setting, such supervised training tends to overfit to the training data, where the motion types are limited

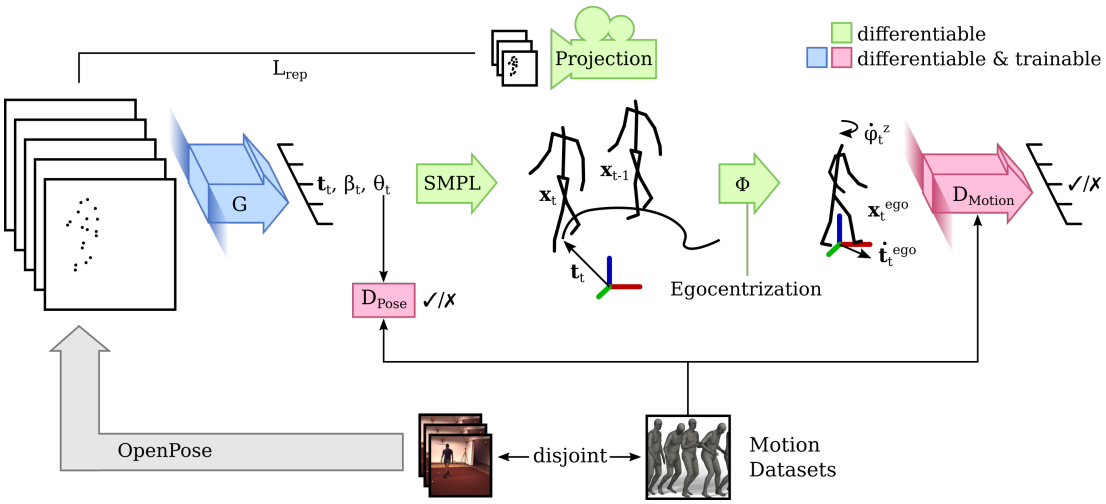


Figure 3.1: Overview of our system. Showing how disjoint motions datasets from the target videos can be used to estimate the subject's global motion. The temporal convolutional generator G estimates the global SMPL parameters within the camera space to compute the reprojection loss. For adversarial training based on motion feasibility, the motion is made independent from the camera via our differentiable egocentrization.

and the conditions of the 2D videos are rather biased. This results in poor performance when testing in-the-wild videos that contain arbitrary motion and background.

Another issue with existing methods using monocular videos is that they mostly compute only the local motion, where the root of the body is fixed, but not the global body motion, which describes how the body is translating in the world. This can be a limitation for applications such as 3D motion capture and detailed motion analysis.

In this chapter, we propose a novel method to compute both the local and global 3D motion of the human body from a 2D monocular video in a fully unpaired manner with disjoint datasets for estimation target and supervision signal. The overview of our system is shown in figure Figure 3.1. Our system receives a set of 2D body joint positions from an off-the-shelf 2D pose detector as input, and outputs the 3D joint positions as well as the root translation and orientation in the world space. It uses an unpaired, disjoint set of 2D keypoint detections and 3D motion capture data for training. Our 2D and 3D datasets come from completely separate sources where no 3D ground truth of the 2D data are included in the training set. The system is trained by a reprojection loss and an adversarial loss. To the best of our knowledge, this is the first method to estimate global motion from 2D keypoint detections on monocular videos using unpaired and disjoint datasets for the estimation target and supervision signals.

Our results are evaluated quantitatively through an ablation study and qualitatively via accompanied videos. We also compare our method with supervised architectures that require a pair of 2D videos and 3D motion capture data, trained on Human3.6M. Our results are comparable to supervised methods in terms of accuracy, even though our system is never trained on the ground truth of Human3.6M motion capture data.

To evaluate our system with the Human3.6M dataset, we retarget its motion data onto the SMPL (Loper et al., 2015) skeleton. To map the dataset consistently, we propose a motion retargeting framework where we optimize the meta-parameters of the target skeleton based on all the motion capture data by all the subjects jointly.

The contribution of this chapter is summarized as follows:

- a novel architecture for estimation of the 3D pose of a human subject including the global root motion from a monocular video using unpaired and disjoint datasets,
- a comprehensive evaluation of our framework in comparison to existing state-of-the-art approaches, and
- a motion retargeting framework that optimizes the meta-parameters of the target skeleton based on the entire Human3.6M dataset.

3.2 Related Works

In this section, we first review techniques on 2D human pose estimation from images. We next review techniques on 3D pose estimation from images, and finally those that make use of temporal coherence to predict the 2D/3D human motions from videos.

3.2.1 2D Pose Estimation

Human pose estimation from videos is a classic computer vision problem (Agarwal and Triggs, 2006). Most successful classical approaches are based on the Deformable Part Models (DPM) (Felzenszwalb et al., 2010), where the system recognizes each joint based on the features and their connectivity with adjacent joints. Hand-crafted features such as HOG, edges, colour histograms etc. are used for detecting the joints.

The performance of pose estimators has improved due to the usage of deep learning techniques. The model by Toshev and Szegedy (2014) significantly outperforms classical approaches which are based on hand-crafted features. Jain et al. (2014) apply

convolutional neural networks for computing the heatmaps for joints and then use the global position before computing their final positions. Newell et al. (2016) use multiple resolutions of the image to increase the precision of the local pose estimation. Cao et al. (2017) improve the body pose estimation using the Part Affinity Field that predicts joint connections.

3.2.2 3D Pose Estimation

There is an increasing interest in the prediction of 3D human pose from 2D images. Martinez et al. (2017) regress 2d detections from Newell et al. (2016) to 3D poses with a simple neural network. Zhou et al. (2017) expand Newell et al. (2016) by a depth regression and formulate geometric losses for semi-supervised learning, enabling combined training on data with 3D and 2D GT only. Mehta et al. (2017a) introduce a green screen dataset with exchangeable backgrounds for better generalization. They also optimize for reprojection in camera space to obtain global poses assuming the camera stays fixed. Kanazawa et al. (2018) estimate both 3D pose and shape based on SMPL (Loper et al., 2015), a parametric model of human shape and pose. A concurrent work by Pavlakos et al. (2018) similarly completes the same task.

These approaches predict human poses from individual images and thus suffer from jerkiness when applied to videos. In addition, due to depth ambiguity from monocular views, they can only predict root relative poses but not the global motion in the world space over time. We call these root relative human poses *local poses*.

Methods that consider temporal coherence can remove the jerkiness of the joints when blindly applying per-frame pose-estimators to video frames. Tekin et al. (2015) use motion compensation for 3D pose prediction but the poses still are root relative. Mehta et al. (2017b) filter poses over consecutive frames and optimize for reprojection in camera space. Their method, therefore, results in global motion sequences with translation as long as the camera is fixed. Dabral et al. (2018) extend Zhou et al. (2017) by a supervised temporal pose refinement for root relative poses. Hossain and Little (2018) and Pavllo et al. (2018) regress sequences of 2D poses to sequences of 3D (root relative) poses using LSTM and TCN, respectively. Xu et al. (2018) reconstruct pose, mesh, and cloth in the global space but needs an initialization per subject, where the subject's standing pose must be scanned by a video from different directions. Peng et al. (2018) learn 3d pose sequences in the world space from videos as a byproduct of learning reinforcement policies within a physical environment. In contrast to our

work, their method needs retraining for every video. Kanazawa et al. (2019) propose a system to predict the 3D motion of the person from a single image. Kocabas et al. (2019) use a temporal discriminator to learn smooth movements of the body. Arnab et al. (2019) use bundle adjustment of a sequence of poses based on consecutive single frame estimations.

We train our system purely with unpaired 2D and 3D motion capture data while also predicting global motion. Among the previous works, the majority of the works only predict the local poses of the subjects (except Mehta et al. (2017b); Peng et al. (2018); Xu et al. (2018)). Although some papers such as Kanazawa et al. (2018, 2019); Kocabas et al. (2019) claim unpaired training, their system is co-trained by paired data such as the Human3.6M training videos. This makes our work unique from existing works; we compare our system with existing methods later in this paper and show that it performs comparably even though it is trained in a fully unpaired fashion.

3.3 Methodology

We regress global 3D motion from 2D keypoint detections by learning a mapping to satisfy a motion critic, a pose critic, and a reprojection loss based on a known camera. We assume the camera is static and its intrinsics and extrinsics are known in advance. We use the off-the-shelf detector OpenPose (Cao et al., 2018) for obtaining keypoint estimates on test videos. A temporal convolutional generator learns a mapping from a sequence of 2D detections to a sequence of body shape, body pose and translation parameters for the differentiable SMPL human body model (Loper et al., 2015) (see section 3.3.1). To reduce the dimensional complexity of global motion, we then express it from the perspective of the subject (what we call *egocentrization*) in a differentiable way (see section 3.3.3). The global joint positions can be projected into 2D with known camera parameters for a reprojection loss (see section 3.3.4). A temporal convolutional discriminator is trained to distinguish the generated and real motion in a motion database (Mahmood et al., 2019) (see section 3.3.5). Further, we use a pose discriminator which only sees the SMPL pose angle parameters (see section 3.3.6). Note that the system never sees any video's 3D ground truth data. We also propose a heuristic for obtaining SMPL parameterized motion sequences from skeletal motion data (see section 3.3.8). This function is useful when marker data is not available and MoSh (Loper et al., 2014) cannot be used.

3.3.1 3D Global Motion Generator for SMPL Sequences

The temporal convolutional generator G (see Figure 3.1) takes a sequence of 2D body keypoint detections $\mathbf{X}_t^{2d} \in \mathbb{R}^{19\times 2}$ from OpenPose as an input and maps it to a sequence of SMPL (Loper et al., 2015) parameters $\{\beta, \theta, \mathbf{t}\}_t$ including the body shape PCA basis $\beta \in \mathbb{R}^{10}$, the pose angles $\theta \in \mathbb{R}^{24\times 3}$ in rotation vector representation for 24 joints of the SMPL skeleton including the global root orientation and the root global translation $\mathbf{t} \in \mathbb{R}^3$ in meters. The body rotation is hereby included in θ as the first joint's rotation. Formally G is defined as:

$$\begin{bmatrix} \hat{\beta}_1, & ..., & \hat{\beta}_{N_t} \\ \hat{\theta}_1, & ..., & \hat{\theta}_{N_t} \\ \hat{\mathbf{t}}_1, & ..., & \hat{\mathbf{t}}_{N_t} \end{bmatrix} = G(\mathbf{X}^{2d}_{1,...,N_t}) \tag{3.1}$$

$$G(\mathbf{X}^{2d}_{1,...,N_t}) \in \mathbb{R}^{85\times N_t} \tag{3.2}$$

3.3.2 Network Architecture

G is a temporal convolutional neural network (see Figure 3.2, left) inspired by the U-Net (Ronneberger et al., 2015b) architecture. It consists of *down sampling blocks*, *up sampling blocks* and *keep blocks*. A down-sampling block halves the temporal width by a 1d convolution layer with stride two while also doubling the number of channels. The up-sampling block uses nearest neighbour upsampling to double the temporal width before a 1d convolution layer halves the channel size. The keep block is a 1d convolution that keeps the input's and output's temporal width and channel size the same. All blocks also feature a ReLU (Nair and Hinton, 2010) after the convolution except when a block is used as the last output layer. To assemble the network N_d down-sampling blocks are followed by N_k keep blocks which in turn are followed by $N_u = N_d$ up-sampling blocks. Also, the nth down-sampling block is connected to the $(N_U - n)$th up-sampling block via residual connections. We chose $N_d = N_u = 3, N_k = 1$, a filter length of 15, a temporal window length of 128 frames at 10fps and a channel size of 512 after the first down-sampling block for this generator.

The network architecture has been chosen for its relative simplicity and empirical robustness after iterating with different architectures. The U-structure makes it possible to easily calculate and control the temporal receptive field and the compression depth. While this is one that works for this task, we do not claim that it is the only or even best network architecture. With some iterative hyper-parameter tuning, other

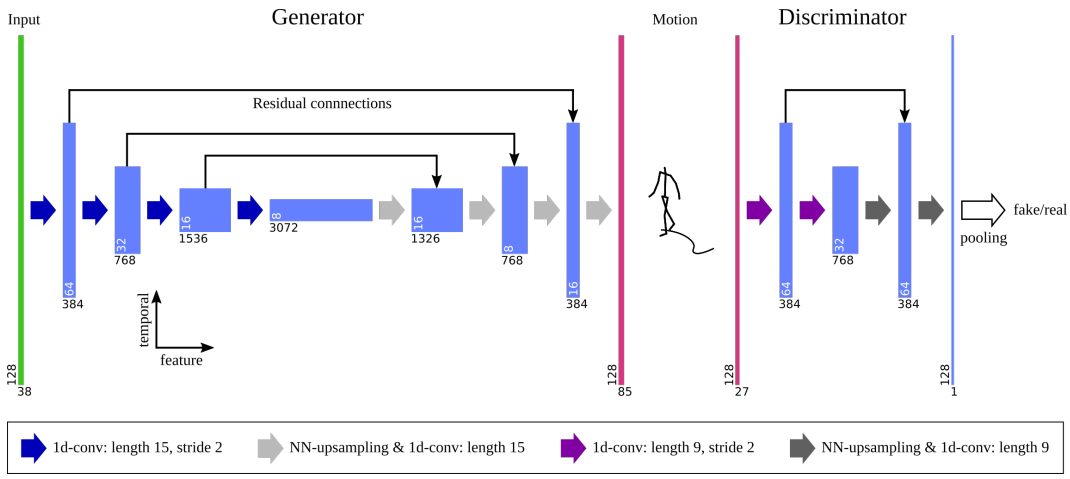


Figure 3.2: The generator (left) and discriminator (right) structures of our system. Note that the generator's output motion is processed through our differentiable egocentrization before it is inputted to the discriminator, which is not depicted here.

temporal neural network designs should also achieve comparable results. See subsection 2.1.3 for a discussion of temporal neural networks.

3.3.3 Differentiable Motion Egocentrization

In order to construct a useful global motion representation for adversarial motion learning we convert the global motion $\{\beta, \theta, \mathbf{t}\}_t$ into the subject's egocentric coordinate system, which we call egocentrization (Φ in Figure 3.1). The motivation stems from the fact that for example, a straight walk is the same motion regardless from which to which point and at which angle it is performed. By transforming the representation from the world coordinates into the subject's egocentric coordinate system, which is a process that we call egocentrization, the motion becomes invariant to arbitrary rotations and translations while keeping the temporal path within the sequence intact. This reduces the dimensional complexity without losing relevant information. The original global translation and rotation can be obtained by forward integration.

While similar representations have been used in data preprocessing pipelines for character animation modelling (Holden et al., 2016) we need to construct it as a differentiable function of a SMPL (Loper et al., 2015) parameter sequence to use it online for training. We call this differentiable egocentrization function Φ. First, we need the SMPL accompanied default world space 3D joint position computation module, here denoted J: $\mathbf{x}^w = J(\beta, \theta, \mathbf{t})$, where $\mathbf{x}^w \in \mathbb{R}^{24\times 3}$ are the position of the joints in the world

space. Φ further maps $\mathbf{x}^w$ to joint positions in egocentric space ($\mathbf{x}^{ego}$), the velocity in egocentric space ($\dot{\mathbf{t}}^{ego}$) and the angular velocity around the z-axis ($\dot{\varphi}_z$), with z being the up axis. For this we set the world space joints $\mathbf{x}^w$ with its hips to the origin $\mathbf{x}_0 = \mathbf{x}^w - \mathbf{t}_{x,y}$ on the horizontal plane using $\mathbf{t}_{x,y}$, the global translation $\mathbf{t}$ with the z-component set to zero. Let $\mathbf{R}$ be the rotation matrix that rotates $\mathbf{x}_0$ around the z-axis so that the subject looks in $+x$ direction and the hips are aligned with the y-axis. This gives

$$\{\mathbf{x}^{ego}, \dot{\mathbf{t}}^{ego}, \dot{\varphi}_z\} = \Phi(\mathbf{x}^w) \tag{3.3}$$

$$\mathbf{x}^{ego} = (\mathbf{R}\mathbf{x}_0^\intercal)^\intercal \tag{3.4}$$

$$\dot{\mathbf{t}}^{ego} = \mathbf{R}(\mathbf{t}_t - \mathbf{t}_{t-1}) \tag{3.5}$$

$$\dot{\varphi}_z = -(\mathbf{R}_{t-1}^\intercal \mathbf{R}_t)_{i=1,j=1}. \tag{3.6}$$

All these operations are differentiable so we can use TensorFlow's autograd feature to obtain the gradients. See subsection 2.1.1 why and how autograd works in the context of backpropagation. Following we show a python implementation of Φ.

```python
class SMPL_ego(SMPL):
    def __init__(self, pkl_path, joint_type='cocoplus', dtype=tf.float32):
        super().__init__(pkl_path, joint_type, dtype)
    def __call__(self, beta, theta, trans, name=None):
        verts, coco_joints, Rs = super().__call__(beta, theta, get_skin=True, name=name)

        a = self.J_transformed[:, 1] - self.J_transformed[:, 0] # left hip - center hip
        b = self.J_transformed[:, 2] - self.J_transformed[:, 0] # right hip - center hip

        x_dash = (a[:, 0:2] - b[:, 0:2]) / tf.norm(a[:, 0:2] - b[:, 0:2],
                                                   axis=1,
                                                   keepdims=True)

        ones = tf.ones_like(a[:, 0])
        zeros = tf.zeros_like(a[:, 0])

        Ri = tf.reshape(tf.transpose(tf.stack([x_dash[:, 0], x_dash[:, 1], zeros,
                                               -x_dash[:, 1], x_dash[:, 0], zeros,
                                                      zeros,        zeros, ones])),
                        [-1, 3, 3])


        root_rotations_z_aligned = (Ri @ Rs[:, 0])
        joints_z_aligned = tf.transpose(
            Ri @ tf.transpose(self.J_transformed[:, 1:]
                              -self.J_transformed[:, 0:1], [0, 2, 1]), [0, 2, 1])
        trans_velocities = \
        (Ri[1:, :2, :2] @ (trans[1:, :2, None]
                           -trans[:-1, :2, None]
                           -self.J_transformed[1:, 0, :2, None]
                           +self.J_transformed[:-1, 0, :2, None]))[:, :, 0]
        trans_heights = trans[1:, 2]
        global_velocities_z_aligned = (joints_z_aligned[1:] - joints_z_aligned[:-1])
                                      + tf.concat([
                                            trans_velocities,
                                            trans[1:, 2:3]-trans[:-1, 2:3]
                                        ], axis=1)[:, None]
        rot_velocities = tf.reduce_sum(Ri[1:, :2, 1] * Ri[:-1, :2, 0], axis=1)

```

```
        return verts[1:], self.J_transformed[1:], coco_joints[1:], Rs[1:], # vanilla SMPL
               root_rotations_z_aligned[1:], joints_z_aligned[1:], trans_velocities,
               trans_heights, rot_velocities, global_velocities_z_aligned, Ri[1:]
```

Listing 3.1: Python implementation of Φ

3.3.4 Reprojection Loss

The only source of evidence for the target 3D human motion are 2D observations from our off-the-shelf 2D keypoint estimator. The reprojection loss is tailored to let G learn from 2D observations by forcing it to generate 3D human motion that explains the 2D keypoint observations.

The reprojection loss L_{rep} aligns the generated global poses $\{\beta, \theta, \mathbf{t}\}_t$ with the input 2D keypoint detections $\mathbf{X}_t^{2d}$ by projecting them with given camera parameters. Because the default SMPL skeleton does not match OpenPose's underlying assumed 3D skeleton, we follow the common practice of using a learned regression from the vertices of the SMPL output mesh to the target skeleton. Here we denote $V(\beta, \theta, \mathbf{t})$ as the differentiable SMPL function to calculate the 3D position of the body mesh vertices. $J_{OP}(V(\beta, \theta, \mathbf{t})) \in \mathbb{R}^{19\times 3}$ maps the vertices further to the OpenPose skeleton with 19 keypoints. We simply use the OpenPose keypoint regressor that is available and trained in the author's original SMPL implementation. The camera projection $P(\cdot)$ then calculates the 2D projections given the camera's rotation $\mathbf{R}_{cam} \in \mathbb{R}^{3\times 3}$, translation $\mathbf{t}_{cam} \in \mathbb{R}^3$, focal length $f \in \mathbb{R}^2$ and center point $c \in \mathbb{R}^2$. Hence the reprojection loss is

$$L_{rep} = \sum_{t}^{N_t} |P(J_{OP}(V(G(\mathbf{X}^{2d}_{1,\dots,N_t})_t)), \mathbf{R}_{cam}, \mathbf{t}_{cam}, f, c) - \mathbf{X}_t^{2d}|_1. \tag{3.7}$$

For this we iterate over all joints in all time steps and sum up the L1-distances between the reprojected and the detected keypoints.

3.3.5 Motion Discriminator

The motion discriminator D_{Motion} (see Figure 3.1) compares the generated egocentric motion with real motion examples. It is trained to tell dataset based real global 3D human motion apart from generated global 3D human motion. Then the generator is trained to fool the discriminator, which is a minimax game (see subsection 2.1.4). Through this the generator learns to propose realistic global 3D human motion sequences.

D_{Motion} is a convolutional neural network with a similar structure as the generator G (see Figure 3.2, right). It uses a shorter filter length of nine, only one down, upsampling and keeps block and 384 channels. This yields a shorter receptive field of about three seconds at ten fps. We use the loss and gradient penalty described in Gulrajani et al. (2017):

$$L_{D_M} = \sum_t [D_{Motion}(\Phi(\mathbf{x}^{real}_{1,\dots,N_t}))_t - D_{Motion}(\Phi(J(G(\mathbf{X}^{2d}_{1,\dots,N_t}))))_t] + c^{Motion}_{pen} R^{Motion}_{pen}, \tag{3.8}$$

where R^{Motion}_{pen} is the gradient penalty and $c^{Pose}_{pen} = 10$.

3.3.6 Pose Discriminator

The pose discriminator learns to differentiate real 3D local human poses sourced from a dataset from generated 3D local human poses. It forces the generator to mimic the 3D poses from the dataset through an adversarial minimax game (see subsection 2.1.4).

We use a time-independent pose discriminator D_{Pose} (see Figure 3.1) on all 23 relative joint angles in a generated θ_t. Hence we ignore the first angle in θ_t, which is the pose's absolute rotation angle. Note that this allows the generator to freely rotate the subject globally and avoid discontinuities that are found in datasets that do not use rotations above 2π. The real pose examples are drawn from AMASS dataset (Mahmood et al., 2019). D_{Pose} is a feed-forward network with two hidden layers with dimensionality 512. Like for D_{Motion} we use the loss and gradient penalty described in Gulrajani et al. (2017):

$$L_{D_P} = \sum_t [D_{Pose}(\mathbf{x}^{real}_t) - D_{Pose}(J(G(\mathbf{X}^{2d}_t)))] + c^{Pose}_{pen} R^{Pose}_{pen}, \tag{3.9}$$

where R^{Pose}_{pen} is the gradient penalty and $c^{Motion}_{pen} = 0.0001$.

3.3.7 Loss Composition and Training

The generator G and the discriminators D_{Motion}, D_{Pose} are trained as a minimax game (see subsection 2.1.4) with adversarial training. The above defined discriminators are trained to distinguish real 3D human motion from generated one, respectively real 3D local poses from generated ones. They are minimized on real data and maximized on generated data. As the minimax counter the generator is then trained to minimize the discriminator scores when given as an input to the discriminators. This forces the generator to mimic real data, at least to the degree of the capabilities of the discriminators.

Note that this is an unbounded loss w.r.t both sides. The stability of the loss is a necessary condition for the generator's ability to learn distributions. We use the improved Wasserstein-GAN (Gulrajani et al., 2017) losses with gradient penalty to stabilize the minimax game without interfering or guiding too heavily, further details are explained in subsection 2.1.4.

The adversarial minimax game is the aspect that forces the generator to propose actual human motion as the solution to the 2D to 3D lifting problem. The reprojection loss alone would be under-constrained by one dimension, the depth. This is why adversarial and reprojection losses are combined. Following the formulation of the generative adversarial minimax game (see subsection 2.1.4 and Goodfellow et al. (2014)), we use the following loss for the generator:

$$L_G = \sum_t [D_{Motion}(\Phi(J(G(\mathbf{X}^{2d}_{1,\dots,N_t})))_t + D_{Pose}(J(G(\mathbf{X}^{2d}_t)))] + c_{rep}L_{rep}. \tag{3.10}$$

where $c_{rep} = 10000$ and $c_\beta = 100$. We use the Adam (Loshchilov and Hutter, 2017) batched stochastic gradient descent with a batch size of 32. One iteration trains D_{Motion} five times, D_{Pose} three times and G one time. We train for 25k iterations. We use the AMASS (Mahmood et al., 2019) motion dataset, which itself is a compilation of multiple motion datasets reparameterized as SMPL sequences.

3.3.8 Motion Retargeting by Optimizing Meta Parameters

For our supervised comparison experiment we need the Human3.6M 3D ground truth as a SMPL parameter sequence. The SMPL skeleton differs from the Human3.6M skeleton and thus we need to retarget the Human3.6M dataset to the SMPL skeleton. [1]

Given a set of motion sequences with N_S subjects, we subsample the dataset to a reasonable size so that it completely fits on a single GPU (we use 1fps for Human3.6M). For each of the SMPL joints, we define sets of joints in the source skeleton: Each SMPL joint j is assigned the most similar source skeleton joint and all neighbouring joints $n^j_1, \dots, n^j_{Nj}$. We optimize the convex combination of these joints to have the same position as the corresponding SMPL joint. Hence given a set of linear coefficients $C = \{c_{ij} | \forall ij\}$ the cost function for a single pose is:

$$L_{single}(C, \beta_S, \theta) = \sum_j^{24} \left\| J(\beta_S, \theta)_j - \sum_{i=1}^{N_j} c_{ij} \mathbf{v}'_{n^j_i} \right\|_2 + \left\| 1 - \sum_{i=1}^{N_j} c_{ij} \right\|_2, \tag{3.11}$$

[1] Previously (Kanazawa et al., 2018, 2019; Kocabas et al., 2019) this conversion was achieved with marker based MoSh(Loper et al., 2014). The resulting data has since been removed from the internet and the official Human3.6M dataset does not feature the marker data.

where $\mathbf{v}'_a$ is the position of a joint a in the source skeleton. The coefficients c_{ij} are shared among all bodies and poses. Thus for all body shapes $B = \{\beta_1, ..., \beta_{N_S}\}$ and poses $\Theta = \{\theta_{S,t} | \forall S, t\}$ we optimize

$$L_{total}(C, B, \Theta) = \sum_{S}^{N_S} \sum_{t}^{N_t^S} L_{single}(C, \beta_S, \theta_{S,t}) \tag{3.12}$$

with respect to C, B, Θ using gradient descent until convergence.

3.4 Evaluation

Four different variants of the proposed method are evaluated with respect to the global translation error and local pose quality: The method described above, an ablation method without the motion discriminator, another ablation method without the pose and motion discriminators, i.e. without any discriminators, hence the loss is just the reprojection loss. And fourth, a supervised method that simply trains G with ground truth Human3.6M motion data.

The global translation of the estimated human on Human3.6M is compared based on the four variants. The influence of different components of the model on the ability to produce global human motion is investigated. Note we do not expect to beat the supervised baseline as it is using ground truth Human3.6M motion for supervision, which is not available to our method. It is simply a benchmark of how close the proposed method can come to a method that learns from ground truth motion.

A good experimental setup to demonstrate our method's capabilities against others would be to recreate the Human3.6M dataset with different cameras and compare the local pose performance against methods that usually train and test on Human3.6M. Sadly, this would have exceeded the budget for this project. A newly created target dataset for the proposed method needs to be at least as diverse in actors and scenarios as Human3.6M, otherwise our discriminators can simply learn to identify and distinguish the generated motion by the few actors and scenarios, instead of the motion quality itself. Hence the proposed method is compared against other methods on the existing Human3.6M dataset and the comparison is based on local, root-relative 3D human pose performance, as that is the typical Human3.6M performance metric. The root relative or local pose performance is a good indicator for general motion quality even with a method like ours that can estimate global human motion. But we do expect to have a significant disadvantage against methods that train with Human3.6M ground

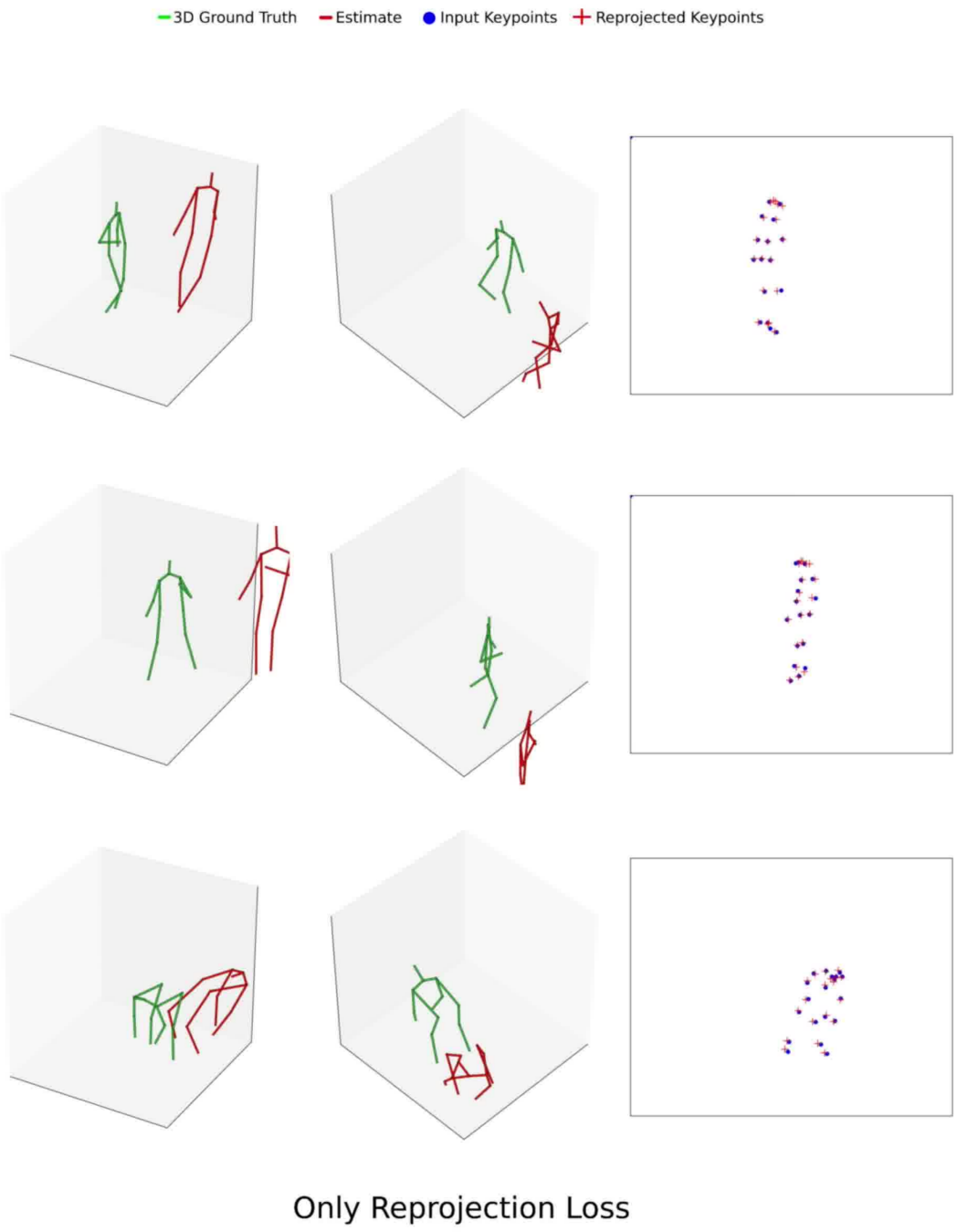


Figure 3.3: With only the reprojection loss the poses hardly resemble humans. Each row shows a different point in time in the evaluation sequence: The first two columns are different views of the ground truth in green and the estimate in red. The last row shows the OpenPose input keypoints in blue and the estimate's reprojected keypoints in red from the view of the original camera.

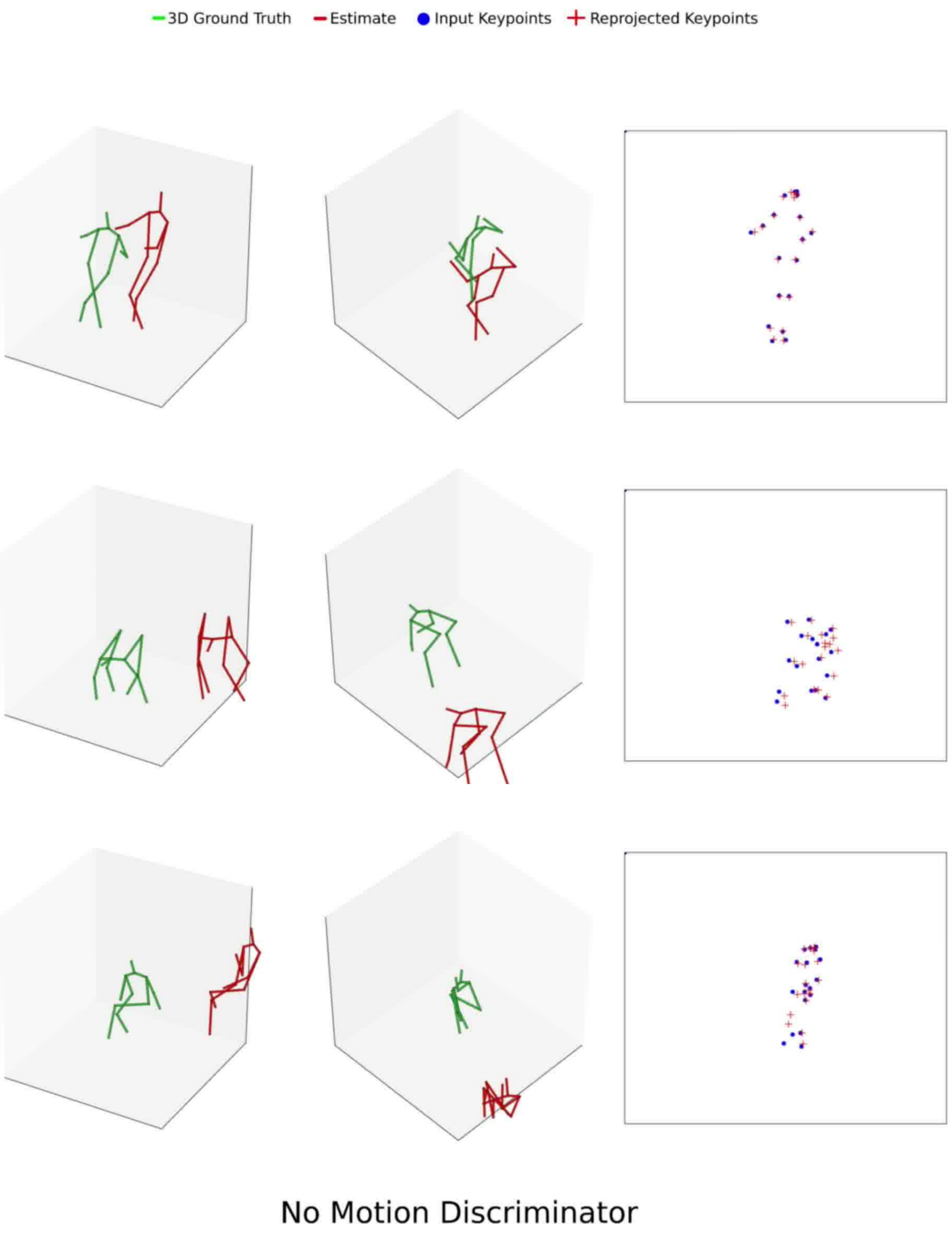


Figure 3.4: The pose discriminator alone yields realistic poses, but due to the missing global motion discriminator, the resulting global motion is not stable. Each row shows a different point in time in the evaluation sequence: The first two columns are different views of the ground truth in green and the estimate in red. The last row shows the OpenPose input keypoints in blue and the estimate's reprojected keypoints in red from the view of the original camera.

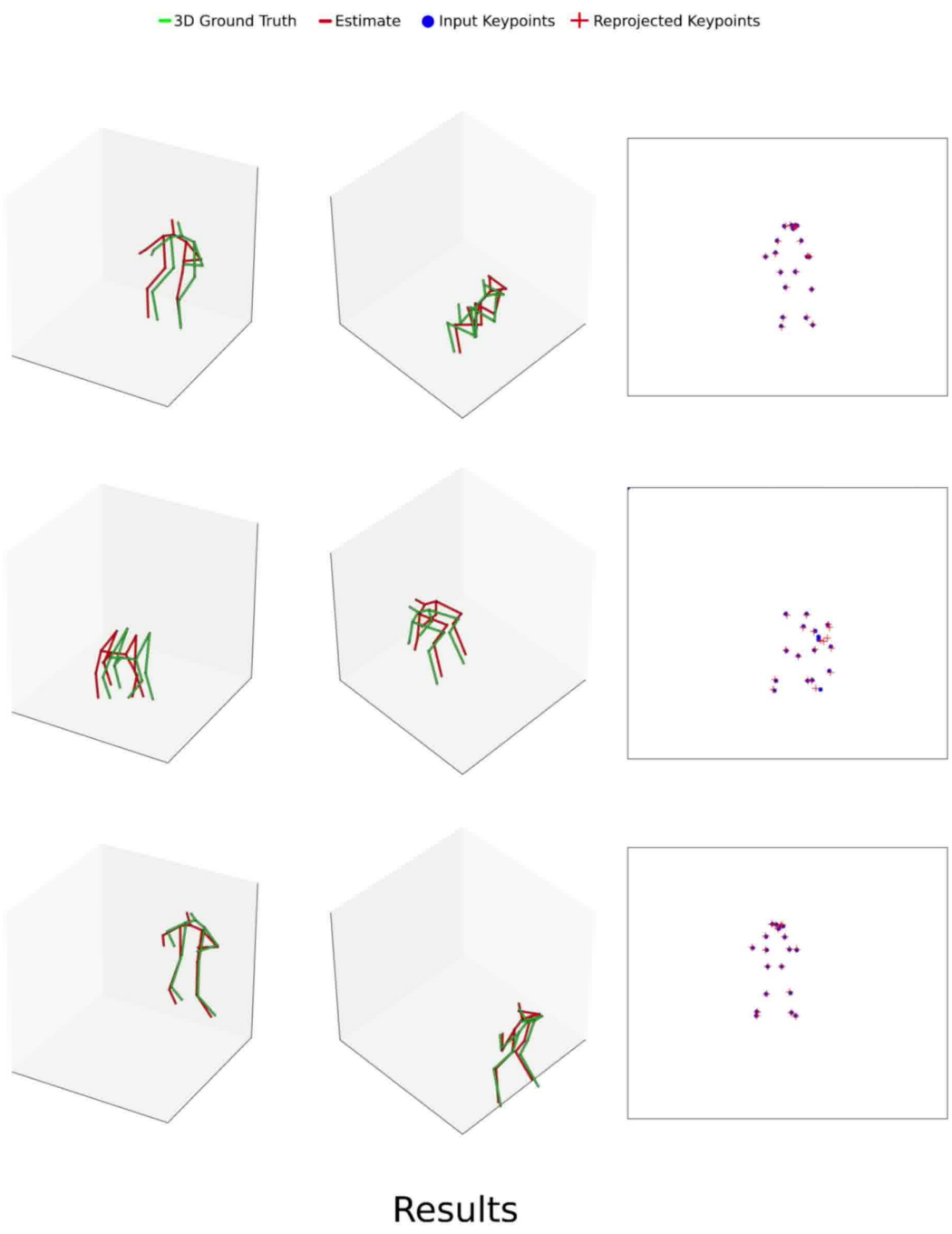


Figure 3.5: The proposed method with pose and motion discriminator shows the best local and global human pose performance. Each row shows a different point in time in the evaluation sequence: The first two columns are different views of the ground truth in green and the estimate in red. The last row shows the OpenPose input keypoints in blue and the estimate's reprojected keypoints in red from the view of the original camera.

truth. They inadvertently **implicitly** overfit to the camera and scale of the scene when training on the Human3.6M scene and camera. The proposed method instead **explicitly** fits to the camera and scale of the scene but does so without needing any Human3.6M ground truth motion. We hypothesize that on a Human3.6M-like dataset with a different camera but no ground truth available for training, we could beat these methods. But due to lack of such a dataset we cannot demonstrate or prove this and instead show how close our method comes on Human3.6M against methods that have the advantage of being trained on Human3.6M ground truth motion.

3.4.1 Training and Setup

The target dataset is Human3.6M (Ionescu et al., 2014). We use the videos as an input to OpenPose (Cao et al., 2018) as a starting point for our method as described in our methodology. We do not use any Human3.6M ground truth motion sequences for training. We use the AMASS Mahmood et al. (2019) dataset as a source of real motion sequences for our discriminator training. Also see subsections 2.3.1 and 2.3.3 for further details on Human3.6M and AMASS. All methods are trained for 100 epochs of Human3.6M videos, respectively their OpenPose keypoints.

The biggest outlier here is the supervised baseline. We train a convolutional neural network with the same architecture as the generator G to supervisedly regress ground truth Human3.6M motion sequences $\{\beta, \theta, \mathbf{t}\}_t$ from 2d keypoint detections from the same camera. The ground truth Human3.6M sequences are obtained by using the method described in section 3.3.8. We train on subjects 1, 5, 6, 7, 8 and report, like in all methods, for subjects 9 and 11 as evaluation.

3.4.2 Translations

We report on subjects 9 and 11 on our target dataset Human3.6M. We show our mean global translation error per frame in millimetres (see table 3.1). Due to a lack of work

Table 3.1: Translational error (mm).

Supervised Baseline	120.1
Ours (only L_{rep}, no discriminators)	901.1
Ours (without D_{Motion})	718.3
Ours	259.3

Table 3.2: Mean per joint position error (MPJPE) in mm. We mark a method as *unpaired* if 2D image space and 3D human motion ground truth pairs are assumed to be unknown. The more strict *disjoint* demands 2D image space and 3D human motion supervision to come from strictly different datasets.

	MPJPE	unpaired	disjoint
Martinez et al. (2017)	62.9	✗	✗
VIBE (Kocabas et al., 2019)	65.6	✗	✗
VNect (Mehta et al., 2017b)	80.5	✗	✗
Pavllo et al. (2018)	46.8	✗	✗
HMR (Kanazawa et al., 2018)	88.0	✗	✗
HMR (Kanazawa et al., 2018) unpaired	106.8	✓	✗
Supervised Baseline	72.2	✗	✗
Ours (only L_{rep}, no discriminators)	246.6	✓	✓
Ours (without D_{Motion})	189.2	✓	✓
Ours	118.2	✓	✓

for a fair comparison, we train a simple supervised baseline for a comparison (see section 3.4.1). Note that the baseline in this setting is quite strong and its practical use would be limited to datasets with ground truth data only. Still, we are coming reasonably close.

We also show in our ablation experiments that the motion discriminator plays an important role. Without it, the generator has trouble positioning the subject close to the ground truth. If the pose discriminator is also removed the error again rises, but not as drastically as before. Note that all figures 3.3, 3.4, 3.5 show a reasonable reprojection performance qualitatively by looking at the re-projections. But only figure 3.5, the version with the motion discriminator, shows a reasonable estimate of the global translation of the subject. We argue that the reason for this is that global motion cannot float freely in space, cannot change its skeleton structure or bone lengths. E.g. the generator needs to learn to keep the pose stable with respect to a foot on the ground, i.e. it learns implicit foot contacts. This constrains the motion to one solution, which is the one that fits the ground truth global translation.

3.4.3 Root Relative Poses

As previously proposed, we report the root relative pose error on subjects 9 and 11 on the Human3.6M dataset. Note that our system never sees any Human3.6M 3D ground truth data except for this evaluation. We align the estimated pose with its 3D ground truth via the root joint and report the mean per joint position error (MPJPE) in mm. See subsection 2.3.1 for details on the evaluation process without procrustes alignment. We compare our results to a selection of previous works as well as our own supervised baseline (section 3.4.1) and our ablation experiments in table 3.2. We mark a method as *disjoint* in table 3.2 if 2D image space and 3D human motion supervision signals come from strictly different datasets.

We almost reach state-of-the-art performance with a significant conceptual disadvantage of not using any Human3.6M ground truth in training. We also use less non-pose training target data (images, videos or key points) than any other method except for Martinez et al. (2017), which uses the same amount. As mentioned before, we argue that there is a possibility that we would beat those methods on a Human3.6M-like dataset with a different camera setup where a ground truth is not available for training. Creating such a dataset would have exceeded our scope and budget.

Our ablation (as shown in table 3.2) shows that both discriminators improve the pose configuration quantitatively. Note that the pose discriminator also restricts the pose angles to a feasible range similar to the discriminator in Kanazawa et al. (2018). Without it, joints can rotate in an unnatural way to reach poses that look valid to the motion discriminator as it is only looking at the positional configuration. The effect of the pose discriminator can also be seen comparing figure 3.3 and figure 3.4. The difference between those figures is exactly and only the pose discriminator. You can see that in figure 3.4 with the pose discriminator, at least poses are valid human poses, although they are misplaced globally. In figure 3.3 there are some valid poses but also ones that are definitely not anatomically possible. Still the pose discriminator is not enough to obtain the best possible local pose quality. The motion discriminator actually improves the local pose quality through the proxy of forcing the generator to adhere to realistic global motion.

3.5 Discussion

The reason the motion discriminator improves not only the global translation but also the local pose configuration error is that it enforces temporal coherence that a pose discriminator alone cannot impose. Even if all poses are enforced to be valid, the body can still float or slide over the ground freely. Also, the body shape and the pose can switch between frames unnaturally. The readers are referred to the supplementary video for the visual details.

The general problem of finding the absolute distance of an object in camera space is impossible to solve without knowing its size. In our method, the sizes of the subjects are learned indirectly because world contacts (like the foot on the ground) need to be hypothesized correctly by the generator to produce a feasible motion. The ground contacts imply a scale when the ground plane location is known.

Our method is robust to dataset changes by design. Since we do not use supervision from any target dataset we provide a lower bound of what is achievable with guaranteed no target dataset-specific overfitting.

A disadvantage to common methods is that for a new setting, the camera needs to be calibrated and the system needs to be retrained. Once this is done similar results like those reported in this paper can be expected to be achieved. For this, our method only needs motion data for training. Motion data is more easily available and often licensed less restrictively than datasets with video and 3D ground truth, which other methods need for their training.

To further advance our method one could investigate how to transfer information about a subject's body shape over longer temporal distances in a computationally feasible way. Currently, we achieve this for short temporal distances by regularizing body shape stability but this is limited in terms of information flow by the length of the receptive field. It might also be useful to extend this method to a multi-view task.

3.6 Conclusion

We propose an unpaired, disjoint system to estimate the 3D human poses from 2D keypoints extracted from the state-of-the-art 2D pose detector. To achieve this task, we propose a fully differentiable pipeline composed of egocentrization, generator, pose discriminator and motion discriminator. Our system shows comparable results to su-

pervised frameworks, which have much more restrictions on the data that can be used for training.

As future work, we are interested in extending our technique to predict the camera parameters in videos where the camera is dynamically moving. I present our work on this in chapter 5.

Chapter 4

Unknown, Static Camera: Synthesizing Simple Sequences

This chapter is based on work published as

Julian Habekost, Kunkun Pang, Takaaki Shiratori, Taku Komura: From Synthetic to One-Shot Regression of Camera-Agnostic Human Performances. In: Pattern Recognition and Artificial Intelligence. ICPRAI 2022. Lecture Notes in Computer Science, vol 13363. Springer, Cham.

The work has been proposed and conducted mainly by myself, exclusively implemented by myself, with original ideas mostly from myself. Co-contributors assumed consulting roles, contributed ideas and helped researching related work.

4.1 Introduction

3D human performance estimation from monocular videos is a challenging topic that attracts researchers' attention from various areas such as computer animation, virtual reality, surveillance, health care etc. One major problem is that this task is entangled with the camera: If a person is lying or standing straight on the floor can either be judged through a high level of visual understanding of the surroundings or other information about the camera angle and position. This is why the early human pose estimation task is only concerned with camera-relative body poses (Martinez et al., 2017; Mehta et al., 2017a). But the difference between laying and standing matters for performance capture, so subsequent work started to assume the camera intrinsics and extrinsics as given (Shimada et al., 2020; Xie et al., 2021).

A. Scale based Depth Ambiguity B. FoV based Depth Ambiguity

Figure 4.1: Depth ambiguities of unknown monocular cameras. For all the depicted cases the person appears similarly sized at the camera's viewfinder. We cannot differentiate between smaller people close and taller people further away (A.). Second, the field of view (FoV) influences how near or far objects and people appear (B.).

We instead propose a method that learns to regress the extrinsics and intrinsics together with the 3D performance from 2D human motion. Our method assumes and is useful when the camera is unknown but static throughout one motion sequence and the motion is performed on a ground plane. Apart from the illustrated rotation ambiguity, this also is supposed to solve the FoV (field of view) depth ambiguity depicted in Figure 4.1 B. That is possible just from 2D motion can be understood when imaging a person coming towards the camera with a constant walking speed: A large FoV will make the person's projected 2D size increase quicker. Further, we can learn the ground plane implicitly through foot contacts and other types of interaction of the subject with the ground plane. To achieve this our method uses

- a synthetic dataset that renders minute-long videos with various settings of the body shape, camera parameters, occluders and backgrounds,
- DensePose (Riza Alp Guler, 2018) to obtain an intermediate 2D motion representation,
- and a model that can do one-shot regressions on arbitrarily large sequences with global temporal information flow.

We show that global motion estimation and explicit camera estimation are possible with our method. Further, our method is the only domain generalization approach to the popular Human3.6M (Ionescu et al., 2014) dataset known to us. We also beat all Human3.6M domain adaptation tasks in local pose performance.

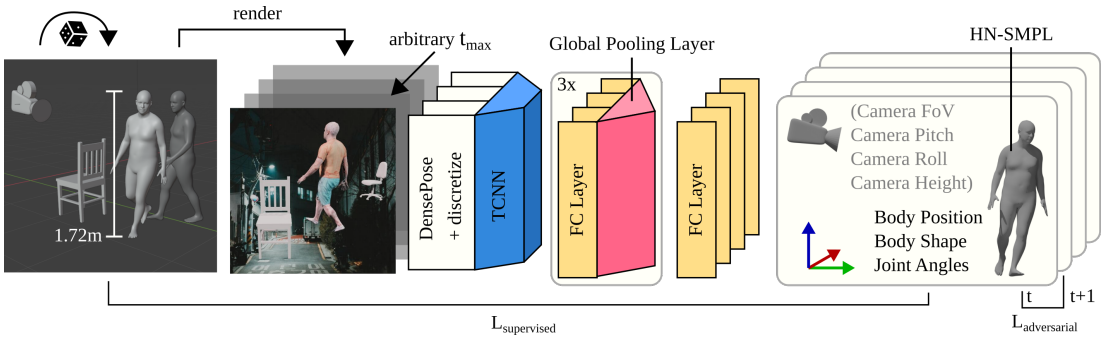


Figure 4.2: We render differently body-shaped subjects scaled to a body height of 172cm together with small occlusions in front of a random background. DensePose (Riza Alp Guler, 2018) gives us projected UV maps that we then discretize for training feasibility. Global pooling enables bidirectional information flow without temporal limits. The HN-SMPL is a height-normalized PCA-based subspace of SMPL.

4.2 Related Work

4.2.1 Local pose estimation

Local pose estimation is the task of estimating a human's pose rotated relative to the camera and with the hip centred at the origin. Martinez et al. (2017) regress 2D keypoint detections (Newell et al., 2016; Cao et al., 2017) to 3D joint positions. Zhou et al. (2017) directly incorporate a depth regression into a formerly 2D-only keypoint regressor. They use an adversarial pose prior to when only 2D supervision is available. Kanazawa et al. (2018) estimate both 3D pose and body shape based on SMPL (Loper et al., 2015), a parametric model of human shape and pose. Pavllo et al. (2018) introduce a temporal model with dilated convolutions.

Apart from our focus on the harder task of estimating global motion rotated relative to the world, another contrast is that these methods assume the camera intrinsics or field of view as known; in most cases (Martinez et al., 2017; Zhou et al., 2017; Kanazawa et al., 2018; Mehta et al., 2017a; Pavlakos et al., 2018; Pavllo et al., 2018; Kocabas et al., 2019) implicitly through training and evaluating on the same cameras in Human3.6M (Ionescu et al., 2014).

4.2.2 Global pose estimation

Here, the estimated human poses are rotated and translated relative to the world's ground plane. Mehta et al. (2017b) use procrustes alignment style post-processing on

per-frame estimations with a given camera to obtain a temporally consistent real-time global pose from a local pose estimator. Shimada et al. (2020) establish a baseline by optimizing 2D and 3D key points from a local pose estimator (Kanazawa et al., 2019) to match a reprojection loss given the camera extrinsics and intrinsics. This shows that with the known camera and 2D key points, the task of local and global pose estimation is equivalent up to a classic test-time optimization problem. It has been extended (Shimada et al., 2020; Xie et al., 2021) by including temporal physical simulations and constraints into this projection-based optimization loop. Rempe et al. (2021) use the same optimization loop with a motion VAE instead of a physics simulation. All of these methods need camera intrinsics and extrinsics to be known, which our work explicitly does not rely on. Further, all of these global pose works are based on some kind of test-time optimization; only ours is a purely one-shot regression.

4.2.3 Domain adaptation

Theoretically, training a deep model with fewer data could lead to worse generalisation performance (Fengxiang He, 2020; Arora et al., 2018). This problem becomes more prominent if there is insufficient data on the target domain. To overcome this, researchers proposed to reformalize pose estimation as a domain adaptation problem. Chen and Ramanan (2017) adapted the trained model to Human3.6M by fine-tuning in a supervised manner, whereas Chen et al. (2019) and Habekost et al. (2020) applied domain adversarial learning without using ground truth. Although these models can learn the dataset-specific intrinsics (i.e. same for training and testing), they may not be suitable for the task without a training set. Apart from this, some work such as Kanazawa et al. (2019); Kocabas et al. (2019); Rhodin et al. (2020) use unpaired or unlabeled data, they use it to complement the supervised learning on Human3.6M.

Here, we go one step further and train the system without collecting the training data or any testing data except for the specific example of interest from the target dataset. The system will be trained on a synthetic dataset and generalize to an unseen Human3.6M dataset. This makes our work unique from existing works.

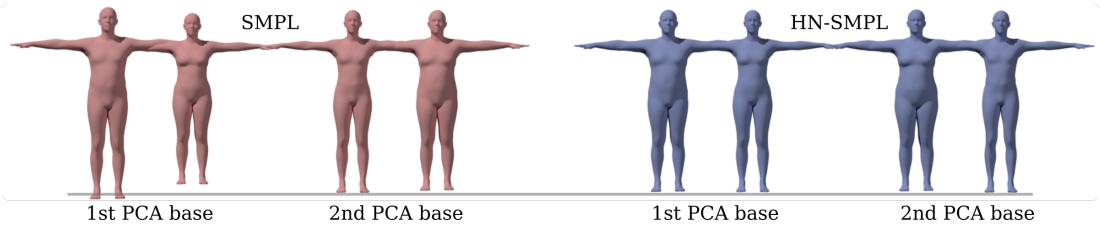


Figure 4.3: HN-SMPL (Height-Normalized-SMPL) expresses all body shape variation except total body height through the bases of a PCA. The difference between the SMPL and the HN-SMPL is shown. When the body shape is varied, the SMPL varies the total height of the person, while for all HN-SMPL body shape parameters, the total height is approximately constant. The pairs show $+1$ and -1 of the noted PCA basis. Note that even though less visible, height variation also does occur on other than the 1st SMPL PCA-base, here for example the 2nd is shown.

4.3 Synthetic Training Data Generation

4.3.1 Height Normalization

Due to the scale ambiguity of monocular video, it is impossible to estimate the height of a subject without reference or calibration (see figure 4.1A). We, therefore, assume that every subject has a height of 1.72m, which is the average human height implied by the SMPL model.

Note that conceptually we could choose any arbitrary height. This is similar to monocular SLAM (e.g. Mur-Artal and Tardós (2015), Mur-Artal and Tardós (2017)) or monocular structure from motion reconstruction (e.g. Ji et al. (2015)) methods that assume an arbitrary scale. The reason to pick 1.72m is that the space around the average SMPL height, exactly at $\beta = \mathbf{0}$, can be assumed to be the space where the model has the best support. Like in any linear gaussian model, the further we move away from the mean, the less well defined the model is.

We circumvent the same scaling ambiguity for translation estimates by this assumption. Consequentially we predict the subject's body-height-normalized body shape and the subject's body-height-normalized translation. If the subject's real body height is known, the normalized translation can easily be scaled by the ratio between known and normalized body height to obtain a translation that adheres to scale.

The PCA-based SMPL body shape space implicitly and non-trivially embeds the total height. We propose a similar body shape space but with a fixed height. We sample the SMPL with random body shapes and T-pose angles, and normalize the vertices by

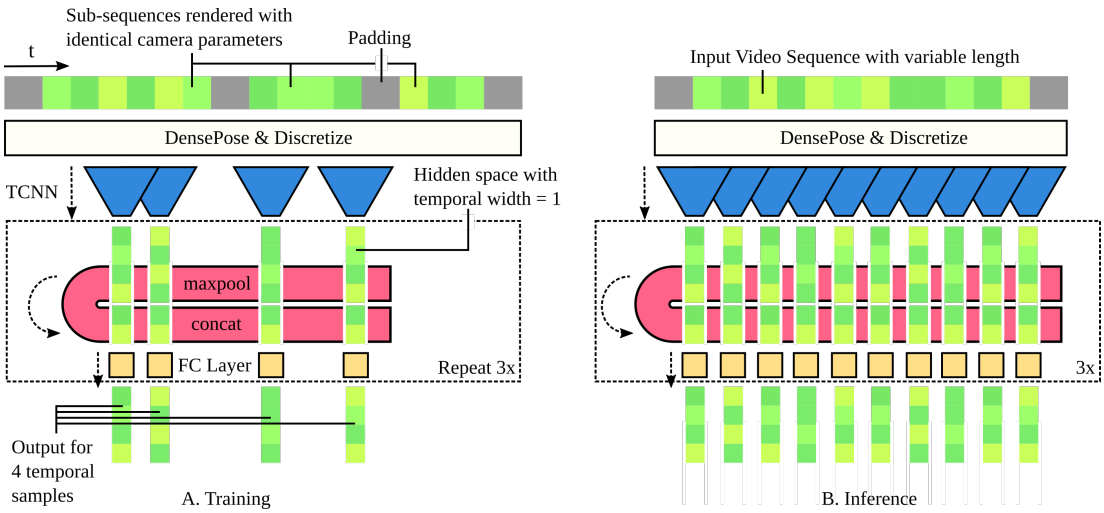


Figure 4.4: The neural network regression model in detail. Input, output and hidden space are depicted in green. Processing steps and network functions have black borders around them. Similarly shaped and coloured network functions share their weights. The forward pass direction is depicted through dotted lines.

scaling the height difference between the top and bottom vertices to 1.72m. These samples of scaled SMPL meshes are used to fit a new PCA, which then embeds a body space that is independent of the absolute body height (see Figure 4.3). We only take the first eight principal components. To be able to map back to the SMPL shape space, we fit the SMPL model to our height-normalized body mesh by iterative gradient descent. We then learn a mapping from the new space to the SMPL body shape space with a simple neural network Θ with two layers and 16 hidden units each. To infer the mesh or joint positions from an HN-SMPL body shape β_{HN}, the SMPL body shape $\beta = \Theta(\beta^{HN})$ can be calculated.

4.3.2 Synthetic Sequence Rendering

For each video sequence S to render we sample a body shape $\beta_S^{HN} \sim \mathcal{N}(\mathbf{1}_{10}, \mathbf{0}_{10})$ and retarget all motions in S to this shape. We, therefore, randomly sample motion clips from the AMASS dataset. After retargeting, we randomly rotate the motion clip perpendicular to the ground plane and randomly translate it by $T_{xy} \sim \mathcal{N}(\mathbf{3}_2, \mathbf{0}_2)$ parallel to the ground plane. Each sequence has a uniquely random camera with a different pitch, roll, height and field of view. Only the yaw is kept such that the camera looks down the Z-axis because the camera's yaw is arbitrary and not inferable. Our random distributions cover the most reasonable settings to capture human performances, from

extremely close to exceptionally far away, from slightly up-looking to almost from above.

We simply concatenate the motion clips until a minimum of 60 seconds or 600 frames at 10fps is reached. Because we do not truncate sequences, the total length is variable and effectively a Poisson distribution starting at frame 600. Most sequences have a total length of around 700, but some rare examples of up to 2000 frame lengths exist.

Note that we do not transition between motion clips within a video sequence and account for this later in the model. If we only use long, continuous motion clips, the dataset size and variety would be reduced. Further, long motion clips often revolve around similar and repetitive actions, yielding global predictability that we want to avoid as our model could overfit it.

4.3.3 IUV Discretization

We choose IUV-Maps obtained with DensePose as an intermediate representation. This aims to leave the domain gap between simulated and real data to DensePose. This also reduces the data dimensionality while keeping most of the useful information. We further reduce and adapt it to our sequence model by discretizing the dense IUV maps. While the IUV maps can be interpreted as a dense superset of 2D human keypoints, the discretization can be seen as piercing out a discrete 2D keypoint subset of the dense IUV map. The advantage of this approach over directly taking a 2D keypoint detector is that we can control the amount and semantics of the keypoints. By making them plentiful and shape-defining we can preserve the implicit body shape information that is in the IUV map.

DensePose gives a three-channelled dense representation $D(x,y) \in \mathbb{N} \times \mathbb{R}^2$ containing the part index $D(x,y)_i$ with the highest posterior probability $P(i = p|x,y)$ and the part-specific projected U- and V-coordinates $D(x,y)_{uv}$ at the image pixels x,y. We partly heuristically, partly manually choose a set of 178 IUV coordinates based on their distance from each other and on their semantic value of the corresponding positions on the template mesh. For each predefined IUV-coordinate C we exhaustively search in the image's pixel space x,y. This results in a pixel coordinate (each, for 178 IUV co-

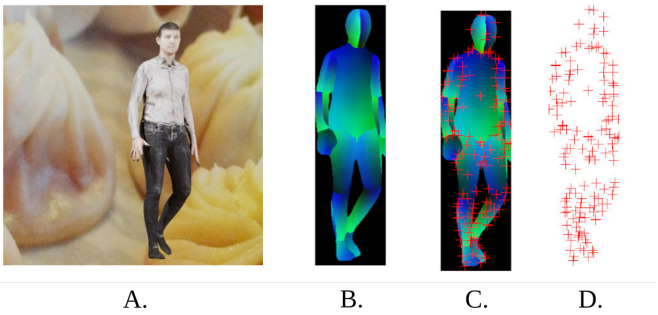


Figure 4.5: The process of UV discretization visualized. DensePose(Riza Alp Guler, 2018) gives a dense map of the SMPL template mesh UV coordinates in image space (B.) for an input image (A.). There are 178 discrete points (shown isolatedly in D.) in the final set. The positions of the 178 points on the template SMPL mesh (not shown here) have been manually selected carefully. After defining the template mesh positions, the discrete keypoints in image space can be derived from the UV projection which is shown by overlapping both in C.

ordinates) that is close to the coordinate of interest on the projected and parameterized 3D mesh without knowing the actual parameterization:

$$D(x,y)_i = C_i \tag{4.1}$$

$$d_{uv} = \|D(x,y)_{uv} - C_{uv}\|_2 \tag{4.2}$$

$$\min_{x,y} d_{uv}. \tag{4.3}$$

We do not specify a threshold for the part-specific UV-distance d_{uv}. As long as the part p is visible, all $C, C_i = p$ for that part will be found. As a result some C lie on the boundary of the part's segmentation mask, which gives a rough indication of the body part's shape.

4.4 Model

4.4.1 Network design

Similar to Habekost et al. (2020), our model Φ is composed of a temporal convolutional neural network (TCNN). The TCNN aggregates local features of a 32-frame-wide neighbourhood. Each 1D convolutional layer down-samples the temporal space by half with stride 2. Five convolutional layers are necessary to arrive at a hidden space with a temporal width of one. In figure 4.4, the max-pooling is applied on the hidden

units over the entire temporal channel so that the model can extract a sequence-level information flow. This method is inspired by PointNet (Qi et al., 2016) and allows scaling the number of temporal samples to an arbitrary length in a single regression. The max-pooling results are concatenated and a feed-forward layer is applied. These steps of max-pooling, concatenation, and fully-connected layer are repeated three times. The last fully-connected layer produces the model's output for every temporal step that the regression that has originally been applied. We use a fixed number of temporal neighbourhoods sampled from the sequence in training. In inference, our model can regress the whole input sequence in one shot.

4.4.2 Network loss

For each time step t, our network Φ maps the discretized IUV coordinates

$$c = \{\{c_{xi}, c_{yi}\}_1, ...\{c_{xi}, c_{yi}\}_{178}\} \in \mathbb{R}^{356} \tag{4.4}$$

to a HN-SMPL body shape $\beta^{HN} \in \mathbb{R}^8$, 24 joint angles $\theta \in \mathbb{R}^{72}$, a 3D translation $T \in \mathbb{R}^3$ and a camera view $V = \{pitch, roll, height, fov\} \in \mathbb{R}^4$ (see figure 4.2). The objective function consists of an adversarial loss, and a weighted (with scalar l_X) L1-loss between the synthetic ground truth components $X_t = \{\beta^{HN}, \theta, T, V\}$ and the network prediction $\Phi(c_t)$ of all time steps t:

$$\ell = \sum_t (l_X \cdot |X_t - \Phi(c_t)|_1) + L_{adversarial} \tag{4.5}$$

4.5 Experiments

We train our default model with 70k synthetic sequences, with a uniquely drawn set of body shapes, camera parameters and obstacles per sequences. For ablation experiments, we also train models without pooling or adversarial loss or neither. The method without neither pooling and adversarial loss is also called *fully connected* (FC) or *linear* model in the context of this evaluation.

First we evaluate on Human3.6M. This allows an extensive comparison of the local human pose quality with other methods. Note that this comparison is not always fair, as other methods supervisedly train on or adapt to Human3.6M. This implicitly overfits to the exact camera setup in Human3.6M. A performance drop can be expected when

other camera setups would be used for evaluation of these methods. How large this drop is is not possible to evaluate without creating a new Human3.6M-like ground truth evaluation-only dataset with a different camera setting. This is not within budget and scope for this work. But in contrast to supervised and domain adaptation methods, our method can be assumed to deliver the best possible performance independent of the camera setup, i.e. across all camera domains. As a qualitative evaluation, we show a birds-eye view of the global human translation on Human3.6M sequences. We plot the translation of different ablation versions to look at their influence on global human pose performance and on the visual quality of the translation trajectory.

Second, we evaluate the translation and camera parameter estimation performance for different ablation variants on our newly created CP dataset. This dataset features five subjects and 25 vastly different camera intrinsics and extrinsics, while the Human3.6M evaluation set only features two subjects, one single camera intrinsic and four very similar camera extrinsics. But we only obtain ground truth camera intrinsics, extrinsics and the ground truth global starting position of the subject. We do not obtain the ground truth human poses, neither local nor global. And we also have much less variation of motion than Human3.6M: only walking in circles. We use the CP dataset to investigate our method's capabilities of explicitly estimating the camera setup and also the translation error under these varied camera setups and more varied subjects than in the Human3.5M evaluation set. Note that the translation error is only compared for the first step for each sequence because the we only have the ground truth starting position.

For the camera setup evaluation on our CP dataset we also compare with general visual methods, at least where possible, i.e. for the camera pitch and field of view. For the camera height these methods lack any concept of scale. Our method also gives a height estimate, given our 1.72 m body size assumption, which we can retrospectively scale via the real subject's body height to the real camera height.

Finally, for the translation performance, we compare not only the ablation variants but also variants trained on different sizes of the synthetic dataset. We compare on both Human3.6M and our CP dataset. The aim here is to investigate how the dataset size influences the generalization error, i.e. if and how much more synthetic training data helps to better estimate real sequences.

Figure 4.6: Qualitative results of two Human3.6M test sequences. Both the pose and the translations are depicted. The camera is out of the rendered frame but its direction is shown with a black arrow.

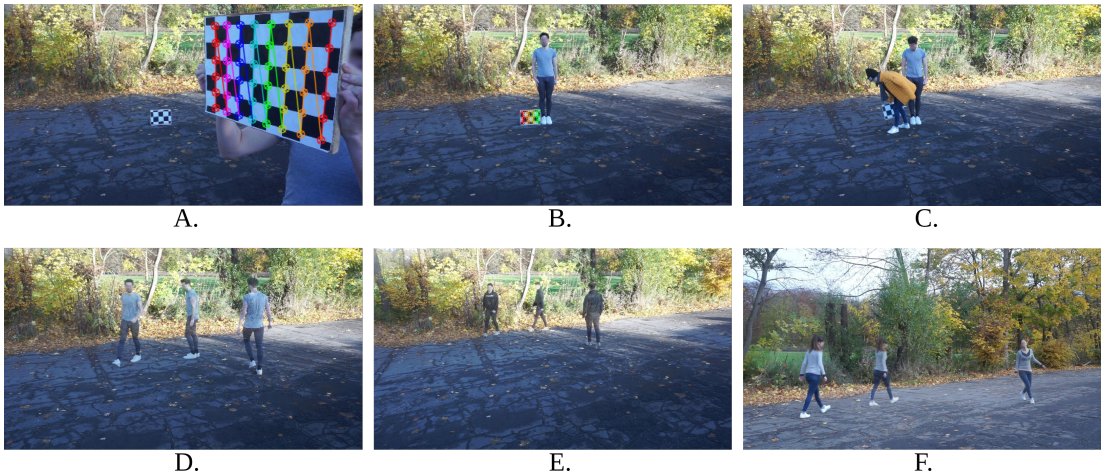


Figure 4.7: Our **CP** evaluation dataset. For different field of views, the camera is first calibrated (A). A marker on the floor (B) is used to infer the ground truth motion starting position.

4.5.1 Datasets

For evaluation, we use subjects 9 and 11 of **Human3.6M** (Ionescu et al., 2014). The other subjects are not used in this work. Note that the Human3.6M field of view is 46.4°. We report the mean per joint error (MPJE) and procrustes-aligned mean per joint error (PA-MPJE). Also see subsection 2.3.1 for further details on Human3.6M and its evaluation protocols.

We also make our own evaluation dataset where people walk in circles from different perspectives (**CP**, see figure 4.7). Each of the 25 sequences produced by one of the five different subjects has a different camera angle and focal length in a distribution similar to the synthetic training dataset. We only obtain the motion starting position as the ground truth and only report the translation error to this starting position. The

ground truth camera intrinsics are obtained with a classic checker board calibration and the subject's starting position and camera height are obtained with a marker positioned on the ground next to the subject's feet while standing in the initial I-pose (as can be seen in figure 4.7 B.). The subject then begins their walking sequence from that I-pose. We make sure to also vary the starting I-pose position across different sequences.

4.5.2 Training

Our synthetic training dataset uses motion sequences from **AMASS** (Mahmood et al., 2019) rendered with textures from **SURREAL** (Varol et al., 2017) and random CC0-licensed images as background. The horizontal field of view is drawn from $[80°, 34°]$ for each sequence. We chose square 1000×1000 pixel images, hence the vertical field of view is the same.

We train on synthetic dataset sizes of 10k, 20k and 70k sequences until the performance of the identical 1k validation sequences cannot be further improved. Note that each sequence is produced with a uniquely, randomly drawn HN-SMPL body shape, set of camera parameters and set of obstacles. We report our model's results with the 70k training sample size, except when stated otherwise. In training, we sample 16 temporal neighbourhoods from each sequence. The neural network has 6 linear layers with 2048 hidden units and concatenates 512 max-pooled hidden units, Adam (Loshchilov and Hutter, 2017) as the optimizer and a learning rate of $1e^{-4}$. We set $l_\theta = 1000, l_\beta = 10, l_T = 100$ and $l_V = 1$. We report results with the 70k training set model, except when stated otherwise.

4.5.3 Human3.6M Local Pose Results

Table 4.1 shows our local pose performance on Human3.6M. We show both the default MPJPE and the Procrustes-aligned MPJPE (see subsection 2.3.1) and sort the results by the default one. The default MPJPE is the harder task as the camera angle needs to be estimated correctly, whereas the PA-MPJPE corrects for an incorrect rotation.

We are only beaten by methods that integrate supervised training on Human3.6M training data into their model. The only exception is for the PA-MPJPE of Kanazawa et al. (2018). Besides, we beat all other domain adaptation methods on Human3.6M. This result is reasonable. The supervised learning approaches learn the dataset-specific camera intrinsics and extrinsics and have no domain gap, whereas our approach does

Table 4.1: Evaluation results on Human3.6M sorted by MPJPE. Domain adaptation (DA) methods use Human3.6M training videos together with non-GT-based supervision. Supervised (S) papers directly train on the Human3.6M training set. Methods with the implicit cameras have seen the Human3.6M cameras, which are identical for training and testing. Domain generalization (DG) methods have not seen any images, cameras or motion from Human3.6M.

Method	**MPJPE**↓	**PA-MPJPE**(↓)	**Task**	**Camera**
Chen and Ramanan (2017)	136.1	-	DA	implicit
Chen et al. (2019)	-	68.0	DA	implicit
Kanazawa et al. (2018) unpaired	106.8	66.5	DA	implicit
Shimada et al. (2020)	97.4	65.1	S	given
Kanazawa et al. (2018) paired	88.0	56.8	S	implicit
Ours	87.9	66.9	**DG**	**unseen**
Rhodin et al. (2020)	66.8	51.6	S	implicit
Kocabas et al. (2019)	65.6	41.4	S	implicit
Martinez et al. (2017)	62.9	47.7	S	implicit

not have domain knowledge with respect to the camera. Interestingly we even beat some supervised methods for the default MPJPE despite this obvious disadvantage.

4.5.4 Human 3.6M Birds-Eye View

Figure 4.8 shows a qualitative evaluation of the translation. Interestingly the model with only TCNN and fully connected layers is already close to the ground truth translation trajectory, albeit with a lot of jitter. To produce a more stable, visually pleasing motion we had to introduce adversarial training. We can see that the adversarial network pushes the model towards a more stable, smoother and more realistic overall motion shape. But without global pooling, the adversarial network's regime seems too strong; the model cannot infer the overall motion shape with only local temporal knowledge and collapses into a smaller motion.

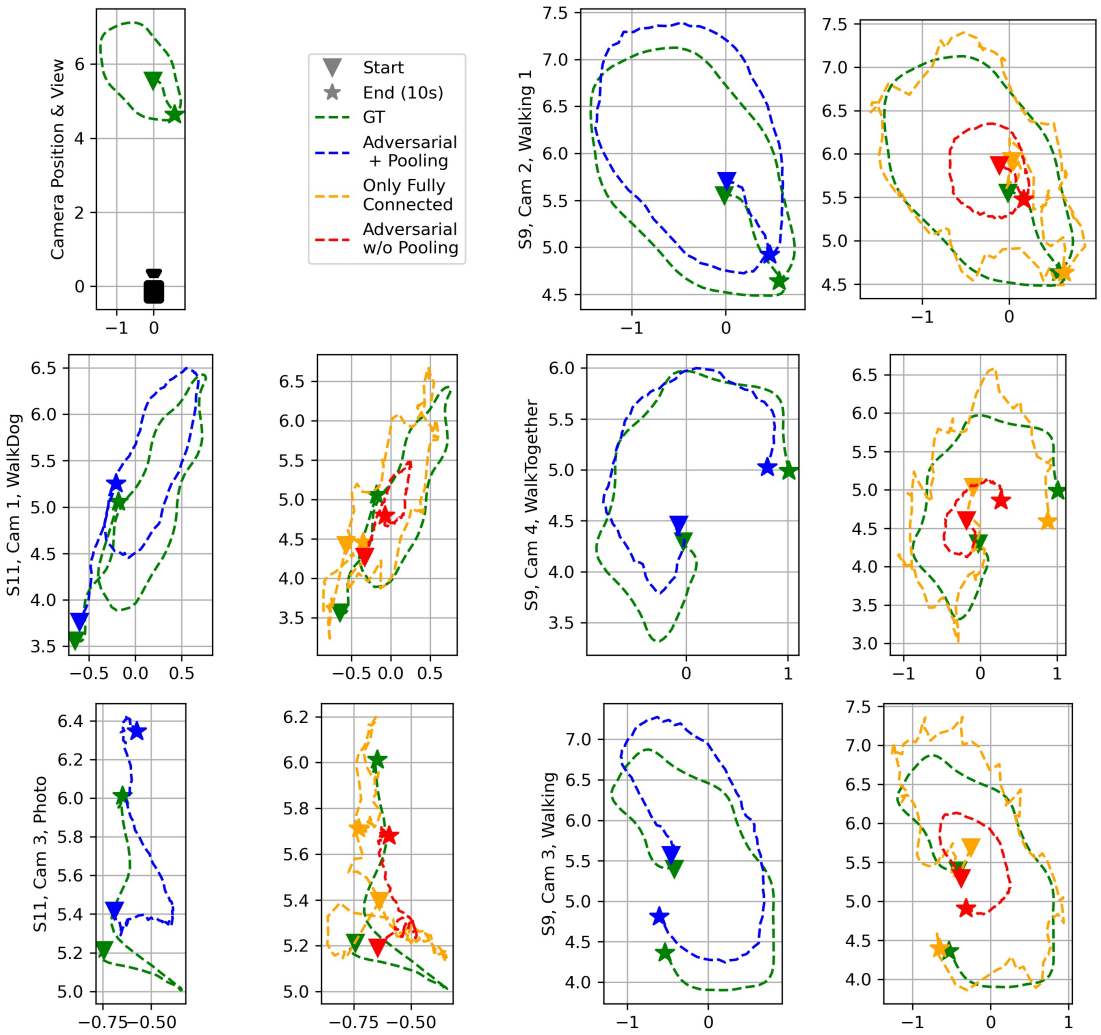


Figure 4.8: Translation results on the first 10 seconds of Human3.6M sequences viewed from above. After the overview and legend, the two plots next to each other belong to the same sequence.

4.5.5 Human3.6M & CP Translation Results

As shown in the table 4.2, the fully connected layer with the TCNN seems to deliver a slightly better performance than our adversarial approach on the 70k dataset. The gap becomes more extensive when a smaller training dataset is used, which is also reflected in the differences between training and validation data. This gap cannot be explained via a domain gap but simply because the training set is too small for enabling the model to learn the synthetic domain perfectly and not overfit to the training set. This is the reason why we believe there is still room for improving the proposed method's translation accuracy by generating an even larger dataset.

Table 4.2: Translation errors (in mm, non-procrustes) for different training set sizes. Both train and val are synthetic datasets.

	10k			
Method	**train**	**val**	**H36**	**CP**
Only FC	248	482	500	843
Advers. + Pool.	308	694	915	1072
	20k			
Method	**train**	**val**	**H36**	**CP**
Only FC	293	395	426	683
Advers. + Pool.	323	476	721	882
	70k			
Method	**train**	**val**	**H36**	**CP**
Only FC	294	380	406	531
Advers. + Pool.	315	414	421	539

There are methods (Shimada et al., 2020; Xie et al., 2021) that report the Human3.6M translation error we report in table 4.2 and they are significantly better. This is simply due to the methods assuming known camera parameters. But when camera parameters are unknown, like for the CP dataset, these do not work. Hence it is impossible to infer the subject's translation in our CP dataset, which we also report in table 4.2. We argue that implicit camera estimation is half of the work necessary for our model to infer the subject's translation and a comparison would be unfair.

4.5.6 CP Camera Results

Table 4.3 shows our camera estimation results compared to other deep learning methods on image data. We do not expect to outperform these models but merely show that the results are sensible and can be regressed from only 2D human motion. Also, only our method can infer the camera's height (inversely scaled to the human's size in the video). Note that our method can regress the camera parameters in one shot. We show this to convince our readers that our model can implicitly learn the camera estimation necessary for global pose estimation.

Table 4.3: Errors of camera estimation on the CP dataset, compared to methods that use images for camera parameter estimation.

Method	**FoV (°)**	**Pitch (°)**	**Height (mm)**
ScaleNet (Zhu et al., 2020)	3.63	2.11	-
CamCalib (Kocabas et al., 2021)	3.14	1.80	-
Ours (linear)	3.84	2.95	32.8
Ours (adversarial + pooling)	3.92	3.21	36.1

4.6 Conclusion

By using a large-scale synthetic dataset coupled with DensePose (Riza Alp Guler, 2018) as a 2D human motion extractor and a one-shot regression network that encourages global information flow, we show that human performance capture is possible without externally calibrated cameras or visual understanding of the surroundings. We demonstrate that the camera's intrinsics and extrinsics can even be estimated explicitly. To our knowledge, by doing so we have created the first domain generalization approach for the popular Human3.6M (Ionescu et al., 2014) dataset. Our result is comparable with some supervised learning approaches and outperforms all domain adaptation-based methods on Human3.6M's local pose performance.

Chapter 5

Unknown, Moving Camera: A cVAE for Terrain and Visual Odometry

This chapter is based on yet unpublished work with co-contributors Takaaki Shiratori and Taku Komura. The work has been proposed and conducted mainly by myself, exclusively implemented by myself, with original ideas mostly from myself. Co-contributors assumed consulting roles, contributed ideas and helped researching related work.

5.1 Introduction

To close in towards our goal of estimating human motion from arbitrary videos, we need to be able to handle moving cameras with unknown intrinsics and terrain that deviates from the flat ground plane assumption. In this chapter we propose a novel method that to the best of our knowledge is the first one to address this specific setting. We assume continuous videos (without ”cuts”), no major occlusion or out-of-frame motion and constant focal length (no ”zoom”).

Camera motion and human motion are linked via a reprojection constraint (Yuan et al., 2021). We exploit this to fuse test-time inference on a global motion model with simple image feature visual odometry.

The global motion model is based on the HuMor (Rempe et al., 2021) conditional variational autoencoder (cVAE), which autoregressively predicts a distribution of next poses from the previous pose. We amend the cVAE to allow for z-direction traversal of the human subject and free traversal of the camera. We also contribute a step plane

estimation which bundles together steps on the same height and which we show to improve performance under certain conditions.

To estimate the camera pose, we use SIFT feature matching and RANSAC with the eight point algorithm (Chojnacki et al., 2003). We deliberately chose a simple image feature visual odometry method to fuse into our system. Big, complex state of the art visual odometry or SLAM systems are out of scope for our work. One reason is that complex, established methods do not allow us to work with a missing or bad camera intrinsics initialization without deconstructing, re-thinking and re-implementing the system.

Note that our motion cVAE as an isolated module already yields an estimate of the camera ego-motion and additionally of the camera intrinsics. We use this to force a convergence between the motion cVAE and the feature matching odometry by biasing both ego-motion results towards the respective other result. We iteratively and alternatingly optimize the cVAE and calculate the feature odometry with growing biases until convergence.

We are the first to show that combined test-time optimisation of global human motion on arbitrary terrain, camera extrinsics and intrinsics can enhance simple feature matching based visual odometry. This means that the learned neural prior on arbitrary-terrain human motion can also act as a prior on camera motion and intrinsics.

We also record the PFST-Handheld dataset for systematically analysing our method, specifically but not exclusively with different types of camera motion. The dataset uses a marker based motion capture system while we also track and synchronize the motion of a handheld camera.

5.2 Related Work

5.2.1 Local pose estimation

This is the task of estimating a human's pose rotated relative to the camera and with the hip centred at the origin. Martinez et al. (2017) regress 2D keypoint detections (Newell et al., 2016; Cao et al., 2017) to 3D joint positions. Zhou et al. (2017) directly incorporate a depth regression into a formerly 2D-only keypoint regressor. They use an adversarial pose prior to when only 2D supervision is available. Kanazawa et al. (2018) estimate both 3D pose and body shape based on SMPL (Loper et al., 2015), a parametric model of human shape and pose. Pavllo et al. (2018) introduce a temporal model

with dilated convolutions. We are interested in the harder task of estimating global 3D poses where the subject is rotated with respect to the global ground plane or gravity.

5.2.2 Global pose estimation with known Camera

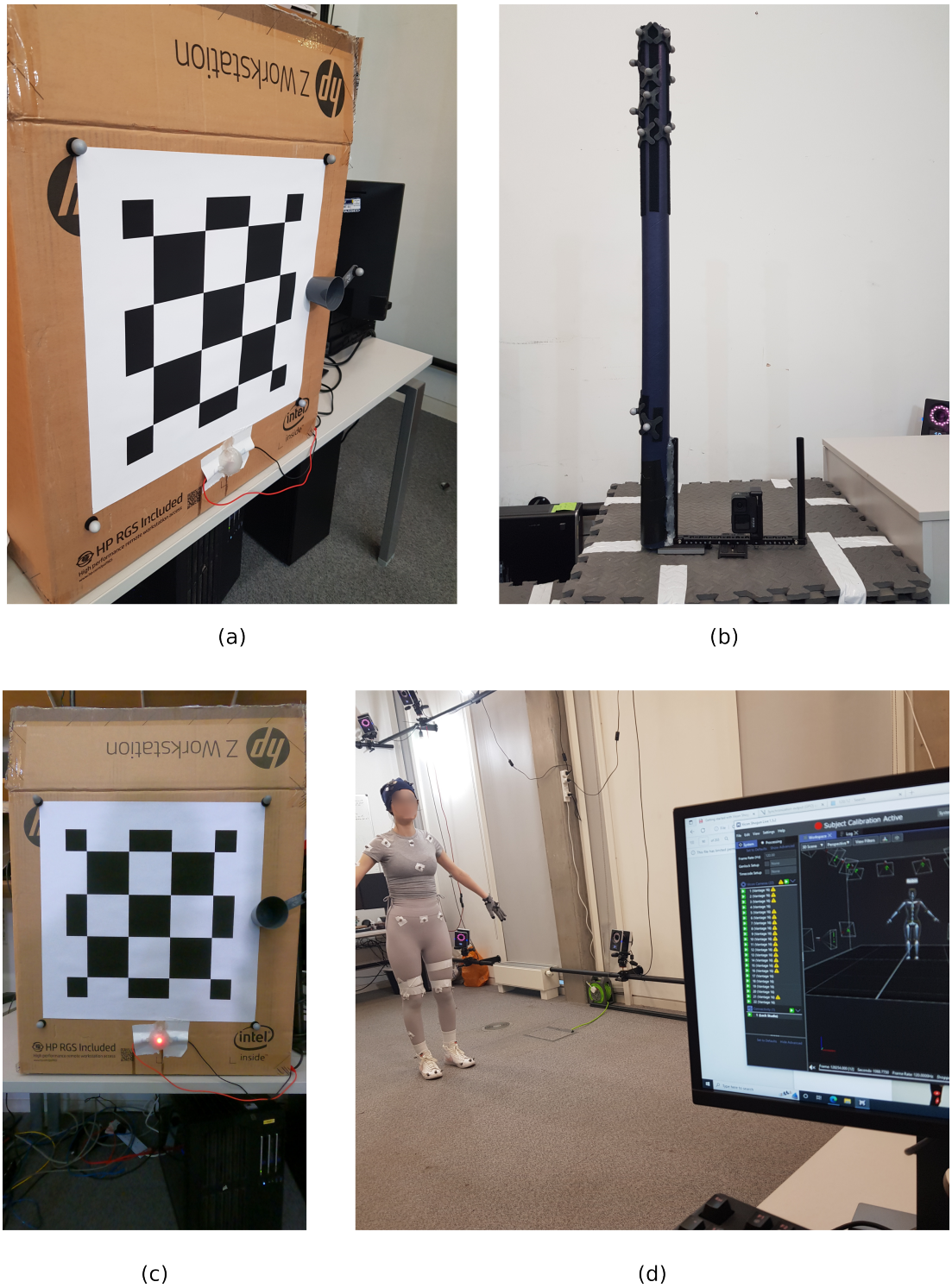
Here the estimated human poses are rotated and translated relative to the world's ground plane. Mehta et al. (2017b) use procrustes alignment style post-processing on per-frame estimations with a given camera to obtain a temporally consistent real-time global pose from a local pose estimator. Shimada et al. (2020) establish a baseline by optimizing 2D and 3D key points from a local pose estimator (Kanazawa et al., 2019) to match a reprojection loss given the camera extrinsics and intrinsics. This shows that with the known camera and 2D key points, the task of local and global pose estimation is equivalent up to a classic test-time optimization problem. It has been extended (Shimada et al., 2020; Xie et al., 2021) by including temporal physical simulations and constraints into this projection-based optimization loop. Habekost et al. (2020) train a neural-adversarial model to place the subject in a scenario with given camera and ground plane. All of these methods need camera intrinsics and extrinsics to be known, which our work explicitly does not rely on.

5.2.3 Global pose estimation with (partially) unknown Camera

Rempe et al. (2021); Yuan et al. (2021) use a motion conditional variational autoencoder (cVAE) in a test-time optimisation loop as a learned motion prior for global motion. Similar to our work, the cVAE predicts the next step from the last one. Rempe et al. (2021) assume a given field-of-view and a fixed camera while Yuan et al. (2021) optimize for both the extrinsics and intrinsics of the moving camera. Both Rempe et al. (2021); Yuan et al. (2021) assume a flat ground plane. Our work can handle arbitrary terrain.

5.3 PFST-Handheld Dataset

We introduce **PFST-Handheld**, a dataset with ground truth human motion over obstacles and synchronized ground truth camera motion. The PFST acronym stands for the variables within the dataset: camera **path** (see Figure 5.2 and Figure 5.3), vertical camera **field-of-view** (115°, 103°, 90°, 71°), **subjects** (male and female) and **terrain** (obstacles, flat). Further, to add variability, the obstacle sequences were recorded in



(a) (b)

(c) (d)

Figure 5.1: PFST-Handheld dataset technical setup: (a) A synchronisation box is made out of VICON markers, a calibration checkerboard and a red LED connected to the VICON system. (b) The handheld rig includes the GoPro HERO9 and VICON markers. (c) The red LED is used for synchronisation of the handheld camera with the VICON motion capture system.

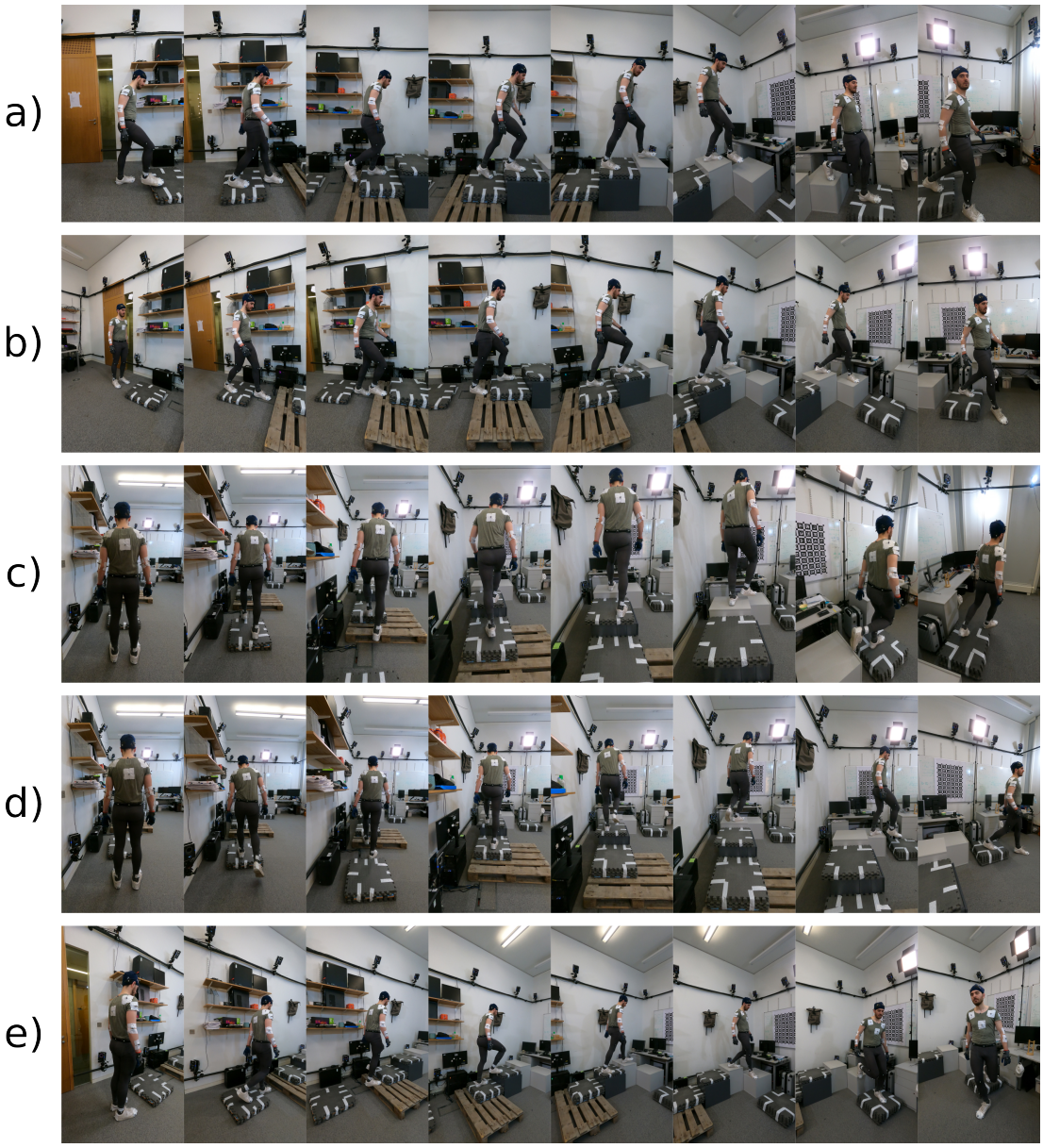


Figure 5.2: **PFST-Handheld Dataset**: Different camera path for obstacle sequences recorded. Each camera path has been recorded with four field-of-views and two subjects. Further, every obstacle sequence exists as a single-step and double-step version. a) 90° lateral, b) radial, c) follow fast, d) follow slow fade, e) 45° lateral

Figure 5.3: **PFST-Handheld Dataset**: Different camera paths for flat ground sequences recorded. Each camera path type has been recorded with four field-of-views and two subjects. f) 90° lateral, g) radial, h) follow fast, i) follow slow fade

a single-step version (one foot on each obstacle) and a double-step version (both feet always arrive at the obstacle). The handheld video is recorded in 1080x1920 pixel in portrait mode and 50 fps with a Go-Pro HERO9 (see Figure 5.1b).

5.3.1 Motion Capture

We use a VICON marker-based motion capture system to track 3D human motion, the camera path and the position of the synchronisation box (see Figure 5.1). We use the MoSh approach (Loper et al., 2014) and fit virtual markers placed approximately on the SMPL model mesh to the recorded body marker positions. The body pose angles θ are optimized for each time step and the body shape β and the virtual marker's placement are optimized for the subject's total sequence. Loper et al. (2014) shows that increasing the amounts of markers increases the estimation quality of the body shape β. Since we use normal clothing with markers taped to the body instead of the VICON black velcro suit, we aim to have a small number of markers; therefore we use the VICON default *FrontWaist* marker set. To compensate for this in the estimation process of the body shape β, we fit each subject's β-T-mesh to a 3D body scan of the subject. We scan the subject in tight clothes with a 3D body scanner with 24 RGB cameras and 12 IR cameras, resulting in a textured mesh of the subject's body.

5.3.2 Synchronised Handheld Video

We build a camera rig to hold the Go-Pro and to attach the infrared markers to (see Figure 5.1b). We use the different lens modi and sample down 4K 100fps footage. The Go-Pro needs to be calibrated spatially and synchronised temporally. Therefore, we build a dedicated box (Figure 5.1a and Figure 5.1c) for synchronisation and calibration of the Go-Pro.

The spatial calibration is achieved via IR markers on the box and a checkerboard pattern on the box. The IR markers allow the VICON system to identify and track the box's position. The checkerboard allows the camera to find the box in its coordinate system. The transformation between the markers and the checkerboard is calculated by taking a few pictures from a few different angles to then manually label the marker positions in 2D. Given known extrinsics, intrinsics and 2D marker positions on a set of images, simple least-squares can then calculate the approximate 3D positions of the markers.

The temporal synchronisation is achieved with a red LED connected to the VICON Lock Studio. The diffused red LED blinks every nth frame as configured in the VICON system. It is spatially aligned to the checkerboard via the same method as the IR markers. When the checkerboard is visible and the intrinsics known, we can infer the LED's projected 2D position. We apply a 2D gaussian kernel and then a temporal high-pass filter on the spatially averaged red channel to obtain the frames in which the LED blinks.

5.4 Motion cVAE with Z-Traversal and Step Plane Estimation

We train a conditional variational autoencoder (Kingma and Welling, 2014) (*cVAE*) to encode human motion similar to Rempe et al. (2021). Figure 5.4 shows how the motion cVAE is used to estimate global human motion and camera motion. The motion cVAE is trained on the AMASS (Mahmood et al., 2019) large scale motion capture dataset to encode a distribution of next egocentric poses $\hat{X}_t$ based on the conditioning of the subject's previous egocentric pose X_{t-1}. The encoder $e(X_{t-1}, X_t)$ yields the parameters to a gaussian $\{\mu_Z \in \mathbb{R}^{24}, \sigma_Z \in \mathbb{R}^{24}\}$ from which a latent code Z_t can be sampled, i.e. $Z_t \sim \mathcal{N}(\mu_Z, \sigma_Z)$. The decoder $d(X_{t-1}, Z_t) = \hat{X}_t$ decodes the next pose based on a latent code sample Z_t and the previous pose. In inference we discard the encoder and obtain a sequence of egocentrized human poses $\hat{X}_1, ..., \hat{X}_I$ by chaining decoders together:

$$\hat{X}_1, ..., \hat{X}_t = d(d(d(..., Z_{t-2}), Z_{t-1}), Z_t) \tag{5.1}$$

5.4.1 Differentiable Motion Egocentrization with Z-Traversal

We expand upon the motivation that a straight walk is the same motion irrespective of its global rotation to up/down traversal, since walking down the stairs is the same motion irrespective of the floor. Similar to the work in chapter 3 we need to bring the SMPL-based global motion representation $\{\beta, \theta, \mathbf{t}\}_t$ (body shape vector $\beta \in \mathbb{R}^{10}$, pose axis-angles $\theta \in \mathbb{R}^{24\times 3}$, and 3D global position $\mathbf{t} \in \mathbb{R}^3$) into a much more efficient representation relative to the subject's egocentric coordinate system (which we called egocentrization in 3.3.3). Contrary to chapter 3, we do not keep the subject fixed to

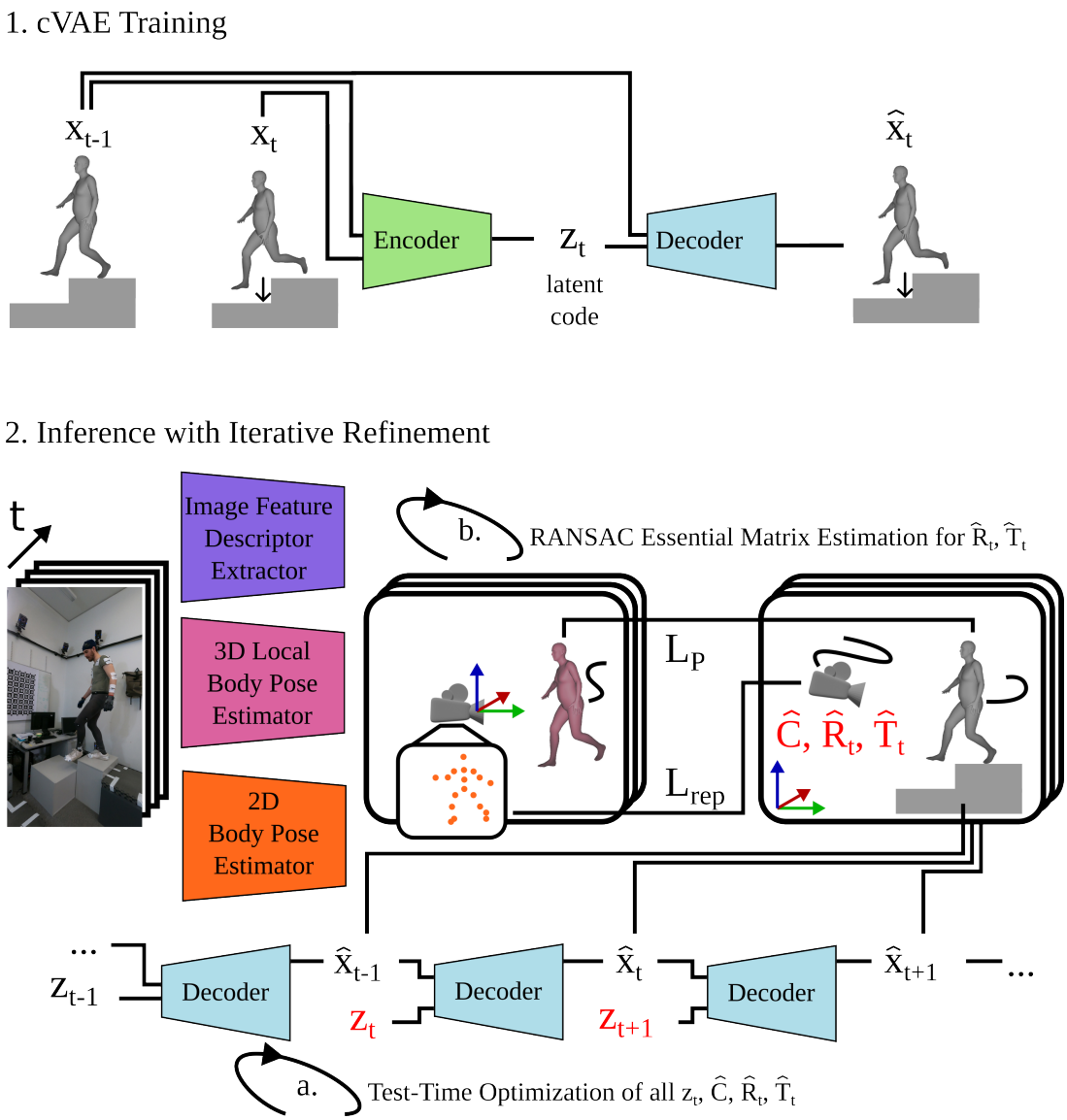


Figure 5.4: 1. Training of the cVAE based on human motion capture data. The cVAE is trained with height traversal and step plane estimation. 2. Inference optimization loop with iterative refinement. Based on a differential projection, the cVAE human motion and camera motion are optimized to fit observation from a 3D and a 2D body body estimator. The camera motion is also calculated with a classic RANSAC/SIFT approach, which is iteratively updated based on the camera evidence from the cVAE optimization, which is then fed back to the cVAE optimization, and so on.

a single ground plane. Instead we allow free traversal in Z-Direction (up/down). It follows that we differentialize $\mathbf{t}$ in all three dimensions.

We also aim at keeping the egocentric representation as close as possible to the original SMPL representation, which makes it easier to optimize it against other targets using the vanilla SMPL representation. Hence we exploit the fact that the pose axis-angles θ represent a kinematic chain and only the first rotation is relevant for the global rotation of the subject.

Another difference to the egocentrization process in chapter 3 is that we directly work in the egocentrized space. For calculating and projecting the absolute motion we need the differentiable de-egocentrization Φ', i.e. the forward integration of egocentrized poses. In this chapter we assume a motion sequence $X_0, ..., X_I$ to be in egocentric space for $i > 0$ and in absolute space for $i = 0$. This makes X_0 an ordinary set of SMPL-parameters $\{\beta, \theta, \mathbf{t}\}_0$ and $X_i, \forall i > 0$ a set of $\{\dot{\phi}_z, \theta_1^{pitch}, \theta_{2:24}, \mathbf{t}^{\dot{ego}}\}$, following the notation in section 3.3.3. The differentiable de-egocentrization Φ' maps this sequence to a absolute motion sequence via forward integration:

$$\Phi' : \{\beta, \theta, \mathbf{t}\}_0, \{\dot{\phi}_z, \theta_1^{pitch}, \theta_{2:24}, \mathbf{t}^{\dot{ego}}\} \mapsto \{\beta, \theta, \mathbf{t}\}_{0,...,i} \tag{5.2}$$

$$\Phi'(X_0, ..., X_i) = \sum_i X_i \tag{5.3}$$

The following listing 5.1 gives a simple and practical implementation of Φ':

```
def deegocentrize_smpl_matrix(sequence_length,
                              rot_velocities,
                              trans_velocities,
                              root_rotations_z_aligned,
                              initial_rotation=tf.eye(3)):

    rot_shape = rot_velocities.shape
    R_rotvel = tf.reshape(tf.transpose(tf.stack([
        tf.cos(-rot_velocities), -tf.sin(-rot_velocities), tf.zeros(rot_shape),
        tf.sin(-rot_velocities), tf.cos(-rot_velocities), tf.zeros(rot_shape),
        tf.zeros(rot_shape), tf.zeros(rot_shape), tf.ones(rot_shape)
    ])), [-1, 3, 3])

    Ri_recon = [initial_rotation * R_rotvel[0]]
    trans_recon = [tf.transpose(Ri_recon[-1]) @ trans_velocities[0, :, None]]
    for i in range(sequence_length-1):
        Ri_recon.append(R_rotvel[i + 1] @ Ri_recon[-1])
        trans_recon.append(
            trans_recon[-1]
            + tf.transpose(Ri_recon[-1]) @ trans_velocities[i + 1, :, None]
        )
    Ri_recon = tf.stack(Ri_recon)
    Rs_0_recon = tf.transpose(Ri_recon, [0, 2, 1]) @ root_rotations_z_aligned

    return (trans_recon, Rs_0_recon)
```

Listing 5.1: Implementation of Φ'

5.4.2 Step Plane Estimation

In comparison to prior work that only operates on the same ground plane (Habekost et al., 2020; Rempe et al., 2021; Habekost et al., 2022), our height traversal introduces a systematic instability with respect to the height of the subject. Without intervention, the open loop nature of forward integration of velocities, here the height velocity summed up to an absolute height, will cause a drift in the absolute value. To compensate for this we utilize the observation that foot contacts on the same plane will have the same absolute height. This is exploited by bundling together and enforcing height-alignedness in the test time optimization for steps that are estimated to lie on the same ground plane by the cVAE.

We do this by encoding a step plane prediction in the cVAE encoder. Alongside each X_i pose, the step plane prediction $S_i \in \{1,0\}$ is a binary classification variable addressing whether the next foot contact will be made on the same xy-plane as where the subject is currently on, i.e. as where the last or current foot contact is placed on. The next foot contact $\kappa(i)$ is defined as the next frame after i in which either foot's velocity $\dot{J}_i^{lFoot}, \dot{J}_i^{rFoot} \in \mathbb{R}$ in absolute space goes below a certain threshold $c_{fc} \in \mathbb{R}$ for the first and only for the first time:

$$J_i^{joint} := [\delta_{joint=1},\ \ldots,\ \delta_{joint=24}] \cdot J(V(\Phi'(X_i)) \quad \implies J_i^{joint} \in \mathbb{R}^3 \tag{5.4}$$

$$\dot{J}_i^{joint} := \|J_{i+1}^{joint} - J_i^{joint}\|_2 \quad \implies \dot{J}_i^{joint} \in \mathbb{R} \tag{5.5}$$

$$\kappa(i) = \begin{cases} i+1 & \text{if } \dot{J}_{i-1}^{foot} > c_{fc} \wedge J_i^{foot} < c_{fc},\ \exists foot \in \{\text{lFoot},\ \text{rFoot}\} \\ \kappa(i+1) & \text{otherwise} \end{cases} \tag{5.6}$$

Note that above we first define $J_i^{joint} \in \mathbb{R}^3$, a way to access individual joint values. We form a selection vector via the Kronecker delta $\delta_{a=b}$ which is $\delta_{a=b} = 1$ if $a = b$ respective $\delta_{a=b} = 0$ if $a \neq b$. The selection vector's components are always all zero except at the position to select, where it is one. We apply the selection vector to the SMPL joint keypoint regressor $J(V(\beta, \theta, \mathbf{t}))$ via an inner product. See subsection 2.3.2 for details on J and V and note that the selection vector formulation is just for mathematical clarity, in our python implementation we use simple indexing on the joint regressor.

We then bundle subsequent foot contact frames $j \in \mathbb{N}$ on the same plane in the set $j \in \mathbb{P}(i)$ if their $S_{\kappa(\kappa(\ldots i \ldots))} = 1$ yield an unbroken streak. To avoid duplicates of same-

plane contact streaks $\mathbb{P}(i)$ when iterating over i, we only use the last frame with $S_i = 0$ before the streak starts as an anchor and define $\mathbb{P}(i)$ as empty for all other frames:

$$\mathbb{P}(i) = \{\} \ \forall i: \ \kappa(i) \neq i+1 \ \vee \ S_i \neq 0, \quad \text{otherwise:} \tag{5.7}$$

$$\mathbb{P}(i) = \left\{ j \ \middle| \ \begin{array}{lll} j = \kappa(i), & S_i = 0, & \\ j = \kappa(\kappa(i)), & S_i = 0, & S_{\kappa(i)} = 1, \\ ..., & & \\ j = \kappa(\kappa(\kappa(...i...))), & S_i = 0, & S_{\kappa(i)}, ..., S_{\kappa(\kappa(...i...))} = 1 \end{array} \right\} \tag{5.8}$$

We also need to select the Z-coordinate of the foot that is in contact. Therefore we define $\kappa_L(i)$ and $\kappa_R(i)$ both to be one of $\{0,1\}$ if the left foot is or is not, respective right foot is or is not the one currently in contact. We use another selection vector $[0,0,1]^T$ to select the Z-coordinate of the joint in $J_{i,Z}^{joint}$. We combine this into $\hat{\mathbf{J}}_{i,Z}^{R/L}$, which is the Z-value of the foot in contact or zero if no foot is in contact:

$$\kappa_L(i) = \begin{cases} 1 & \text{if } J_{i-1}^{lFoot} > c_{fc} \ \wedge \ J_i^{lFoot} < c_{fc}, \\ 0 & \text{otherwise} \end{cases} \tag{5.9}$$

$$\kappa_R(i) = \begin{cases} 1 & \text{if } J_{i-1}^{rFoot} > c_{fc} \ \wedge \ J_i^{rFoot} < c_{fc}, \\ 0 & \text{otherwise} \end{cases} \tag{5.10}$$

$$J_{i,Z}^{joint} := J_i^{joint} \cdot [0,0,1]^T \quad \Longrightarrow J_{i,Z}^{joint} \in \mathbb{R} \tag{5.11}$$

$$\hat{\mathbf{J}}_{i,Z}^{R/L} := \kappa_L(j) \cdot J_{j,Z}^{lFoot} + \kappa_R(j) \cdot J_{j,Z}^{rFoot} \equiv \begin{cases} J_{j,Z}^{lFoot} & \text{if } \kappa_L(j) = 1 \\ J_{j,Z}^{rFoot} & \text{if } \kappa_R(j) = 1 \\ 0 & otherwise \end{cases} \tag{5.12}$$

Finally we define a loss L_S to optimize all foot contacts estimated to belonging to the same plane in the same streak $\mathbb{P}(i)$ belonging to a sequence with length I to converge to a plane contact streak specific average height:

$$L_S = \sum_{i}^{I} \sum_{j}^{j \in \mathbb{P}(i)} \left| \hat{\mathbf{J}}_{j,Z}^{R/L} - \sum_{j}^{k \in \mathbb{P}(i)} \frac{\hat{\mathbf{J}}_{k,Z}^{R/L}}{|\mathbb{P}(i)|} \right| \tag{5.13}$$

In our assumption and hence formulation the step plane is always parallel to the ground plane. This is why we characterize the plane only via the height or Z-coordinate

of the foot contact. In our optimization the loss L_S makes sure that foot contacts estimated to be on the same step plane by the cVAE encoder are actually on the same step plane in the resulting motion. The average height of a plane contact streak, i.e. the step plane height, is hereby not fixed but a result of the optimization process.

5.4.3 Global Camera and Human Motion via cVAE Optimization

Given a video sequence of a subject walking on terrain filmed with a handheld, moving camera with fixed focal length, we can use the motion cVAE to estimate the motion of the subject, motion of the camera and focal length of the camera. Therefor we apply OpenPose (Cao et al., 2018), an off-the-shelf 2D keypoint detector, and HMR (Kanazawa et al., 2018), an off-the-shelf 3D local pose estimator, to the input video. Note, that HMR delivers a 3D pose that is based on a weak projection within a bounding box of a 2D camera. The HMR 3D local poses are denoted $\hat{Y}_t^{Local3D}$ and follow the vanilla SMPL parameterization, while we denote the OpenPose keypoints $\hat{Y}_t^{2D}$. The camera is defined by an intrinsic matrix C, rotation R_t and translation T_t. We make the intrinsic matrix dependent on the unit focal length f, which is the focal length based on unit image with and height. We assume the unit focal length to be the same vertically and horizontally and a center point that lies in the exact center of the image pane:

$$C(f) = \begin{pmatrix} f \cdot w & 0 & \frac{w}{2} \\ 0 & f \cdot h & \frac{h}{2} \\ 0 & 0 & 1 \end{pmatrix} \tag{5.14}$$

The reprojection loss L_{rep} takes the de-egocentrized, absolute poses, projects them with the current best camera estimate and compares them to the OpenPose keypoints:

$$L_{rep} = \sum_t^I |\hat{Y}_t^{2D} - \langle C(f) \cdot (\Phi'(\hat{X}_0, ..., \hat{X}_I)t \cdot R'_t - T_t) \rangle_{hom \to 2D}| \tag{5.15}$$

$$\text{with } \langle [x, y, w]^T \rangle_{hom \to 2D} = [\frac{x}{w}, \frac{y}{w}]^T \tag{5.16}$$

The 3D local pose loss L_P aligns the joint rotations $\theta(\hat{X}_i)$ from the cVAE decoder chain with the joint rotations obtained from the HMR off-the-shelf local 3D pose estimator. We compare the rotations in $SO3$ space and transform the axis-angle representation into rotation matrices via the Rodrigues formula (Dai, 2015), here denoted $rg(\theta)$. The root rotation is compensated by the camera's current best rotation estimate:

$$L_P = \sum_{i}^{I}(|rg(\theta(\hat{Y}_i^{3D})_1) - rg(\theta(\hat{X}_i)_1) \cdot R_t'| + \sum_{j}^{2...24} |rg(\theta(\hat{Y}_i^{3D})_j) - rg(\theta(\hat{X}_i)_j)|) \quad (5.17)$$

We simply define a loss L_B that sets the body shape target to the average body shape from the HMR off-the-shelf estimator:

$$L_B = \sum_{i}^{I} |\beta(\hat{X}_i) - \sum_{j}^{I} \frac{\beta(\hat{Y}_j^{3D})}{|I|}| \quad (5.18)$$

The final combined loss L is the weighted sum of the partial losses:

$$L = a_B \cdot L_B + a_P \cdot L_P + a_{rep} \cdot L_{rep} + a_S \cdot L_S \quad (5.19)$$

We optimize this loss on the decoder chain of the cVAE. We use the Adam (Loshchilov and Hutter, 2017) optimizer in tensorflow, which does gradient descent with a momentum. We optimize the latent codes $Z_0, Z_t, ..., Z_I$, the unit focal length f, the camera rotation R_t and camera translation T_t. With this approach alone we already get an estimate of the subject's and the camera's motion while the subject is traversing terrain.

5.5 Bundling with Visual Odometry

We choose SIFT (Lowe, 2004) image feature matching and fundamental/essential matrix estimation with RANSAC (Fischler and Bolles, 1981), a very simplistic visual optometry method to support the camera ego-motion estimation. Note that through the reprojection constraint, this also helps the body motion estimation. We iteratively and alternatingly calculate the ego-motion via matching+RANSAC and cVAE optimization. We inject the camera ego-motion estimation from the motion cVAE optimization into the matching+RANSAC procedure and vice versa with growing biases to let both estimates converge.

5.5.1 SIFT based Fundamental Matrix and Biased RANSAC

We apply SIFT (Lowe, 2004) image feature matching. A SIFT feature is a image position and descriptor vector. If the descriptor vectors of two SIFT features in consecutive images are similar given a distance metric, the detected features might stem

from the same object corner or 3D landmark. We can use the normalized eight point algorithm (Chojnacki et al., 2003) to estimate the fundamental matrix F from eight matched features. The essential matrix E can be calculated with the help of the intrinsic matrix C (Hartley and Zisserman, 2004):

$$E = C'^{T} \cdot F \cdot C \tag{5.20}$$

Singular value decomposition on E yields the rotational and translational components in E, which is the translation and rotation between the two consecutive images scaled according to the intrinsic matrix.

The normalized eight-point algorithm gives a quality metric $d_{rep}(\{y,x\}_{i,\ldots,p})$ via reprojection consistency. This quality metric can be used with the RANSAC algorithm to obtain a result that is more robust along multiple sets of randomly drawn feature matches and more likely the correct solution. Since we want to exploit prior knowledge from the motion cVAE optimization result, we bias the quality metric used for RANSAC. For a rotation $\hat{R}_t^{8P}$ and a translation $\hat{T}_t^{8P}$ matrix obtained through the normalized eight-point algorithm, we simply add a distance to the cVAE counterparts $\hat{R}_t^{cVAE}, \hat{T}_t^{cVAE}$ and weight it with an iteratively growing bias factor b_{cVAE} and a constant internal factor $s_{\frac{R}{T}}$

$$\{x\} := \{y,x\}_{i,\ldots,p} \tag{5.21}$$

$$d_{cVAE}(\{x\}) = s_{\frac{R}{T}} \cdot |\hat{R}_t^{8P}(\{x\}) - \hat{R}_t^{cVAE}| + |\hat{T}_t^{8P}(\{x\}) - \hat{T}_t^{cVAE}| \tag{5.22}$$

$$d_{RANSAC}(\{x\}) = d_{rep}(\{x\}) + b_{cVAE} \cdot d_{cVAE}(\{x\}) \tag{5.23}$$

5.5.2 Inference with Iterative Refinement

We now iteratively and alternatingly optimize the cVAE and calculate the matching+RANSAC odometry. To allow for a convergence from both sides, we add the same growing bias loss to the cVAE's loss as we did for matching+RANSAC. $\hat{R}_t^{8P}, \hat{T}_t^{8P}$ are supplied from the previous biased matching+RANSAC calculation and $\hat{R}_t^{cVAE}, \hat{T}_t^{cVAE}$ are a differentiable result of the cVAE decoder chain that is subject to optimization. The loss is added and scaled with a bias factor b_{RANSAC}.

We start with motion cVAE optimization followed by matchin+RANSAC. We increase the bias factors b_{cVAE}, b_{RANSAC}. We then repeat this until convergence. We show the effect of different bias scheduling schemes in our evaluation results.

5.6 Evaluation

First, we investigate the contribution of our method's individual components to both scenes with obstacles and flat scenes. For this we build different versions of our method, starting with one that closely resembles the HuMoR (Rempe et al., 2021) cVAE, and then adding height traversal for obstacles, step plane estimation and finally the matching+RANSAC optimization schedule. The last method with all components represents the complete method.

Then we investigate different bias scheduling schemes for our complete method. In the complete method two different camera paths are alternatingly optimized with two different objectives: the cVAE objective and the matching+RANSAC objective. We show that combining both procedures with our iterative refinement and bias scheduling approach results in a better camera path estimation performance than both methods individually do. Without iterative refinement and bias scheduling, we do not expect these two camera paths to converge to the same solution because no information is shared between them. The biases in both optimization targets slowly and with each iteration increasingly pull both camera paths towards each other. The speed at which we increase the biases are the different bias schedules. We also show a very strong, constant bias, which is the closest that we can conceptually get to simply using only one camera path in both optimization steps. For bias scheduling to be useful the best performance ought to be somewhere between the extremes of no scheduling and strong, constant schedule.

Third, we use our complete methods on different camera path type subsets of our PFST-Handheld dataset. The camera path is the way the camera was used to follow the subject. Conceptually different kinds of camera motions in monocular video ego-motion estimation are associated with different degrees of conceptual difficulty.

Finally we compare our complete method with the closest method with respect to the task that we are aware of, GLAMR (Yuan et al., 2021). For a fair comparison we only evaluate flat motion, since GLAMR does not support obstacle motion.

5.6.1 Metrics

We evaluate our method on our PFST-Handheld dataset (see section 5.3) containing ground-truth camera and body motion for terrain and floor motion. We use the following metrics:

Subject's mean position error. This is a measure of global human motion estimation quality. We align the trajectories of estimated and ground truth root joints by a limited procrustes alignment. We only allow for rotation along the up/down-axis and a singular, all-dimensional scale factor. This accounts for the fact that the cVAE estimates the ground plane, hence rotational alignment around the ground plane axes would counteract this effort.

Camera path position error. This is a measure of camera motion estimation quality. Note that body and camera motion estimation quality are conceptually correlated due to reprojection constraints. Where possible, we prefer the subject's mean position error, which is more expressive due to less alignment. Vanilla RANSAC does not give a body motion or ground plane estimation. To compare cVAE to vanilla RANSAC we can only evaluate the camera path and a full 3D rotational alignment becomes necessary. The scale alignment is still a singular, all-dimensional alignment.

Field-of-view error. This is a measure of focal length estimation quality. The focal length can be expressed as a field-of-view in degree, from which we calculate the deviation. We report the error of the vertical field-of view, which is the larger one as the videos are recorded in portrait format.

Root relative mean per joint position error (MPJPE). This is a measure of the body pose estimation quality. The subject's poses are centered around their respective root joints, the absolute position is discarded. We scale fit the estimate to the ground truth with a singular, all-dimensional scale factor. We do not apply any further alignment. The MPJPE is the average metric distance between the respective joints in mm.

5.6.2 Training

We use the AMASS (Mahmood et al., 2019) dataset to train the cVAE following HuMoR (Rempe et al., 2021). For the cVAE optimization we use an AdamW (Loshchilov and Hutter, 2017) optimizer with a learning rate of $1e^{-4}$. We set $s_{\frac{R}{T}} = 2$. For all methods that use bias scheduling, we use a bias scheduling of $b_i = i \cdot 10$ if no other schedule is explicitly mentioned.

5.6.3 System Ablation

For our system ablation study (see table 5.1) we also implement a motion cVAE for human motion that is fixed to the ground plane, the *cVAE (fixed ground plane)*, using the egocentrization from chapter 3. This motion cVAE effectively is a streamlined

Table 5.1: System ablation study. The values represent the subject's mean position error in mm for all scene, scenes with obstacles and flat scenes.

Method	all	obstacle	flat
cVAE (fixed ground plane)	661	1023	119
cVAE	346	462	270
cVAE + Step Plane Estimation	313	440	122
cVAE + matching+RANSAC	65	70	58
cVAE + Step Plane Estimation + matching+RANSAC	**54**	**66**	**37**

HuMoR (Rempe et al., 2021) but with a moving camera and without stereo or depth information, which are simply added to the optimization loss in the original HuMoR. The *cVAE* builds upon this but includes our formulation for the egocentrization that allows height traversal. The *cVAE + Step Plane Estimation* and *cVAE + matching+RANSAC* self-explanatorily add the respective part of our method to the cVAE with height traversal. Using or not using step plane estimation is a matter of adding or leaving out the L_S-loss to the cVAE optimization loss. The *cVAE + matching+RANSAC* method is actually the first and only non-complete method to use bias scheduling as both camera path optimization methods, cVAE-based and matching+RANSAC, are used. Finally, *cVAE + Step Plane Estimation + matching+RANSAC* is our complete method.

The results can be seen in table 5.1. First, it can be observed that on a flat plane the cVAE that allows height traversal without any additional step plane estimation (line 2 in table 5.1) the performance is significantly worse than the cVAE that does not allow height traversal (line 1). Adding the step plane estimation improves the performance back to height-restricted levels. Expectedly, the cVAE with fixed ground plane has difficulties with a subject traversing terrain or obstacles. The height-relaxed cVAE performs better, but the step plane estimation only marginally increases the performance here. The largest performance increase can be seen by bundling in the matching+RANSAC procedure.

5.6.4 Iterative Refinement and Bias Scheduling

Combining the cVAE optimization and the matching+RANSAC procedure for camera path estimation is only useful if it results in a better camera path estimation than both methods can achieve separately. Here we show that this can be the case and how our solution of combining both methods has to be setup for a better combined performance.

Table 5.2: Bias scheduling study. The values represent the procrustes aligned camera path position error in mm (number of iterations to converge in parenthesis).

Bias Schedule	cVAE+SPE		matching+RANSAC	
$b_i = 0$	250	(1)	632	(1)
$b_i = 1000$	283	(2)	283	(1)
$b_i = i \cdot 10$	40	(37)	40	(36)
$b_i = i \cdot 20$	40	(20)	40	(19)
$b_i = i \cdot 50$	41	(9)	41	(8)
$b_i = i \cdot 100$	110	(5)	110	(4)
$b_i = i \cdot 200$	188	(3)	188	(2)

Note that due to the nature of the cVAE optimization and the matching+RANSAC procedure, it is not possible to simply trivially combine them. RANSAC is not an iterative optimization process, it is based on random sampling. Each random sample set yields a full algorithmic camera path result that is independent of another random sample set's calculated camera path. In our iterative refinement process we iteratively and alternatingly optimize the cVAE's camera path and estimate a matching+RANSAC camera path. We solve the conceptual disjointedness in both steps by biasing each method's calculation towards the last camera path calculated with the respective other method. We call the progressing of the strengths of the biases over the iteration steps *bias schedules*.

We show the results of different bias scheduling schemes in table 5.2. Without the biases ($b_i = 0$), there is no reason for the methods to converge. We can treat them as to competing methods for camera path estimation here. With high, constant biases, we can see convergence to a sub-optimal result that does not improve upon the best non-converging result from the *cVAE+SPE*. All other slower schedules do actually pareto dominate the non-converging methods. The linear schedules see an increasing but saturating performance the slower the biases grow. Since specifically the cVAE optimization takes some significant computational resources and time, it makes sense to try finding the bias schedule that results in the pareto optimal outcome as quickly as possible. Here this is $b_i = i \cdot 20$.

Choosing a proper bias scheduling scheme to combine the cVAE camera path optimization and the matching+RANSAC estimation improves upon all of: using a simple,

Table 5.3: Analysis of the camera path type's influence on the result.

Camera Path Type	fov error (°)	subject mean position error (mm)
lateral (90° and 45°)	6.1	44
radial	0.9	31
follow fast	3.1	82
follow slow fade	1.5	65

strong constant bias; only using cVAE camera path optimization and only using matching+RANSAC estimation.

5.6.5 Camera Path Type

We group the dataset by camera path type (see Figure 5.2 and Figure 5.3). Note that e.g. for the *radial* type path, there are videos of two subjects, with each a flat and an obstacle motion, each taken with four different focal lengths.

The more the subject moves along the depth axis relative to the camera, the more accurate the focal length estimation. The *radial* camera motion (0.9°error) has the most depth-axis camera-relative subject motion because the subject is first walking towards the camera and then away from it. In contrast, both *lateral* (6.1° error) and *follow fast* (3.1° error) aim to keep the depth distance between camera and subject constant by following the subject as constantly as possible. This makes it harder for the cVAE to infer useful depth information from the subject's motion. It's even easier to keep the depth distance constant for *lateral* camera following than with the *follow fast* camera motion from behind the subject, which might explain why the focal length estimate is worse. Interestingly, the results for the subject's motion are reversed: while there is not much depth information to estimate the camera's field of view in *lateral* (44mm error) camera motion, it is also not necessary to estimate depth for a good subject's motion estimate.

5.6.6 State of the Art Comparison

We are not aware of a method that supports non-flat terrain, moving camera and focal length estimation. GLAMR (Yuan et al., 2021) does so with the exception of only being able to handle flat motion. So we compare against it using only our *flat* sequences. We beat it both in root relative MPJPE and subject mean position error. Note

Table 5.4: Comparison to state of the art. Only flat motions are evaluated, since no method known to us supports both moving camera and human terrain traversal.

Method	root relative MPJPE	subject mean position error (mm)
GLAMR (Yuan et al., 2021)	45	62
Ours	**35**	**37**

that GLAMR supports multiple subjects within one video and severe occlusion. Both are not supported by our method or featured in our dataset. It is possible that a multi-person GLAMR would outperform our method applied to multiple subjects in the same video independently.

5.7 Conclusion

Our novel method is able to infer camera intrinsics, camera motion and human motion for a subject filmed with a moving, handheld camera walking on arbitrary terrain. We do not learn or assume features from our test dataset that we recorded for the purpose of evaluating our method. Therefore we expect a good generalization within the domain of tasks that our model is able to handle.

We assessed the usefulness of our components in a system ablation. We show that both step plane estimation helps stabilizing the motion model when the subject is walking on flat terrain. With *SPE* we beat a state of the art competitor that only supports flat motion evaluated on only flat motion from out dataset.

The forced convergence of the motion model and the simple visual odometry yields better camera ego-motion results than both methods give independently. Note that the motion model alone can already give a camera ego-motion estimate via reprojection constraints with respect to the human subject.

Our analysis of subject-relative camera motion paths shows that the ability to infer the camera's intrinsics from human motion depends on the relative motion between camera and subject with respect to the camera's depth axis. Note that most general purpose visual odometry methods are not able to infer the focal length but instead need the camera's calibration as an input. We effectively use the human as a calibration subject.

We deliberately chose a simple visual odometry method to fuse into our system. Big, complex state of the art visual odometry or SLAM systems are out of scope for our

work. One reason is that complex, established methods do not allow us to work with a missing or bad camera intrinsics initialization without deconstructing, re-thinking and re-implementing the system. Though, conceptually this should be possible. Given our results we hypothesize that fusing in the cVAE motion can even improve state of the art visual odometry; but this is to be tested in future work.

Chapter 6

Conclusion

In chapter 3 we built an adversarial regression model to estimate global human motion without paired video-motion ground truth labels. In chapter 4 we then trained a regression model for global human motion estimation together with camera extrinsics and intrinsics estimation. We trained this regression model by creating a synthetic training dataset and exploited DensePose as an intermediate representation for better domain gap bridging between real and synthetic data. In chapter 5 we trained a human motion model and invented a method to combine it with classic camera pose estimation methods. With this we were able to estimate global motion even if recorded with a handheld camera and when walking over obstacles.

We showed that for the settings of a known and static camera (chapter 3), unknown and static camera (chapter 4) and unknown, moving camera (chapter 5) directly estimating global motion can improve the local, root relative pose estimation performance and the global motion estimation performance (hypothesis **H1**). We also show that if the camera calibration is unknown, we can jointly estimate global motion, camera intrinsics and extrinsics to enable global motion estimation, which effectively uses the human subject as a calibration reference (hypothesis **H2**). We show that this is possible when the camera is static and the subject is moving on a flat plane (chapter 4) as well as when the camera is moving and the subject is traversing terrain with obstacles (chapter 5).

6.1 Supervision from Human Motion Datasets

All the approaches have the property in common that they do not use any of the training labels of the benchmark datasets. The supervision for the human motion always comes

from general motion sequence datasets that are not paired or linked to video sequences. The adversarial regression model in chapter 3 is the only method that does use the training videos without labels, the other two methods do not use neither training videos nor their labels. This means that the adversarial regression model is a little bit less flexible when it comes to generalizing across different camera setups. New training videos with calibrated cameras would need to be created in order to achieve a performance similar to the reported one on a setup with a new camera setting. But note that this excludes the necessity for motion capture and is still much cheaper than creating new videos with ground truth labels This would in comparison be something expected to be necessary for the bulk of methods supervisedly trained on training videos with their paired human motion labels.

The creation and rendering of the synthetic dataset in chapter 4 was by far the most expensive one time computation within this PhD project. From start to full training it took at least two months of computation on several machines. This makes creating synthetic datasets an approach that resembles brute force. Further, the performance gains from improving the amount of data never actually saturated in the project, so we simply had to stop the computations because of time constraints.

6.2 Regression vs. Optimization

In chapter 3 and chapter 4 we trained regression models while chapter 5 is based on a test-time optimization approach. Regression Models tend to be harder to control: Trade-offs, e.g. the trade-off between projection fit and motion quality, can only be made in training. Note that the adversarial regression model and the cVAE motion model's test-time optimization are conceptually two very similar ideas: they both learn distributions of human motion, they both use the egocentrization trick and they both use a differentiable projection formulation to calculate a reprojection loss or energy. But the adversarial regression model is much harder to train and to control. The output 3D human motion is actually not guaranteed to be close to the 2D motion projection and we also do not get an estimate of the general motion quality. In the test-time optimization approach we simply can define the trade-off between both in the weighting of the respective sub energies. Another problem is that the adversarial regression model needs both lots of training motion and training videos. As described, an advantage over other approaches is that training videos and motion can be unpaired or even completely disjoint. But a huge set of different videos with preferably lots of variation in

subjects, tasks, etc is needed; and the larger and more varied the video dataset, the better the adversarial regression model can be. For the test-time optimization approach we do not have the necessity to look at multiple videos at once, hence we do not have the dependency on a large and varied video dataset. The only pre-test training that we need is the human motion cVAE trained on human motion sequences. When it comes to videos, the test-time optimization approach only ever looks at the one it is interested in; and that is in test-time. This makes it much easier to train and control; and is also the reason why we see better results with it. Because of the conceptual similarity of the approaches, it would actually be possible to adapt the adversarial regression model to the same setting, i.e. a free roaming unknown camera and human motion over obstacles, using similar ideas. But then we would need to create videos that also vary in camera paths and obstacle courses which is an exponential increase of the data necessary, and not feasible even without the need for motion capture ground truth.

6.3 Future Work

For future work, there are still many problems unsolved towards the ability to process and obtain human motion estimates from arbitrary videos in the wild. Dynamically changing focal length, also known as zoom, could be handled by our cVAE by relaxing the focal length parameter to allow dynamic changes. The focal length parameter could be a time series that is directly optimized.

We also completely disregarded multi-person videos. We can actually expect that multiple people in one video do offer a more robust camera path estimation when following the cVAE test-time optimization approach. The reason is that the different subject's residual projected motions that can not be explained by their own global motions and have to be attributed to the camera motion; all those have to agree for multiple subjects. Consequentially they also positively influence each other's predicted global motion. For this to work there would have to be some tracking applied beforehand so that we can follow the subjects through the video.

One could also work on switching out the simple feature matching based camera pose estimation in the cVAE based optimization loop in chapter 5 and replace it with a full SLAM system, like ORB-SLAM (Mur-Artal and Tardós, 2015), instead. This would aim in the direction of semantic SLAM where semantic knowledge about the world is used for stabilizing a SLAM system.

Another work could be to re-identify subjects after jump cuts, specifically when the perspective is changed. This could be done with an appearance based re-identification approach. One could also hallucinate in-between motion when the person is out of frame for a significant amount of time. One could use context like speech or the style of the motion before and after to come up with generated proposals for the subject's in-between motion.

Bibliography

Agarwal, A. and Triggs, B. (2006). Recovering 3d human pose from monocular images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

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

Arjovsky, M., Chintala, S., and Bottou, L. (2017). Wasserstein generative adversarial networks. In Precup, D. and Teh, Y. W., editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 214–223. PMLR.

Arnab, A., Doersch, C., and Zisserman, A. (2019). Exploiting temporal context for 3d human pose estimation in the wild. *CVPR*.

Arora, S., Ge, R., Neyshabur, B., and Zhang, Y. (2018). Stronger generalization bounds for deep nets via a compression approach. In *ICML*.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg.

Bradski, G. (2000). The OpenCV Library. *Dr. Dobb's Journal of Software Tools*.

Cao, Z., Hidalgo, G., Simon, T., Wei, S.-E., and Sheikh, Y. (2018). Openpose: realtime multi-person 2d pose estimation using part affinity fields. *arXiv preprint arXiv:1812.08008*.

Cao, Z., Simon, T., Wei, S.-E., and Sheikh, Y. (2017). Realtime multi-person 2d pose estimation using part affinity fields. In *CVPR*.

Chen, C.-H. and Ramanan, D. (2017). 3d human pose estimation = 2d pose estimation + matching. In *CVPR*.

Chen, C.-H., Tyagi, A., Agrawal, A., Drover, D., MV, R., Stojanov, S., and Rehg, J. M. (2019). Unsupervised 3d pose estimation with geometric self-supervision. In *CVPR*.

Chojnacki, W., Brooks, M. J., van den Hengel, A., and Gawley, D. (2003). Revisiting hartley's normalized eight-point algorithm. *IEEE Trans. Pattern Anal. Mach. Intell.*, 25:1172–1177.

Dabral, R., Mundhada, A., Kusupati, U., Afaque, S., Sharma, A., and Jain, A. (2018). Learning 3d human pose from structure and motion. In *ECCV*.

Dai, J. S. (2015). Euler–rodrigues formula variations, quaternion conjugation and intrinsic connections. *Mechanism and Machine Theory*, 92:144–152.

Felzenszwalb, P. F., Girshick, R. B., McAllester, D., and Ramanan, D. (2010). Object detection with discriminatively trained part-based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Fengxiang He, D. T. (2020). Recent advances in deep learning theory. *CoRR*, abs/2012.10931.

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

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press. http://www.deeplearningbook.org.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial networks. *Advances in Neural Information Processing Systems*, 3.

Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., and Courville, A. C. (2017). Improved training of wasserstein gans. In *Neurips*.

Habekost, J., Pang, K., Shiratori, T., and Komura, T. (2022). From synthetic to one-shot regression of camera-agnostic human performances. In El Yacoubi, M., Granger, E., Yuen, P. C., Pal, U., and Vincent, N., editors, *Pattern Recognition and Artificial Intelligence*, pages 514–525, Cham. Springer International Publishing.

Habekost, J., Shiratori, T., Ye, Y., and Komura, T. (2020). Learning 3d global human motion estimation from unpaired, disjoint datasets. In *BMVC*.

Hartley, R. and Zisserman, A. (2004). *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2 edition.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9:1735–80.

Holden, D., Saito, J., and Komura, T. (2016). A deep learning framework for character motion synthesis and editing. *ACM Transactions on Graphics (TOG)*.

Hossain, M. R. I. and Little, J. J. (2018). Exploiting temporal information for 3d human pose estimation. In *ECCV*.

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

Jain, A., Tompson, J., Andriluka, M., Taylor, G. W., and Bregler, C. (2014). Learning human pose estimation features with convolutional networks. In *ICLR*.

Ji, S., Shan, J., Shao, X., Yuan, X., Yang, P., Wenbin, W., Tang, H., and Shibasaki, R. (2015). Particle filtering methods for georeferencing panoramic image sequence in complex urban scenes. *ISPRS Journal of Photogrammetry and Remote Sensing*, 105.

Kanazawa, A., Black, M. J., Jacobs, D. W., and Malik, J. (2018). End-to-end recovery of human shape and pose. In *CVPR*.

Kanazawa, A., Zhang, J. Y., Felsen, P., and Malik, J. (2019). Learning 3d human dynamics from video. In *CVPR*.

Kavan, L., Collins, S., Žára, J., and O'Sullivan, C. (2007). Skinning with dual quaternions. In *Proceedings of the 2007 Symposium on Interactive 3D Graphics and Games*, I3D '07, page 39–46, New York, NY, USA. Association for Computing Machinery.

Kingma, D. P. and Welling, M. (2014). Auto-encoding variational bayes. In Bengio, Y. and LeCun, Y., editors, *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*.

Kingma, D. P. and Welling, M. (2019). An introduction to variational autoencoders. *Foundations and Trends® in Machine Learning*, 12(4):307–392.

Kocabas, M., Athanasiou, N., and Black, M. J. (2019). Vibe: Video inference for human body pose and shape estimation. *arXiv:1912.05656*.

Kocabas, M., Huang, C.-H. P., Tesch, J., Müller, L., Hilliges, O., and Black, M. J. (2021). SPEC: Seeing people in the wild with an estimated camera. In *ICCV*.

Lea, C., Flynn, M. D., Vidal, R., Reiter, A., and Hager, G. D. (2017). Temporal convolutional networks for action segmentation and detection. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1003–1012, Los Alamitos, CA, USA. IEEE Computer Society.

Lecun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. (2014). Microsoft coco: Common objects in context. In Fleet, D., Pajdla, T., Schiele, B., and Tuytelaars, T., editors, *Computer Vision – ECCV 2014*, pages 740–755, Cham. Springer International Publishing.

Loper, M., Mahmood, N., and Black, M. J. (2014). Mosh: Motion and shape capture from sparse markers. *ACM Transactions on Graphics (TOG)*.

Loper, M., Mahmood, N., Romero, J., Pons-Moll, G., and Black, M. J. (2015). SMPL: A skinned multi-person linear model. *SIGGRAPH Asia*.

Loshchilov, I. and Hutter, F. (2017). Fixing weight decay regularization in adam. *CoRR*, abs/1711.05101.

Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2):91–110.

Mahmood, N., Ghorbani, N., Troje, N. F., Pons-Moll, G., and Black, M. J. (2019). Amass: Archive of motion capture as surface shapes. In *ICCV*.

Martinez, J., Hossain, R., Romero, J., and Little, J. J. (2017). A simple yet effective baseline for 3d human pose estimation. In *ICCV*.

Mehta, D., Rhodin, H., Casas, D., Fua, P., Sotnychenko, O., Xu, W., and Theobalt, C. (2017a). Monocular 3d human pose estimation in the wild using improved cnn supervision. In *3DV*.

Mehta, D., Sridhar, S., Sotnychenko, O., Rhodin, H., Shafiei, M., Seidel, H.-P., Xu, W., Casas, D., and Theobalt, C. (2017b). Vnect: Real-time 3d human pose estimation with a single rgb camera. *ACM Transactions on Graphics (TOG)*.

Mirza, M. and Osindero, S. (2014). Conditional generative adversarial nets. *CoRR*, abs/1411.1784.

Mur-Artal, Raúl, M. J. M. M. and Tardós, J. D. (2015). ORB-SLAM: a versatile and accurate monocular SLAM system. *IEEE Transactions on Robotics*, 31(5):1147–1163.

Mur-Artal, R. and Tardós, J. D. (2017). Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE Transactions on Robotics*.

Nair, V. and Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In *ICML*.

Newell, A., Yang, K., and Deng, J. (2016). Stacked hourglass networks for human pose estimation. In *ECCV*.

Pandey, A. and Wang, D. (2019). Tcnn: Temporal convolutional neural network for real-time speech enhancement in the time domain. In *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6875–6879.

Pavlakos, G., Zhu, L., Zhou, X., and Daniilidis, K. (2018). Learning to estimate 3D human pose and shape from a single color image. In *CVPR*.

Pavllo, D., Feichtenhofer, C., Grangier, D., and Auli, M. (2018). 3d human pose estimation in video with temporal convolutions and semi-supervised training. *arXiv*, abs/1811.11742.

Peng, X. B., Kanazawa, A., Malik, J., Abbeel, P., and Levine, S. (2018). Sfv: Reinforcement learning of physical skills from videos. *ACM Transactions on Graphics (TOG)*.

Qi, C. R., Su, H., Mo, K., and Guibas, L. J. (2016). Pointnet: Deep learning on point sets for 3d classification and segmentation. *arXiv preprint arXiv:1612.00593*.

Rempe, D., Birdal, T., Hertzmann, A., Yang, J., Sridhar, S., and Guibas, L. J. (2021). Humor: 3d human motion model for robust pose estimation. In *ICCV*.

Rhodin, H., Sporri, J., Katircioglu, I., Constantin, V., Meyer, F., Erich Muller, M. S., and Fua, P. (2020). Learning monocular 3d human pose estimation from multi-view images. In *CVPR*.

Riza Alp Guler, Natalia Neverova, I. K. (2018). Densepose: Dense human pose estimation in the wild. In *CVPR*.

Ronneberger, O., Fischer, P., and Brox, T. (2015a). U-net: Convolutional networks for biomedical image segmentation. In Navab, N., Hornegger, J., Wells, W. M., and Frangi, A. F., editors, *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, pages 234–241, Cham. Springer International Publishing.

Ronneberger, O., Fischer, P., and Brox, T. (2015b). U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, 323:533–536.

Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117.

Shimada, S., Golyanik, V., Xu, W., and Theobalt, C. (2020). Physcap: Physically plausible monocular 3d motion capture in real time. *ACM Transactions on Graphics (TOG)*.

Taubin, G. (2011). 3d rotations. *IEEE Computer Graphics and Applications*, 31(6):84–89.

Tekin, B., Rozantsev, A., Lepetit, V., and Fua, P. (2015). Direct Prediction of 3D Body Poses from Motion Compensated Sequences. *arXiv e-prints*, page arXiv:1511.06692.

Toshev, A. and Szegedy, C. (2014). Deeppose: Human pose estimation via deep neural networks. In *CVPR*.

van den Oord, A., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A. W., and Kavukcuoglu, K. (2016). Wavenet: A generative model for raw audio. *CoRR*, abs/1609.03499.

Varol, G., Romero, J., Martin, X., Mahmood, N., Black, M. J., Laptev, I., and Schmid, C. (2017). Learning from synthetic humans. In *CVPR*.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. (2017). Attention is all you need. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Xie, K., Wang, T., Iqbal, U., Guo, Y., Fidler, S., and Shkurti, F. (2021). Physics-based human motion estimation and synthesis from videos. In *ICCV*.

Xu, W., Chatterjee, A., Zollhöfer, M., Rhodin, H., Mehta, D., Seidel, H.-P., and Theobalt, C. (2018). Monoperfcap: Human performance capture from monocular video. *ACM Transactions on Graphics (TOG)*.

Yuan, Y., Iqbal, U., Molchanov, P., Kitani, K., and Kautz, J. (2021). Glamr: Global occlusion-aware human mesh recovery with dynamic cameras. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11028–11039.

Zhou, X., Huang, Q., Sun, X., Xue, X., and Wei, Y. (2017). Towards 3d human pose estimation in the wild: A weakly-supervised approach. In *ICCV*.

Zhu, R., Yang, X., Hold-Geoffroy, Y., Perazzi, F., Eisenmann, J., Sunkavalli, K., and Chandraker, M. (2020). Single view metrology in the wild. In Vedaldi, A., Bischof, H., Brox, T., and Frahm, J.-M., editors, *ECCV*.