Motion Priors for Pose Estimation and Animation Workflows

Luke Smith

Master Thesis
April 2022

Prof. Dr. Robert W. Sumner

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

cgl
computer graphics laboratory

Abstract

TODO: Abstract

Zusammenfassung

TODO: translate to German

Contents

List of Figures

List of Tables

Introduction

Introduction

- Existing pipeline description - 2d keypoints - 3d keypoints - Optimisation of cameras, ground plane, etc. - Existing pipeline problems - Robustness - Investigation: Motion Priors - Use for plausible motion

Related Work

To begin with, a review of Pose Estimation techniques is presented, as this forms an important part of the motion learning pipeline, which often operates on 2D/3D pose estimations. Next the motion prior literature is explored and related works for learning human motion are presented.

2.1. Pose Estimation

The existing pipeline for 2D pose estimation is based on Open Pose [CHS+19]. OpenPose TODO

2.2. Motion Priors

Classical methods are presented in [SMK22], the involve matching/interpolating from existing databases.

The authors of HuMoR [RBH+21] presented a novel approach for learning and using a plausible motion prior. They train a conditional VAE that learns a distribution over latent transitions, in a canonical reference frame, between *states* that consist of a root translation, 3D joint positions, joint angles, and the respective velocities. They most notably use this model as a prior in a 'test time optimisation', which generates plausible sequence motions optimising for an initial state and a sequence of transitions starting from frame by frame estimates (2D/3D joints or points clouds). This optimisation includes, alongside others, a motion prior term based upon the conditional distribution $p(z_t|x_{t-1})$ that encourages plausible motion for the learned sequence. Note that the CVAE decoder also predicts ground plan contact alongside change in state, which are used as regularisers during their main use case 'test time optimisation'. The test time opti-

misation can operate on many modalities, 2D/3D joints, point clouds, etc., as the optimisation contains a Data Term ϵ_{data} that can be tailored to the modality as the HuMoR state is information rich, containing 3D joints (hence can fit to 2D joints through projection or directly to 3D) and can parametrise the SMPL model (hence the SMPL mesh can be correlated to point clouds). The initialisation for the test time optimisation is based upon VPoser TODO: Complete.

HuMoR discussions:

- Assumptions:
  - The method necessitates knowledge of the ground plane, which is presently needed (empirical observation) for convergence during training (as the dataset is of motions with a flat ground), and thus also at test time even though it is not conceptually necessary
  - Assumes static camera
- Limitations:
  - Single person formulation

The authors of HuMoR [RBH+21] were inspired by the Motion VAE [LZCvdP21] paper. This paper uses an Conditional VAE (with assumed standard normal prior conditioning (vs. NN in HuMoR)) that directly outputs the next state (rather than the change in state in HuMoR). The model is used Autoregressively to predict motion (rather than the main presented use of HuMoR which is to fit motion to a sequence of existing 2D/3D joint predictions, though HuMoR can equally well be used autoregressively), and is trained with the typical ELBO in a supervised manner.
Some notes to self about MotionVAE

- RL algo trained to walk the latent space
- Some notes about things they mention in the related work section:
  - They cite [Wang et al. 2019] who train a stochastic generative model with output *processed by a refiner network to remove foot skating and add robustness*.
- Main differences to HuMoR
  - c.f discussion section in HuMoR
  - Conditional prior
  - Predict change in motion
  - Predict ground contacts
  - Much additional regularisation in training
  - Difference state representation (root projected to ground)
  - Use of SMPL by HuMoR
  - Difference in network architectures
    - HuMoR just uses MLPs and MotionVAE decoder is a 'MANN-style mixture-

of-expert neural network' (6 networks, gating network weighting their outputs)

* RELU in HuMoR, ELU in MotionVAE
* MotionVAE decoder has latent variable input at each layer (not sure about HuMoR)

–

The learned-inbetweenings [TWH+22] paper is very similar to MotionVAE. They present a similar architecture except that they predict transitions as in HuMoR [RBH+21], and that they seem to only predict the lower body joint velocities and rotations. The decoder architecture is similar with a gating network and multiple expert networks. They also train a sampler to sample from the latent space, similar to MotionVAE which trains a RL model. Some notes to self about Learned-Inbetweenings

- Contains a nice probabilistic formulation of the motion generation problem

There seem to be quite a number of works that present the CVAE architecture as a base with varying state representations, conditioning variables and loss terms, [RBH+21, TWH+22, LZCvdP21, MWFD21]. [RBH+21] has a learned prior, , [TWH+22] has a normal prior and predicts

DeepPhase [SMK22] proposes a convolutional autoencoder that operates on fixed length TODO: I beleive fixed length sequences of 3D joint velocities, learning a latent space that it encourages to represent sinusoidal functions (through phase/frequency/amplitude/offsets) that represent periodic features of motion. The auto-encoder maps to and from sequences of 3D joint velocities, and the latent variables represent a sequence of phase/etc. values over the whole motion, hence the change in parameters can represent a shift between different periodic motions and thus can describe non-periodic motions.

MEVA [LGK20] postulates that learning a single motion model results in smooth motion, as on average human motion is smooth (i.e we are not shaking while walking (their words)), hence they propose a two stage pipeline, in which the results of VAE that esimates coarse motion is passed into a human shape regressor that refines the poses, the inputs are temporally correlated features hence temporal consistency is maintained. The paper also presents some motion specific data augmentation techniques, speed variation through sampling, mirroring, and root rotations. Some notes to self about Learned-Inbetweenings

- TODO: very useful
- Nice related work section
- Nice dataset section

Another approach is presented in VIBE

2.3. Overview of Approaches

2.3.1. Motion Priors

We are most interested in models that learn plausible, task independent, human motion. These are refered to by [LZCvdP21] as *Motion-then-control* models. We limit our scope to parametric models.

- **Motion Priors**
- MotionVAE [RBH+21]
  - Standard normal CVAE
  - Outputs next pose
  - Decoder is mixture of networks
  - Trained with rollout and scheduled sampling
  - State positions referenced to root projection onto ground
  - Nice investigation into using RL in the latent space for character control
  - NOTE: Latent dimension size: 32 (typical physics based humanoid degrees of freedom).
  - The state having velocities and the decoder predicting change in pose seems to implicitly model the time aspect of the motion, rather than explicitly modelling it like in [MWFD21].
- HuMoR [RBH+21]
  - Parametrised conditional prior CVAE
  - Outputs change in state and person ground contacts
  - SMPL regularisers (a subset of their state parametrises the SMPL model)
  - Motion learned in a canonical reference frame (TODO: not sure about MotionVAE)
  - Trained without rollout (I beleive?)
  - State positions referenced as in SMPL model (to $(0, 0)$?)
  - Ground plane initialized with RCNN
  - Very nice feature of having velocities in the state and of predicting change in motion, this implicitly captures the direction of motion in time as well as in space
- Learned-inbetweenings paper [TWH+22]
  - Bascially MotionVAE but outputs change in state like HuMoR
- Structed latent space for 4D motion [MWFD21]
  - VAE operating on a fixed number of frames but with 'varying duration' by including

a timestamp per frame (hence with wider spacing in the timestamp the movement is over a longer duration)

  - Conditions decoder with SMPL shape
  - It encorporates the timestamp to distinguish the direction of motion (i.e to avoid having you sample backwards in time when you walk the latent space. It's a direct next pose prediction and so would cluster close poses in the latent space regardless of time (I beleive))
  - They perform a comparison to other reconstruction methods to directly evaluate the VAE, not sure other papers did that much
  - Didn't find the paper so interesting
- DeepPhase [SMK22]
  - Autoencoder operating on fixed length sequences of 3D joint velocities
  - Latent space enforced to match sinusoidal functions that represent periodic motion
  - Periodic functions can change over the length of the sequence thus shift between periodic motions and represent non-periodic motions
- MEVA [LGK20]
  - Separates pipeline into learning coarse motion VAE and refining this prediction (they postulate the VAE can only learn smooth motion, then use a SMPL regressor to refine the predictions from temporally correlated features)
  - VAE operates on features extracted with temporal convolutions directly from the image rather than on SMPL/joint position/velocity based state
  - Presents nice augmentation techniques for AMASS, speed variation, mirroring and root rotations
  - Nice related work and dataset sections
- TransformerVAEPrior [CSY+22]
  - Learns a latent space using AMASS, then has another network (transformer based) to project to this latent space directly from video, decodes straight to a motion sequence
  - Operates on a **sequence** level, rather than just between 2 poses like HuMoR, latent code represents an entire sequence so the sequence can be decoded in one step => faster
  - Also uses AMASS
  - Can also be used as a motion rectifier like Humor, they say it's faster as it's a direct prediction of sequence
  - TODO: not sure about the length of the input sequence
- [HSKJ15]

  - Old paper but relevant ideas
  - Simple autoencoder (not variational), CNN based
  - Operates on sequence level, like TransformerVAEPrior
  - Denoising Autoencoding training (corrupting inputs with noise and recovering uncorrupted version), might be helpful for us as we want to de-corrupt data essentially
  - ConvAutoEnv2016 [HSK16] improves the architecture a little I beleive?
- Hierarchical Motion Model VAE [LVC+21]
  - Skeletal aware convolutions/pooling/unpooling
  - Fixed length motion sequence, but describes a sliding window method for using their latent space to refine longer sequences
    * Project onto latent space and decode, take center frame, move window one step forward, repeat
    * Not sure what they do at the beginning and the end of the sequence
    *
  - 2 latent vectors?? local and global??
- **Diffusion**
- Avatars grow legs (sparse to complete diffusion) [DKP+23]
  - denoising sequence of SMPL params conditioned by sparse tracking inputs
    * Sparse input per frame is concatenated onto the corresponding denoised frame
  - input: sparse tracking inputs (orientation/translation of headset and two hand controllers)
  - output: SMPL root orientation/translation, pose parameters
  - Fixed input/output sequence length
  - Architecture
    * MLP
- PhysDiff [YSI+22]
  - denoising sequence poses conditioned on promt, e.g 'walking round a bend'
  - Goal: create sequence of motion from promt
  - Physics-based motion projection step in the diffusion
    * Achieved using a motion imitation policy to control a character in a physics simulator
  - Architecture
    * TODO

- EDGE [TCL22]
  - Denoising sequence poses conditioned on music features
  - Describes inpainting which could be useful to us
  - Architecture
    - Transformer based
- **Motion aware but not focused on prior**
- VIBE
  - Operates directly on video
  - CNN features processed by GRUs (gated reccurence unit) to temporally correlate the features
  - Features fed into a NN regressor to estimate SMPL params as in [KBJM17]
  - Discriminator network jointly trained to introduce an extra loss to encourage plausible motion
- 3DDynamicsFromVideo [KZFM18]
  - Operates directly on video sequences
  - PRedicts pose at t-1,t,t+1 from all image features during training and also jointly trains a NN that predicts t-1,t,t+1 from just image at t
- Some more historic RGB methods presented in HuMoR [RBH+21] if needed
- Optimisation methods that refine predictions presented in HuMoR [RBH+21], e.g with smoothness priors or scene contact info
- **Pose estimation**
- HULC
  - Uses a scene point cloud to help generate dense contact estimation labels that are used to guide pose manifold sampling
- VPoser
  - Not actually a motion prior, it's a pose prior
  - Is used in Humor to help initialise the sequence of states
  - VPoser is used by optimising pose directly in the latent space, the latent space is trained to be a normal distribution hence if you penalise the norm of the latent vector you are encouraging it to be close to what you've learned to be viable human poses (i.e close to the normal dist)
- **TODO: Things I haven't looked into so deeply**
- RNNS
  - RNNs seem to be commonly used for generating future motion condition on control

variable

  - Mixture-density network RNNs (MDN-RNNS)
    * Referenced in [LZCvdP21]
    * Output a distribution as a gaussian mixture model
  - SpatioTemporalRNN
    * (https://arxiv.org/pdf/1908.07214.pdf) cited by learned-inbetweening paper [TWH+22]
    * Learns a manifold through encoder/decoder
    * Separates 'spatial' and 'temporal' in encoder and decoder??
    * Predicts in batches with RNN, they claim this forces the model to capture mid and long term connections (I beleive the explicit velocity modelling in HuMoR should do the same thing)
  - Deep latent variable model
    * https://dl.acm.org/doi/pdf/10.1111/cgf.14116

- Other VAE motion prediction
  - Unified3DHumanMotionSynthesis
  - ActionConditionedTransformer
    * Throw a transformer at the problem
    * Direct pose sequence prediction
    * Condition on action
- Time-convolutional autoencoders
  - Referenced in [LZCvdP21]
  - Learns a latent motion manifold
- Humor claims normalising flows and neural ODEs show potential but then only links to papers explaining these concepts and not actually using them for this purpose so not sure
  - (Normalising flow: map to a simple distribution with an invertible function => tractable marginal likelihood (unlike with VAEs where we have to deal with an ELBO), but I'm not sure we care about the marginal likelihood in this case)
  - [HAB19] MoGlow referenced, it predicts next pose directly by sampling from a simple fixed distribution and then transforming to a pose via a normalising flow conditioned on conditioning variables and pose histories that are updated and introduced through a hidden lstm in the flow.

2.3.2. Pose Completion

The other possibile research direction related more to pose completion (or motion completion (though I assume motion completion would use similar methods to what was mentioned before)) in an animation setting.

- Protores [OBH+21]
  - Learned inverse kinematics solution
  - Variable number of effector inputs processed and then mixed with a 'Proto' layer in the encoder
  - Decoder takes the pose embedding and decodes the full pose (contains several blocks to separate the semantically different parts of the decoding process)

Main

TODO: Intro to central work section

A number of directions were explored during this thesis. To begin with, in Section 3.1 and 3.2, an attempt was made at adapting a state of the art method, [RBH+21]. Next, a new approach to the problem alongside some novel architectures were explored to try and mitigate the issues encountered in Section 3.2.

3.1. HuMoR Investigation

TODO: Describe the humor method better, update the hand drawn diagrams with draw.io, read through and check everything is correctly introduced and I'm not assuming any pre knowledge of Humor

The authors of [RBH+21] propose a compelling approach to the problem of learning a human motion model (namely the HuMoR model), alongside a well described method for employing this model in the task of improving a sequence of independently estimated 2d poses obtained from video data (this method is referred to as 'TestOps'). Considering that our problem definition matched exaclty this TestOps, that our desire was to use a motion model with the intuition that it would improve motion prediction in certain important situations such during occlusion, and the promising results showed by the paper, it was deemed that HuMoR represented a sensible starting point for our research.

TODO: Describe the HuMoR method in more detail

3.1.1. Method

Our investigation began with a largely qualitative evaluation of the HuMoR model, which had two main aims. First was to stress test the system, to see where it failed, where it succeded, and if the mentioned benefits in the HuMoR paper [RBH+21] were as described. Second was to evaluate the model with the defects of the current Disney system in mind to see if it could complement it's functionality. The current systems most notable failure points arise in occluded situations?? TODO: Any other failure cases

To acheive this goal, a selection of videos were taken in the lab containing a variety of motion; fast, slow, and abnormal, and a including number of occluded scenes. The HuMoR system was ran on these videos and the results were investigated. The TestOps is performed in 3 stages, where only the 3rd makes use of the HuMoR motion model, we could therefore compare the stage 2 results to the stage 3 results to see where the model was providing an improvement over a more classical optimisation that includes smoothness and pose prior losses.

3.1.2. Advantages of HuMoR

We see a number of situations in which the model shows a clear improvement over the stage where the HuMoR model is not used.

In an occluded situation, as shown in Figure 3.1, where the 2d pose predictions don't have any information on the legs, the model manages to produce a realistic sitting motion.

(a) OpenPose **(b)** Stage 2 **(c)** Stage 3

Figure 3.1.: *HuMoR achieving occluded sitting*

We also note in many of the less difficult videos, that HuMoR produces clean motion and deals with many abnormal movements without obvious issues. It therefore doesn't seem to regress the easy situations, but can improve the more difficult situations, notably occlusions.

3.1.3. Drawbacks of HuMoR

We noted that while HuMoR manages to sit when occluded, it often fails to walk. This seems to be due to the fact that OpenPose often predicts both legs on the frames just before the occlusions where only one leg is actually un-occluded, as can be seen in Figure 3.2. This results in a sequence of poses where the frames before an occlusion indicate that the person is no longer walking, hence making it significantly more difficult for HuMoR to begin walking again during

the occlusion. This issue is thus probably largely due to HuMoRs' TestOps' dependence on OpenPose and thus it's inheritence of OpenPoses' failure points.

(a) OpenPose **(b)** Stage 2 **(c)** Stage 3

Figure 3.2.: *Occluded walking failure point, 2 legs predicted instead of 1*

We also found that the choice of axis angle representation for various rotations led to the emergence of common known issue that arising from the discontinuities present in the representation [ZBL+18]. The shortest path between certain angles in axis angle space can lead to a 360 degree rotation, as seen in Figure 3.3, among other issues.

(a) **(b)** **(c)**

Figure 3.3.: *Axis angle issue: dubious rotations*

We found that the TestOps system would occasionally get itself into a tangle, as seen in Figure 3.4, we assume due to the autogregressive nature resulting in a potentially catastrophic cumulation of errors, and found this happended most notably for a sequence containing someone rolling on the floor. It is interesting to note that these sorts of motions were not often, if ever, present in the training data.

(a) OpenPose **(b)** Stage 2 **(c)** Stage 3

Figure 3.4.: *HuMoR in a tangle*

Next, we saw overly smoothed motion for certain clips, most notably for particularly stylised motion such as dancing. We also found that if there is a single frame without Open Pose predictions then as of that frame the TestOps fails. Finally, and most importantly, we found that

the TestOps was extremely slow. It took around 20mins per 2s clip, and that we could batch at most 4 2s clips together, so 20mins per 8s on a Nvidia GeoForce 1080ti with 11Gbs memory.

We concluded that there were a number of potential benifits, notably in occluded situations, and that many of the issues, including the axis angle representation issue, the dependence on humor, and smooth motion, have a good chance of being fixed with different design choices. The glaring issue however was the unresonable amount of time the system took, rendering it unusable for our purposes as we wish the system to be close to real time.

3.1.4. Profiling

To investigate this speed issue, we profiled the code, as can be seen in Figure 3.5. We found that the program spent 90% of it's time in the Stage 3 optimiser closure, 56% of it's time performing the backwards step and 32% of it's time in the rollout function. Hence it was clear that the speed issue was due to the slow act of rolling out and the large computation necessary to perform the backwards step on the large computation graph resulting from the rollout.

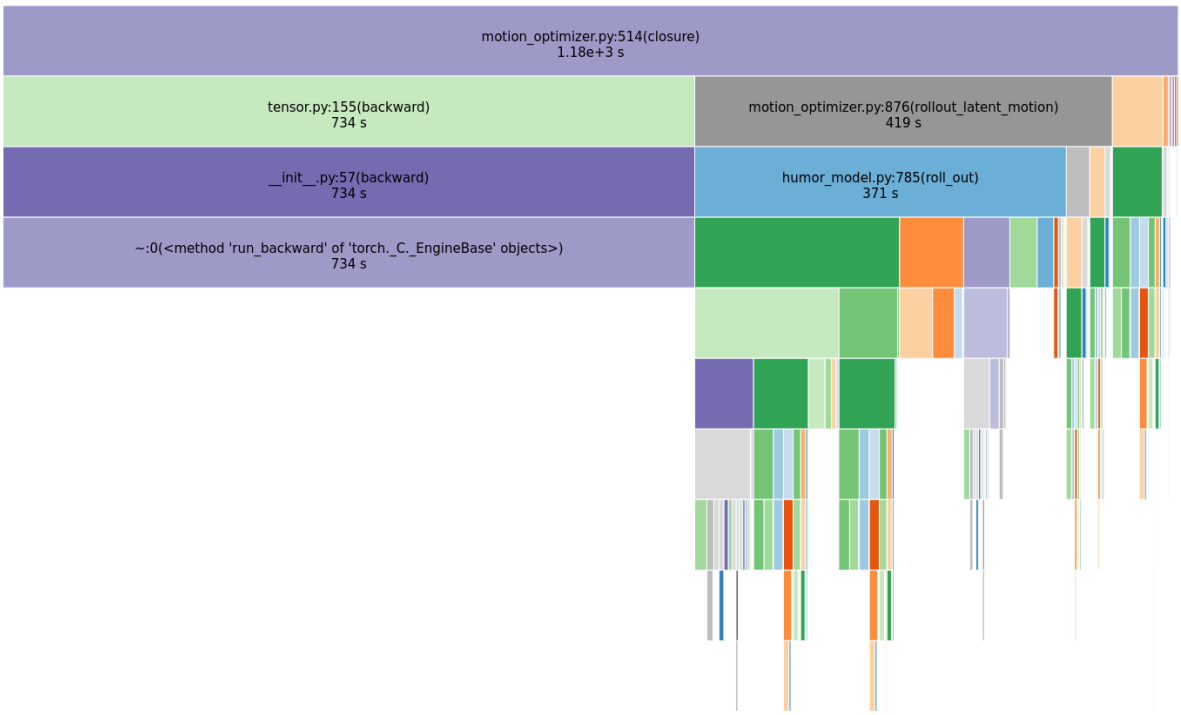


Figure 3.5.: *TestOps profiling*

3.1.5. Conclusion

We came to the final conclusion that the HuMoR motion model alongside the TestOps optimisation showed potential, but that it must be sped up before we can conclusively say if it is of particular use to us. We note that to speedup the method, we must primarily focus on im-

proving/removing the rollout. Thus, in the next section, an attempt to speed up the TestOps is made.

3.2. Improving HuMoR TestOps

This section presents the attempt that was made at modify the TestOps of the HuMoR paper [RBH+21] in such a way that we attain comparable results but in a significantly shorter timespan.

TODO: See if I have introduced the terms properly (decoded sequence, x's, x''s, etc.)

3.2.1. Speeding up the TestOps

Realising that the main speed limitations are due to the concept of rollout over the entire sequence, we decided to break this long range dependence. Through short overlapping rollouts, it is possible to acheive more parrallelisation and smaller computation graphs, and our intuition suggests that the need for such a large context window is not necessary, that only a small number of autoregressed frames are needed to be able to propagate information sensibly and to get meaningful gradients. Anecdotaly, it would seem intuitive that if a video contains sitting, then walking, then sitting again, the second act of sitting would not depend in on the early act of sitting, hence rolling out over the whole sequence seems unnecessary.

The HuMoR TestOps rollout is presented graphically in Figure 3.6. x is the state, and z are the latent variables, the optimised variabels are highlighted in pink, and the losses are coloured, gray x's indicate they are calculated during the optimisation from the optimised variables. As can be seen, first all the states are autoregressed from the initial state and the latent variables, and then the losses are applied. This can be seen to result in very long range dependencies in the computation graph, thus making the gradients time consuming to calculate.

TODO: Update Figure 3.7 and remove the 'NEW' norm, labels on green loss, and potentially red loss?

The most obvious way to break this autogregression is to maintain and optimise a separate sequence of x's, updating this seperate sequence with reference to the x''s decoded from the latent variables as in Figure 3.7. We can take into account the decoded x''s in several manners:

- Copy over
  - Directly replace the optimised sequence of x's with the decoded x''s.
- Blend
  - Perform a weighted addition of the optimised x's and decoded x''s.
- Loss term
  - Add a loss term on the difference between the optimised x's and decoded x''s.

All decoding steps can now be performed in parrallel, rather than sequentially, which greatly

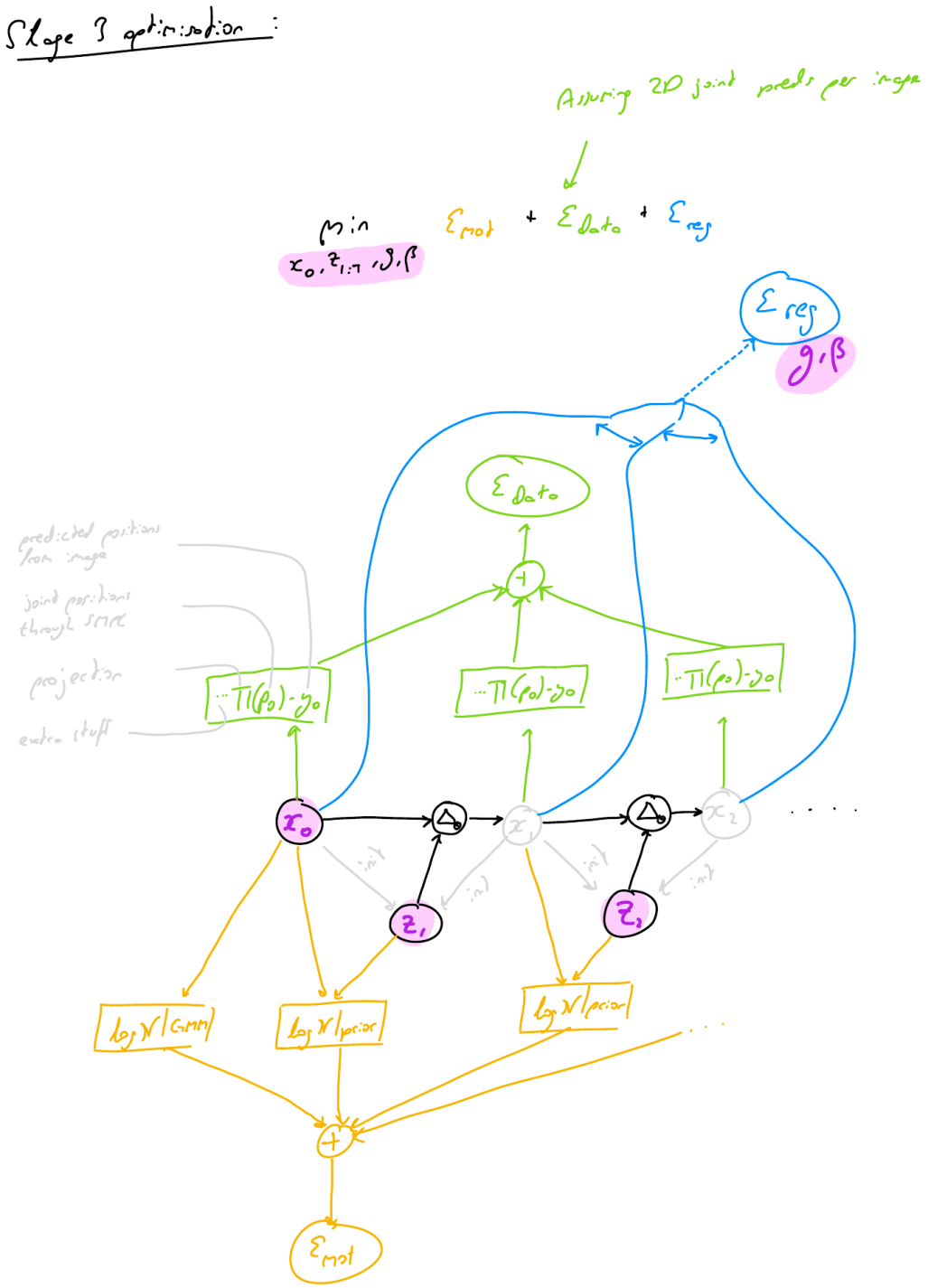


Figure 3.6.: *TestOps Computation Graph*

reduces the computation time.

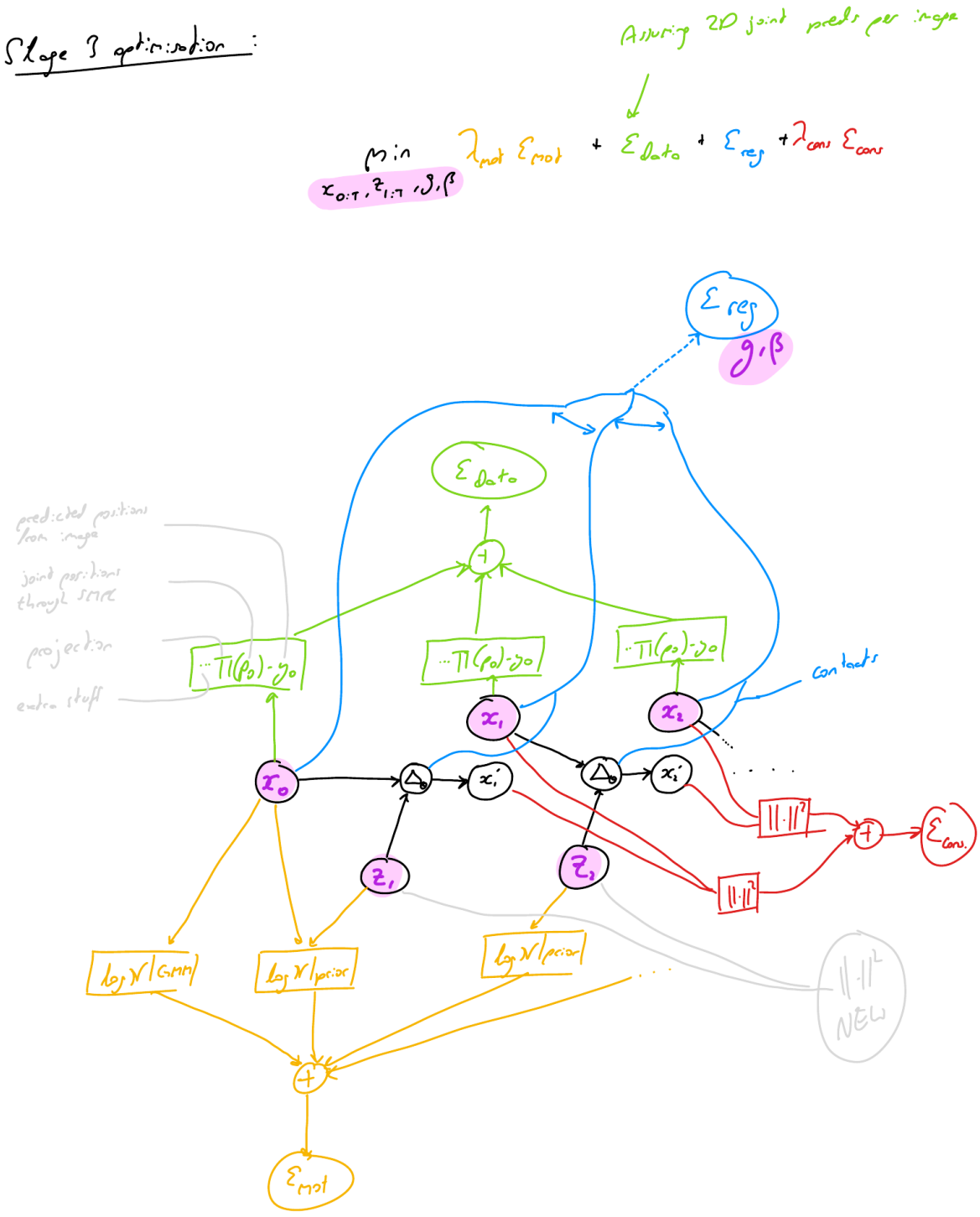


Figure 3.7.: *Decoupled Computation Graph*

This new method also allows us to experiment with different amounts of rollout. We can once again decode the decoded sequence of x''s to get x'', and so on and so forth. These extra sequences are all decoded with the same latent variables thus the updates to the latent variables will take into account the losses on all the decoded sequences. Graphically this can be seen in Figure 3.8, where each next line . TODO: describe the figure better

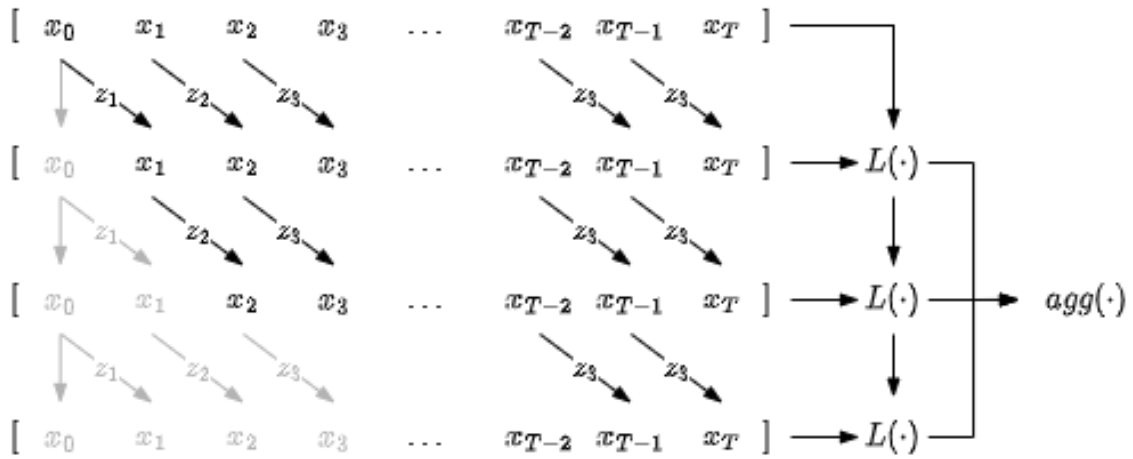


Figure 3.8.: *Decoded sequences*

The information contained in each x can flow forwards through the short range rollouts, and over the iterations of the optimiser. Note however that the information flow will be slower than the HuMoR TestOps, as in the TestOps all subsequent x's are regressed from the initial x_0. In the HuMoR TestOps information can flow backwards from any future frame's x in one update through the gradients, as all x's are linked in the computation graph through the auto-regression (though our intuition suggests that the information does not actually need to flow so far, as previously discussed). In the new method however, the information flow will be much more limited, as the rollouts will be significantly shorter, though again the information should flow through the iterations fo the optimiser.

3.2.2. Implementation notes

A number of issues were encountered during the implementation that are worth mentioning. After implementing the new method, attempts were made to get the optimiser to produce sensible results. The general approach was to use a small number of losses, so as to get a better intuition of each loss, with the goal of better understanding how to balance the losses in the optimiser, but some errors in individual losses were encountered.

Firstly, as expected, the Axis-Angle representation became a problem. When running through functions to convert to root-local reference frame for the HuMoR model for decoding (as it operates on said reference frame), and then back to world reference frame for the opftimiser, the Axis-Angle vector flipped direction. Though the flipped vector still represented the same rotation, the angle representations of the same angle in the optimised sequence of x's and the decoded sequence of x''s were different, hence encorporating the information from decoded

sequence cause the angle to move somewhere between the two flipped values which was no longer a sensible angle representation, thus resulting in root flips and other strange effects.

Secondly, an issue due to the SMPL model was found. When using a 2d reprojection loss (comparing the projected SMPL joints to the OpenPose 2d predictions) on a sequence where only the left arm and face had any OpenPose predictions, the legs were being moved by the optimiser. This was wrong as there were only losses on the arm and face joints, hence no gradients should flow to the legs. It was eventually found that the skinning weights of the SMPL model [LMR+15] contain spurious long range connections. 3% of the LBS (linear blend skinning) weights in the SMPL model are non-zero but less than $1e-2$, which seems rather too low to have any meaningful effect on the skinning, but which allows for spurious gradients to flow.
For the comparison to the OpenPose predictions the OpenPose skeleton must be obtained from the SMPL mesh. Certain joints are regressed, other are simply taken directly as vertices from the mesh, as in the case of the nose joint in the OpenPose skeleton which is taken to be vertex 332 [SMP]. We noted that this vertex 332 is skinned 99.8% by the 'head' joint, but also 0.2% by the right hip, left knee and right knee. This issue was solved by pruning all weights below 1e-2. It is interesting to note that there are known spurious connections in the SMPL blend shapes [OBB20], but that we have found no reference to spurious connections in the LBS weights.

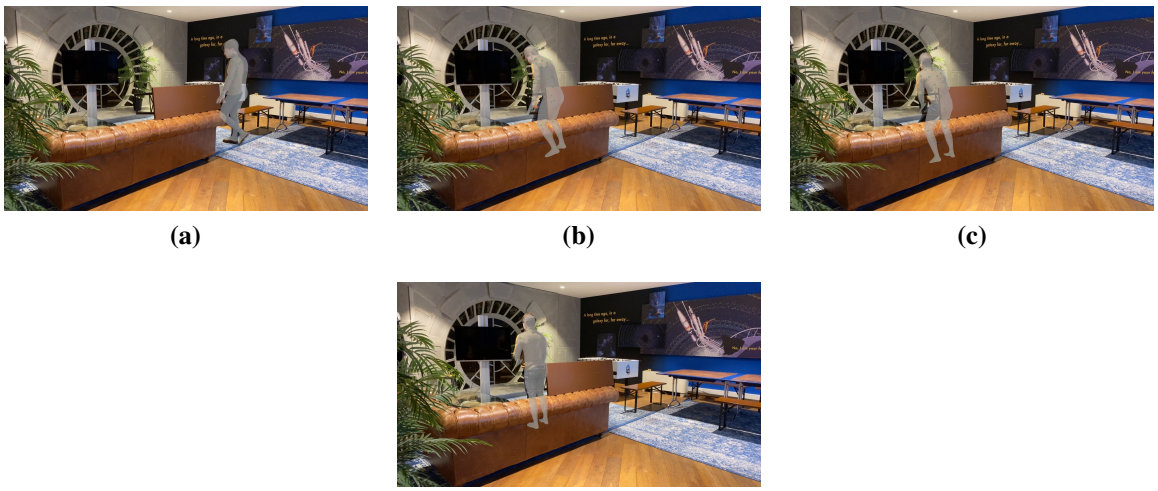
3.2.3. Experiments

TODO: explain why the initial rollout was bad, and the compounding of errors forward TODO: maybe we should discuss more in the humor investagtion section about the bad initial rollout TODO: Check everything is explicitely named, e.g have a graphic clearly naming the optimised x's and the decoded x's, and what the z's are

To begin with, we looked into the initial z's that were obtained from projecting the Stage 2 results through the HuMoR encoder. We found that in occluded situations these z's encode a sitting motion without any optimisation as seen in Figure 3.9, that simply the projection leads them to encode the necessary occluded motion. However we note that when autoregressively rolled out over a longer sequence the results, in Figure 3.11, deviate largely from the initial sequence of x's seen in Figure 3.10. This shows us that though the initial z's encode some sensible motion locally, and can be used to accurately recover the next frame, when they are chained together small deviations from the x motion accumulate to create an overall deviation. For example if the arm is moved slightly too much by a single z, then all the future frames will have the arm in slightly the wrong position, and multiple z's are wrong in the same direction, then the arm will raise up. This deviating sequence is the initial starting rollout in the Stage 3 of the HuMoR TestOps in which the rollout is optimised, though it seems unfortunate to us that from a sensible sequence of x's obtained from Stage 2 we get such a bad sequence through the autoreggresive rollout. We do however note that though while this starting point represents a sequence that deviates largely, the z's might not need to move so far to avoid the accumulation effect. The intuition in our case was that we are starting from some sensible x's and z's, and therefore that it should be possible to refine the x's by take into account the HuMoR model through the z's whilst updating the z's.

With some preliminary experiments in which we began optimising x's and z's we found that

Figure 3.9.: *Stage2 z's encode a sitting motion*



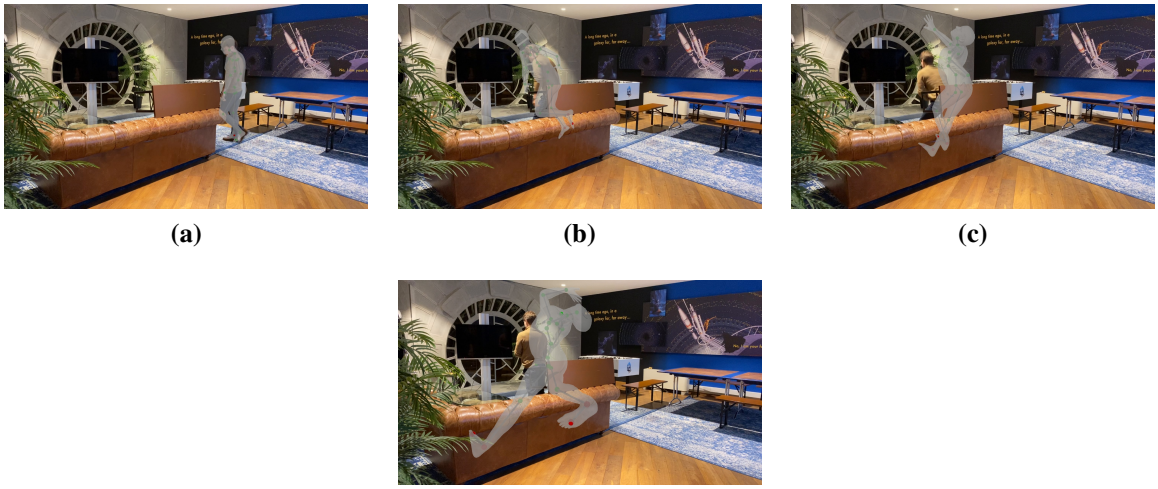
(a) **(b)** **(c)**

(d)

Figure 3.10.: *Stage 2 x's*

the most successful approach to blending the decoded and optimised x's was to have an L1 loss on the distance between them. We saw that when simply copying over, the propagation of small errors forward was too strong (as desribed in Figure 3.11) and the system failed to recover as the long range gradients present in the HuMoR TestOps method weren't present, and we found when blending rather than completely copying over that the weighting of the blending was another parameter that was difficult to choose, hence having a loss was the path of least resistence.

Next we experimented with jointly optimising x and z with a larger variety of losses. It was



(a) **(b)** **(c)**

(d)

Figure 3.11.: *Deviation due to rollout of Stage 2 z's*

noted that the sitting down motion was not acheived consistenly. However as previously mentioned, the initial z's encode a sitting motion, thus we conject that the x's and z's were fighting and this sitting motion was lost, that is to say the z's move to match the x's motion in which the skeleton simply clips into the floor without bending legs, thus the z's begin encoding less leg bending. To avoid this issue in future experiments, the z's were detached from the consistency loss making the optimised x's match the decoded x's.

This led us on to some experiments in which we either optimise z whilst fixing x, or vice versa. With the z's fixed at the stage 2 results, we managed to acheive a sensible sitting motion for a clip containing occluded sitting, but when we applied the same optimiser settings to a longer clip the results were no longer satisfactory. We also noted that when it did work, it required a large number of iterations, in the 1000s, which took 10mins or more, thus the speedup was not as evident as hoped (though our method would continue scaling better). In the inverse case, fixing x to the final HuMoR optimised states and optimising z, we noted that we consistently converge in the direction of the final HuMoR z's, which indicates that we are optimising in the right direction, with the best results acheived at a rollout of 5 decoding steps.

All these experiments were performed with varying numbers of decoding steps, 1, 2, 5, and 10. We found that the effect of more stable results but slower compute times with longer rollouts did manifest as intuition would suggest.

These experiments suggest that the next direction to pursue would be an optimisation scheme in which we flip flop optimising z then x. However considering our difficulty in achieving consistent results for the optimisation of the x's with fixed z's, our feeling that this might not be the optimal solution to the problem, my desire to try a new direction, and the time constraints, we deemed it time to move on and try a new method.

3.2.4. Conclusion

We found that jointly optimising the x's and z's did not prove an easy task to solve, which may indicate that

- it is a particularly difficult problem, and therefore that the optimiser struggles to satisfy the constraints and find a good minimum for both x and z at the same time
- the coupling of the x's and z's in the HuMoR decoder is an important issue. We found it to be at least a minor issue as it was necessary to decouple the z's from the consistency loss (matching the optimised xs to the decoded x''s) to avoid the z's being modified in such a way as to fit the unoptimised x's at the beginning of the optimisation
- the optimiser is badly balanced

This at the very least suggests that a new decoder that does not couple x's and z's would be useful and worth investigating if a similar method is pursued.

In the investigations we found it possible to acheive sensible results on a given clip (for example fixing the unoptimised initial z's, we can a plausible sitting motion in a small video of sitting in occlusion) but that the same settings for the optimiser failed to acheive a good result on an alternative clip. This may indicate that

- again, this is a difficult problem that the optimiser will struggle to solve
- the optimiser is badly balanced

as before.

While many of the issues encountered may simply be due to a badly balanced optimiser, considerable effort was made to mitigate this potential issue, and therefore it is our impression that this problem is at the very least a difficult one. We have improved our intuition, fixed several issues, and understood more about the problem and our goals. We therefore can make a more informed decision with respect to which direction to pursue next.
We found that taking a system that was designed with autogregression in mind, and trying to parrallelise it, proved more difficult than expected. The experiments lead us to beleive that jointly optimising a sequence of poses and latent variables obtained through a single frame decoder may be a more difficult formulation of the problem than others. We therefore conclude that a method in which we model longer motion sequences may be more fruitful.

Conclusion and Outlook

TODO: Chpt 1

Appendix

Appendix intro

A.1. Section

TODO:

Bibliography

[CHS+19] Z. Cao, G. Hidalgo Martinez, T. Simon, S. Wei, and Y. A. Sheikh. Openpose: Realtime multi-person 2d pose estimation using part affinity fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019.

[CSY+22] Xin Chen, Zhuo Su, Lingbo Yang, Pei Cheng, Lan Xu, Bin Fu, and Gang Yu. Learning variational motion prior for video-based motion capture, 2022.

[DKP+23] Yuming Du, Robin Kips, Albert Pumarola, Subastian Starke, Ali Thabet, and Artsiom Sanakeyeu. Avatars grow legs: Generating smooth human motion from sparse tracking inputs with diffusion model, 2023.

[HAB19] Gustav Eje Henter, Simon Alexanderson, and Jonas Beskow. Moglow: Probabilistic and controllable motion synthesis using normalising flows. *CoRR*, abs/1905.06598, 2019.

[HSK16] Daniel Holden, Jun Saito, and Taku Komura. A deep learning framework for character motion synthesis and editing. *ACM Trans. Graph.*, 35(4), jul 2016.

[HSKJ15] Daniel Holden, Jun Saito, Taku Komura, and Thomas Joyce. Learning motion manifolds with convolutional autoencoders. In *SIGGRAPH Asia 2015 Technical Briefs*, New York,NY,USA, 2015. Association for Computing Machinery.

[KBJM17] Angjoo Kanazawa, Michael J. Black, David W. Jacobs, and Jitendra Malik. End-to-end recovery of human shape and pose. *CoRR*, abs/1712.06584, 2017.

[KZFM18] Angjoo Kanazawa, Jason Y. Zhang, Panna Felsen, and Jitendra Malik. Learning 3d human dynamics from video. *CoRR*, abs/1812.01601, 2018.

[LGK20] Zhengyi Luo, S. Alireza Golestaneh, and Kris M. Kitani. 3d human motion estimation via motion compression and refinement. *CoRR*, abs/2008.03789, 2020.

[LMR+15] Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and

Michael J. Black. Smpl: A skinned multi-person linear model. *ACM Trans. Graph.*, 34(6), nov 2015.

[LVC+21] Jiaman Li, Ruben Villegas, Duygu Ceylan, Jimei Yang, Zhengfei Kuang, Hao Li, and Yajie Zhao. Task-generic hierarchical human motion prior using vaes. *CoRR*, abs/2106.04004, 2021.

[LZCvdP21] Hung Yu Ling, Fabio Zinno, George Cheng, and Michiel van de Panne. Character controllers using motion vaes. *CoRR*, abs/2103.14274, 2021.

[MWFD21] Mathieu Marsot, Stefanie Wuhrer, Jean-Sébastien Franco, and Stephane Durocher. Multi-frame sequence generator of 4d human body motion. *CoRR*, abs/2106.04387, 2021.

[OBB20] Ahmed A. A. Osman, Timo Bolkart, and Michael J. Black. STAR: sparse trained articulated human body regressor. *CoRR*, abs/2008.08535, 2020.

[OBH+21] Boris N. Oreshkin, Florent Bocquelet, Félix G. Harvey, Bay Raitt, and Dominic Laflamme. Protores: Proto-residual architecture for deep modeling of human pose. *CoRR*, abs/2106.01981, 2021.

[RBH+21] Davis Rempe, Tolga Birdal, Aaron Hertzmann, Jimei Yang, Srinath Sridhar, and Leonidas J. Guibas. Humor: 3d human motion model for robust pose estimation. In *International Conference on Computer Vision (ICCV)*, 2021.

[SMK22] Sebastian Starke, Ian Mason, and Taku Komura. Deepphase: Periodic autoencoders for learning motion phase manifolds. *ACM Trans. Graph.*, 41(4), jul 2022.

[SMP]

[TCL22] Jonathan Tseng, Rodrigo Castellon, and C. Karen Liu. Edge: Editable dance generation from music, 2022.

[TWH+22] Xiangjun Tang, He Wang, Bo Hu, Xu Gong, Ruifan Yi, Qilong Kou, and Xiaogang Jin. Real-time controllable motion transition for characters. *ACM Transactions on Graphics*, 41(4):1–10, jul 2022.

[YSI+22] Ye Yuan, Jiaming Song, Umar Iqbal, Arash Vahdat, and Jan Kautz. Physdiff: Physics-guided human motion diffusion model, 2022.

[ZBL+18] Yi Zhou, Connelly Barnes, Jingwan Lu, Jimei Yang, and Hao Li. On the continuity of rotation representations in neural networks, 2018.