

# Motion Priors for Pose Estimation and Animation Workflows

Luke Smith

Master Thesis  
May 2023

Prof. Dr. Robert W. Sumner



Eidgenössische Technische Hochschule Zürich  
Swiss Federal Institute of Technology Zurich



computer graphics laboratory



# Abstract

Motion capture, the process of automatically recording the motion of a person or object, has solidified it's position as a key technology in modern animation pipelines due to its numerous benefits. It provides real time, high quality, physically plausible results, however the upfront costs for tailored software and hardware, cumbersome motion capture suits, and other issues leads to a high barrier not only for individual artists but also small teams. At Disney Research|Studios, this barrier is being tackled with a system that aims to capture motion directly from RGB video. The system performs pose estimation on the individual frames, and runs the result through an optimizer to clean up the motion. While promising, the system occasionally struggles with certain artifacts and with some occluded motions. The goal of this thesis is therefore to consider how we might learn a general human motion model that could improve upon these shortcomings. To begin with we investigate a state of the art approach based on Variational Auto Encoders, and attempt to improve upon its issues to make it useable for our purposes. We then consider a relatively new class of models, that of diffusion models, which is proving to be a powerful tool in many domains. We consider the application of such models in the context of motion modelling, and implement and evaluate a baseline model with the view of coming to a go/no-go decision on the feasibility of such models for our purposes.



# Zusammenfassung

Motion Capture, die automatische Aufzeichnung der Bewegung einer Person oder eines Objekts, hat sich aufgrund ihrer zahlreichen Vorteile als Schlüsseltechnologie in modernen Animationspipelines etabliert. Sie liefert qualitativ hochwertige, physikalisch plausible Ergebnisse in Echtzeit, doch die Vorabkosten für maßgeschneiderte Software und Hardware, umständliche Motion-Capture-Anzüge und andere Probleme führen zu einer hohen Hürde nicht nur für einzelne Künstler, sondern auch für kleine Teams. Bei Disney ResearchStudios wird diese Hürde mit einem System angegangen, das darauf abzielt, Bewegungen direkt aus RGB-Videos zu erfassen. Das System führt eine Posenschätzung für die einzelnen Frames durch und lässt das Ergebnis durch einen Optimierer laufen, um die Bewegung zu bereinigen. Das System ist zwar vielversprechend, hat aber gelegentlich Probleme mit bestimmten Artefakten und mit einigen verdeckten Bewegungen. Ziel dieser Arbeit ist es daher zu untersuchen, wie ein allgemeines menschliches Bewegungsmodell erlernt werden kann, das diese Unzulänglichkeiten beseitigt. Zunächst untersuchen wir einen aktuellen Ansatz, der auf Variational Auto Encoders basiert, und versuchen, seine Probleme zu verbessern, um ihn für unsere Zwecke nutzbar zu machen. Anschließend betrachten wir eine relativ neue Klasse von Modellen, die Diffusionsmodelle, die sich in vielen Bereichen als leistungsfähiges Werkzeug erweisen. Wir betrachten die Anwendung solcher Modelle im Zusammenhang mit der Bewegungsmodellierung und implementieren und bewerten ein Basismodell, um eine Entscheidung über die Durchführbarkeit solcher Modelle für unsere Zwecke treffen zu können.



# Contents

<b>List of Figures</b>	<b>vii</b>
<b>1. Introduction</b>	<b>1</b>
<b>2. Related Work</b>	<b>3</b>
2.1. Motion AutoEncoders . . . . .	3
2.2. Motion Diffusion . . . . .	4
<b>3. HuMoR</b>	<b>7</b>
3.1. Relevant Model Details . . . . .	7
3.1.1. Architecture . . . . .	7
3.1.2. TestOps . . . . .	8
3.2. HuMoR Investigation . . . . .	10
3.2.1. Method . . . . .	10
3.2.2. Advantages of HuMoR . . . . .	10
3.2.3. Drawbacks of HuMoR . . . . .	10
3.2.4. Profiling . . . . .	12
3.2.5. Investigation Conclusion . . . . .	13
3.3. Improving HuMoR TestOps . . . . .	13
3.3.1. Speeding up the TestOps . . . . .	13
3.3.2. Implementation notes . . . . .	16
3.3.3. Experiments . . . . .	17
3.3.4. Improvement Conclusion . . . . .	21
<b>4. Motion Diffusion Models</b>	<b>23</b>
4.1. Method . . . . .	23
4.1.1. Diffusion Models . . . . .	23
4.1.2. Disney Motion Diffusion (DMD) . . . . .	24

## *Contents*

4.2. Experiments . . . . .	27
4.2.1. Test Cases . . . . .	27
4.2.2. Baseline Model . . . . .	29
4.2.3. Model with Foot Contacts . . . . .	31
4.2.4. Autocompletion . . . . .	31
4.3. Conclusion . . . . .	32
4.4. Future Work . . . . .	32
4.4.1. Model . . . . .	32
4.4.2. Model use . . . . .	33
4.4.3. Model evaluation . . . . .	33
4.4.4. Data . . . . .	33
<b>5. Conclusion and Outlook</b>	<b>35</b>
<b>A. Appendix</b>	<b>37</b>
A.1. Section . . . . .	37
<b>Bibliography</b>	<b>38</b>

# List of Figures

3.1.	HuMoR C-VAE Architecture [RBH <sup>+</sup> 21] . . . . .	8
3.2.	HuMoR achieving occluded sitting . . . . .	10
3.3.	Occluded walking failure point, 2 legs predicted instead of 1 . . . . .	11
3.4.	Axis angle issue: dubious rotations . . . . .	11
3.5.	HuMoR in a tangle . . . . .	11
3.6.	TestOps profiling . . . . .	12
3.7.	TestOps Stage 3 Computation Graph . . . . .	14
3.8.	Decoupled Computation Graph . . . . .	15
3.9.	Decoded sequences . . . . .	16
3.10.	Stage 2 $\mathbf{z}_{1:T}$ encode a sitting motion . . . . .	18
3.11.	Stage 2 $\mathbf{x}_{0:T}$ . . . . .	18
3.12.	Deviation due to rollout of Stage 2 $\mathbf{z}_{1:T}$ . . . . .	19
3.13.	Optimising $\mathbf{z}_{1:T}$ with fixed $\mathbf{x}_{0:T}$ . . . . .	20
4.1.	DMD Architecture & Denoising Process . . . . .	25
4.2.	Inpainting Procedure . . . . .	27
4.3.	Motion Generation - Baseline model . . . . .	29
4.4.	Occluded Legs Inpainting - Baseline model . . . . .	30
4.5.	Occluded Legs Inpainting - Baseline model . . . . .	30



# Introduction

Motion capture is an integral part of many modern animation pipelines. It can be defined simply as the act of recording, by automatic means, the motion of a person, animal or object. Throughout this thesis we will primarily interest ourselves in the capturing of human motion. The advantages of motion capture, over traditional animation, are numerous and important in their scope. Motion capture allows for

- quasi-real time results
- high quality motion with realistic object interactions
- often reduced costs as compared to hand animation
- complexity that is constant with respect to the motion being captured

however also has it's inherent costs, notably

- upfront costs for tailored software and hardware
- cumbersome motion capture suits
- lengthy setup times and a non-trivial capture systems requiring expert knowledge
- artifacts due to retargeting of skeletons.

These drawbacks mean that the technology is often prohibitive for small teams and individual artists/animations. It is clear though that overcoming these barriers would provide huge benefits and open up the technology to a wide range of new users. For example individual artists could more efficiently prototype and develop animation sequences starting from a self captured motion, and small teams could readily make use of this technology to do more with less.

The important question therefore is simply; **how might we create a motion capture system without such upfront costs, specialized hardware and need for motion capture suits?** At Disney ResearchStudios an approach is being investigated that aims to capture motion directly

## *1. Introduction*

from RGB video. The system currently follows the following steps:

1. record a video of a human motion sequence (potentially from multiple angles)
2. perform per frame pose estimation using a machine learning model
3. run the resultant motion sequence through an optimiser to improve the quality of the motion (and potentially lift to 3d)

and shows great promise, however has a number of drawbacks. The most notable drawback comes from the fact that the per frame pose estimation system does not produce temporally consistent results. The predictions are made independantly per frame, hence the result motion is jittery, and can contain other artifacts. This is the motivation for the introduction of an optimisation system at the end of the pipeline to counteract these artifacts. However it must be noted that, although smoothness is mostly acheived, it does not fix all issues. The optimiser struggles to handle occluded motion sequences, it does not always deal with limb flips (where the left/right leg/arm are predictions are horizontally flipped for a frame), and cannot always fix completely wrong but confident predictions. It is of interest therefore to try and improve this aspect of the pipeline.

The data driven approach proposed in this thesis, in its most abstract form, is simply that of learning a model that understands human motion. Depending on its conception, such a model could be employed in a number of manners; it could be directly applied to the task of rectifying a motion sequence, it could be used as a loss in the existing optimiser, or it could even simply be used to improve upon certain failings of the optimizer as an additional step to the pipeline, such as fixing occluded motion.

This thesis tackles the task of exploring such motion models with a primary goal of shedding light on the most promising model architecture and the most effictive manner in which such a model might be used. We consider a state of the art approach in Chapter 3, evaluate it in Section 3.2 and attempt to improve upon it in Section 3.3. This does not prove as fruitful as hoped and so we move on to a new class of powerful models called diffusion models in Chapter 4. We explore the use of these models in the context of motion modelling in Section 4.1.2 and finally evaluate them on a variety of tasks with a view of better understanding their capabilities in Section 4.2.

# Related Work

The study of synthesising human motion has a long history motivated in no small part by the desire to create realistic and captivating media in the gaming and film industries. Early adaptive methods rely on motion matching [WH97] [Cla] in which interpolation between similar motions from a database of captured motion is performed. This however does not scale well to out of database motion and often generates generic, non stylized motion, though efforts have been made to introduce learned aspects to such methods [HKPP20] to improve upon their shortcomings.

**TODO: MORE?**

More recent branches of motion modelling commonly base themselves upon the use of machine learning techniques, notably deep learning, to learn a prior over plausible motion. This is a more general approach that can be applied to a wider range of tasks, and shows promise in overcoming some of the issues of motion inbetweening, as such systems can learn to better generalise to out of distribution motion sequences.

Within the area of deep learning, many techniques have been investigated, temporal convolutions [HSK16a], recurrent models [HYNP20], and reinforcement learning [CKP<sup>+</sup>21] are but a few examples.

## 2.1. Motion AutoEncoders

A well explored model is that of the AutoEncoder (AE) [BKG21] or Variation-AutoEncoder (VAE) [KW22]. These are popular models as they encourage the learning of a latent representation [BKG21] of human motion, thus the intuition is that they learn not just to reproduce the data, but actually how humans move, providing a more robust prior.

**TODO: MORE? c.f MEVA for nice related work section**

## 2. Related Work

Holden et al. [HSKJ15] [HSK16b] present simple CNN based autoencoder architectures that operates on motion sequences. The notion of skeletal aware convolutions and pooling/unpooling operations for a VAE, alongside a sliding window method for motion rectification are presented by the authors of [LVC<sup>+</sup>21]. [CSY<sup>+</sup>22] presents a novel approach of leaning a latent space, then projecting directly to this latent space from a motion sequence using a separate model, again operating directly on a motion sequence. The authors of MEVA [LGK20] postulate that a VAE often learns only smooth motion, as we are asking too much of the model, thus present a pipeline in which a smooth motion and coarse motion VAEs are jointly used. Holden et. al make another appearance with DeepPhase [SMK22], an autoencoder with a latent space enforced to match sinusoidal functions that represent periodic motion. Contrary to the common trope of sequence level models, the authors of [TWH<sup>+</sup>22] and of [LZCvdP21] operate in a frame to frame regime, predicting temporally local change of motion. Finally, a number of works present the Conditional-VAE architecture [SLY15] as a base with varying state representations, conditioning variables and loss terms, [RBH<sup>+</sup>21, TWH<sup>+</sup>22, LZCvdP21, MWFD21].

As we can see the literature is rich and diverse, but we found ourselves drawn to the HuMoR model [RBH<sup>+</sup>21], due to it's state of the art performance and it's described use for the exact task that we desire to solve, that of rectifying a motion sequence captured through frame by frame pose estimation. The authors of [RBH<sup>+</sup>21] present a C-VAE architecture that learns a distribution over latent transitions, conditioned on the previous pose. They use this architecture alongside an optimisation method that rectifies human motion obtained from, among other modalities, RGB video through frame-by-frame pose estimation.

## 2.2. Motion Diffusion

With the notable success of diffusion models in the image generation literature [HJA20, DN21, RBL<sup>+</sup>21], diffusion models have begun to spread into many other fields within machine learning [YZS<sup>+</sup>23], including to the field of human motion modelling.

The authors of Avatars grow legs [DKP<sup>+</sup>23] denoise a sequence of SMPL [BKL<sup>+</sup>16] parameters condition on sparse tracking inputs, notably taking the form of the orientation/translation of a headset and two hand controllers. They show that plausible motion can be generate from very sparse signals, thus indicating to us that the use of diffusion models in the rectification of occluded motion sequences is promising. Next, PhysDiff [YSI<sup>+</sup>22] provides a text conditioned diffusion model with the unique use of a physics based motion projection step in the diffusion process that helps to ensure physical plausibility of the generated motion. The authors of EDGE [TCL22] propose an attention based, audio conditioned diffusion framework for dance motion generation. With a similar architecture to that of EDGE [TCL22] but using transformers as the base of the denoising network such that the attention mechanism can more easily be exploited across the whole motion sequence, the authors of MDM [TRG<sup>+</sup>22] describe a text conditioned denoising architecture.

As we have seen, diffusion models can be employed in the field of motion modelling for a wide variety of tasks. Motion can be generated [TRG<sup>+</sup>22, TCL22, DKP<sup>+</sup>23] conditioned on various inputs. Motion sequences can also be edited through inpainting [LDR<sup>+</sup>22, TRG<sup>+</sup>22] to change only parts of a sequence, and motion inbetweening can also be achieved in a similar

manner [TRG<sup>+</sup>22]. This wide variety of tasks that can be completed by a single model is a very attractive property of the diffusion framework, and inspired us to investigate these models later in the thesis.



# HuMoR

The authors of HuMoR [RBH<sup>+</sup>21] present a novel approach for learning and using a plausible motion prior. They train a conditional VAE that learns a distribution over latent transitions from a given state, and can decode a sample from this distribution to a change of state. They notably use this model as a prior in a 'test time optimisation' (TestOps), which generates plausible sequence motions optimising for an initial state and a sequence of transitions starting from frame by frame estimates. The TestOps can operate on many modalities, 2D/3D joints, point clouds, etc., as the optimisation loss contains a Data Term  $\epsilon_{data}$  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 performance of HuMoR as described in the paper [RBH<sup>+</sup>21], alongside it's use in a problem that directly matches our own (TestOps operating on RGB video) lead us to evaluate and investigate the model Section 3.2, and subsequently try to extend and improve 3.3 upon it once the limitations made themselves known.

## 3.1. Relevant Model Details

A brief overview of the salient features of the HuMoR model and TestOps optimisation procedure is provided here, some details are ommited for clarity and brevity.

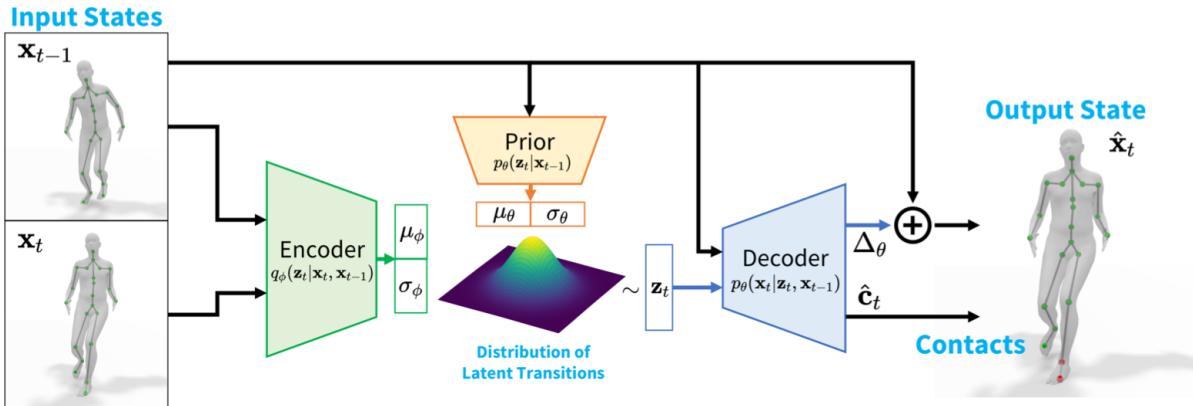
### 3.1.1. Architecture

The HuMoR architecture is that of a C-VAE [SLY15], as can be seen in Figure 3.1. The goal of this architecture is to learn a mapping from a given pose ( $\mathbf{x}_{t-1}$ ) to a distribution over latent

### 3. HuMoR

transitions to the next pose (the **prior**), and a mapping from a latent transition sampled from this distribution to a change in pose (**decoder**) that can be used to obtain the next state . This is achieved during training through an additional **encoder** that has the full information of the next state ( $\mathbf{x}_t$ ) also available to it, such that it can find the ideal latent transition, and therefore is used to guide the training of the **prior**.

The networks operate on states,  $\mathbf{x}_t$ , consisting of a root translation, root orientation, 3D joint positions, and all their respective velocities, and additionally joint angles. This rich and redundant state representation is a superset of the SMPL models' state [BKL<sup>+</sup>16] (given some shape parameters  $\beta$  required by SMPL).



**Figure 3.1.: HuMoR C-VAE Architecture [RBH<sup>+</sup> 21]**

#### 3.1.2. TestOps

**TestOps** is the term given by the authors of HuMoR to their method of using the HuMoR model in an optimisation procedure to obtain a smooth motion estimation from, among other things, an RGB video (our focus).

The concept of an autoregressive 'rollout' is important here. As we can see in Figure 3.1, the model output can be used to obtain a new state. This new state can then be autoregressively fed back into the **decoder**, alongside either a given latent transition or one obtain through the **prior**, to obtain the next state. The procedure of starting from a given state  $\mathbf{x}_0$ , and with a given sequence of latent transitions  $\mathbf{z}_{1:T}$ , obtaining a sequence of states  $\mathbf{x}_{1:T}$  through the autoregressive application of the HuMoR model is called a **rollout**.

The TestOps procedure is as follows for RGB video. First, frame by frame pose estimation is performed using OpenPose [CHS<sup>+</sup>19] to obtain a 2d pose estimates. Next, a three stage optimisation procedure is proposed. The first two stages are used to obtain a data that can be used to get a rough estimate of input to the final stage, being an initial state  $\mathbf{x}_0$  and a sequence of latent transitions  $\mathbf{z}_{1:T}$ . The third stage then refines this estimate using the HuMoR model and outputs a polished sequence of states  $\mathbf{x}_{0:T}$ .

## Stage 1

The global root translation and orientation (in 3d space) is optimised using a loss comparing their projection to the 2d pose estimates. This is a small subset of the full state, and is optimised first so that we don't try and do too much, too soon.

## Stage 2

Next, the joint positions and angles are optimised (without the root) through the use of a projection loss to the 2d pose estimates, and of a number of regulariser losses, including smoothness and foot contact losses. An additional pose prior loss is included in this stage that ensures plausible poses using the VPoser [PCG<sup>+</sup>19] model.

## Stage 3 (Initialisation)

Using the root translation and orientation from Stage 1, and the joint positions from stage 2, the velocities of each of these elements can be calculated through finite differences, and all the variables, alongside the joint angles, can be considered the initial estimate  $\mathbf{x}_{0:T}$  for the sequence of states. The initial estimates  $\mathbf{x}_{0:T}$  are then fed through the **encoder** network to obtain initial estimates of the state transitions  $\mathbf{z}_{1:T}$ .

## Stage 3 (Optimisation)

The final optimisation then operates on a starting state  $\mathbf{x}_0$ , and a sequence of latent transitions  $\mathbf{z}_{1:T}$ . For each optimisation step, a rollout (as described above) is performed to obtain a sequence of states  $\hat{\mathbf{x}}_{1:T}$ , and various losses are applied to these states. The regularisers as described in Section 3.1.2 are applied, a reprojection loss to the 2d pose estimates is used, and finally the **prior** is used to evaluate the likelihood of each transition given the previous state. These losses are summed and the gradients are propagated back to the optimisation variables,  $\mathbf{x}_0$  and  $\mathbf{z}_{1:T}$ . This procedure is repeated for a number of steps, and a sequence of states  $\hat{\mathbf{x}}_{0:T}$  is returned after a final rollout.

## Extra Notes

The computation graph entailed by this Stage 3 procedure is discussed later and can be seen in Figure ??.

Please note that some extra parameters are also optimised during this whole procedure, notably the ground plane and SMPL shape parameters, and a loss on the initial state  $\mathbf{x}_0$  using a learned gaussian mixture model is applied during stage 3. The details however are not included as they are not vital to the investigation nor to the understanding of the important components of the method.

### 3. HuMoR

## 3.2. HuMoR Investigation

### 3.2.1. 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 succeeded, and if the mentioned benefits in the HuMoR paper [RBH<sup>+</sup>21] were as described. Second was to evaluate the model with the defects of the current Disney Research|Studios system in mind to see if it could complement it's functionality, notably if it could improve upon occluded motion, joints flipping and confident but false predictions.

To achieve this goal, a selection of videos were taken in the Disney Research|Studios lab containing a variety of motion; fast, slow, abnormal, and with occlusions. The HuMoR system was ran on these videos and the results were investigated. The TestOps is performed in 3 stages as described in Section 3.1.2, 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 as an improvement over a more classical optimisation method.

### 3.2.2. Advantages of HuMoR

We see a number of situations in which the model shows a clear improvement over stage 2 where the HuMoR model is not used.

In an occluded situation where the 2d pose predictions don't have any information on the legs, the model manages to produce a realistic sitting motion, as shown in Figure 3.2.



**Figure 3.2.:** HuMoR achieving occluded sitting

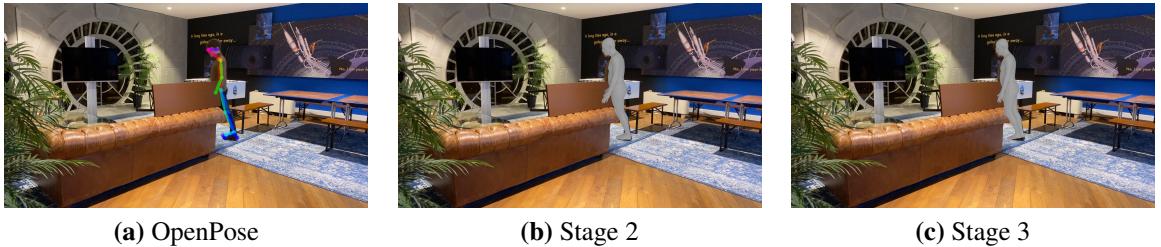
We also note in many of the less difficult videos, that HuMoR produces clean motion and deals with many abnormal movements without obvious issues. It therefore doesn't seem to regress the easy situations, but can improve the more difficult situations, notably occlusions.

### 3.2.3. 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 3.3. This results in a

### 3.2. HuMoR Investigation

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 largely due to a dependence on OpenPose and thus on an inheritance of OpenPoses' failure points.



**Figure 3.3.:** 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 arise from the discontinuities present in the representation [ZBL<sup>+</sup>18]. The shortest path between certain angles in axis angle space enforced by the smoothness regularizer can lead to a 360 degree rotation, as seen in Figure 3.4, among other issues.



**Figure 3.4.:** Axis angle issue: dubious rotations

We found that the TestOps system would occasionally get itself into a tangle, as seen in Figure 3.5, we assume due to the autoregressive nature resulting in a potentially catastrophic cumulation of errors, and found this happened 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, and so it is likely that the model can fail catastrophically in many more situations, provided that they are sufficiently different from the training data.



**Figure 3.5.:** HuMoR in a tangle

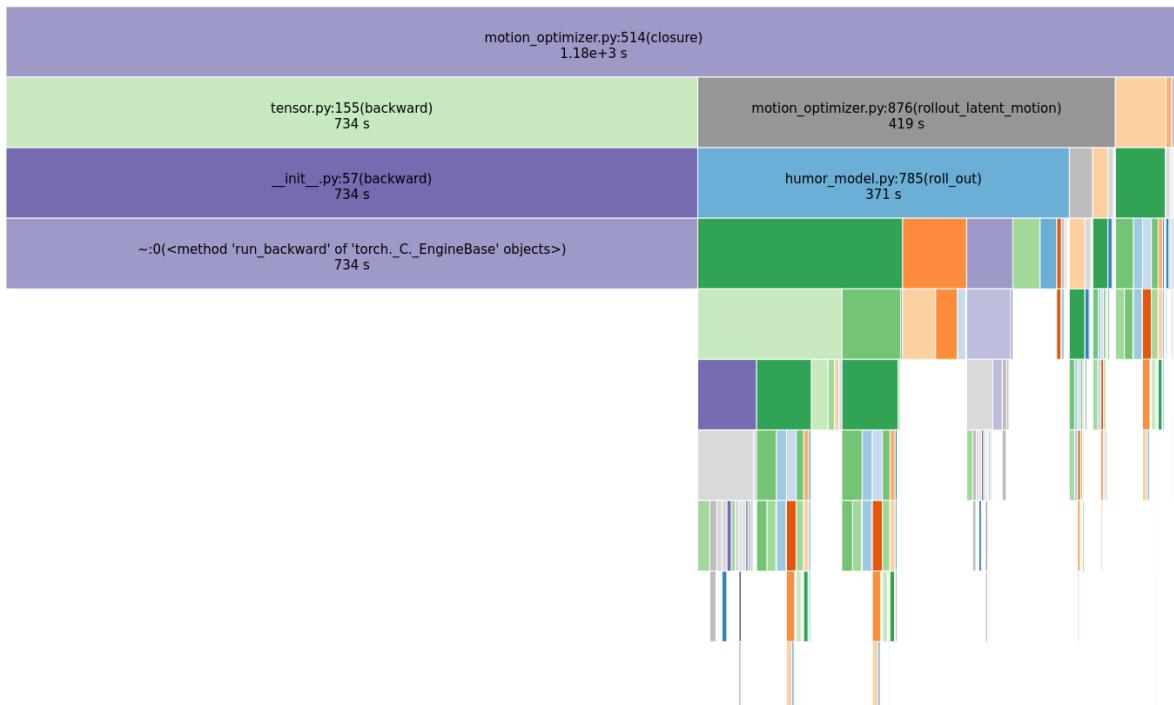
### 3. HuMoR

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 GeForce 1080ti with 11Gbs memory.

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

#### 3.2.4. Profiling

To investigate this speed issue, we profiled the code, as can be seen in Figure 3.6. We found that the program spent 90% of its time in the Stage 3 optimiser closure, 56% of its time performing the backwards step and 32% of its time in the rollout function. Hence it was clear that the speed issue was due to the slow act of rolling out in stage 3 and the large computation necessary to perform the backwards step on the enormous computation graph resulting from the rollout.



**Figure 3.6.:** TestOps profiling

### 3.2.5. Investigation 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 improving/removing the rollout of stage 3. In the next section, an attempt to do just this is presented.

## 3.3. Improving HuMoR TestOps

This section presents the attempt that was made at modifying the TestOps procedure in such a way that we attain comparable results but in a significantly shorter timespan.

### 3.3.1. 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 achieve more parallelisation 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. Anecdotally, it would seem intuitive that if a video contains sitting, then walking, then sitting again, the second act of sitting would not depend on the early act of sitting, hence rolling out over the whole sequence seems unnecessary.

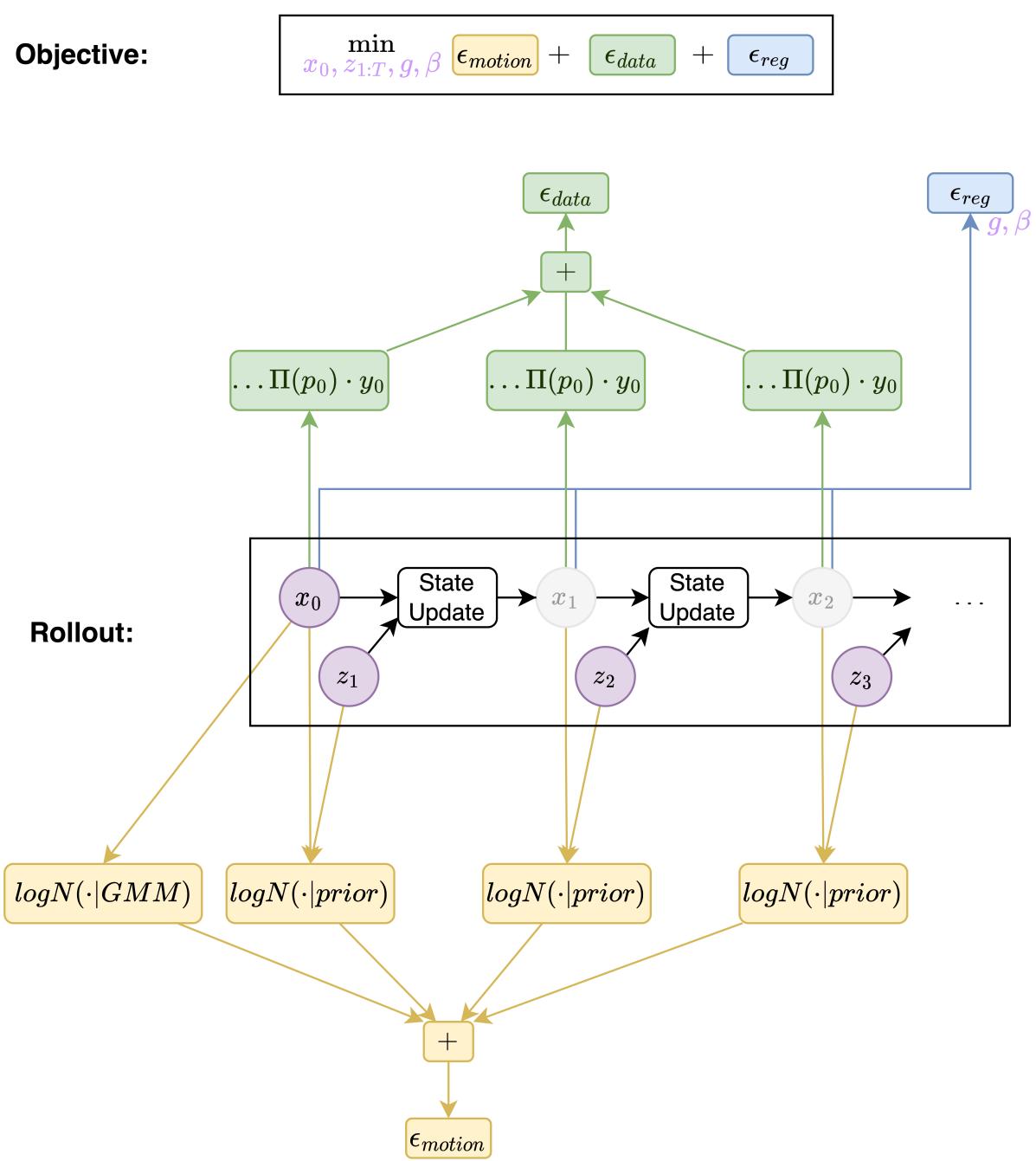
The HuMoR TestOps rollout is presented graphically in Figure ???. To begin with **all** the states are autoregressed from the initial state and the latent variables, and only then are the losses applied. This results in very long range dependencies in the computation graph, thus making the gradients time consuming to calculate.

The most obvious way to break this autoregression is to maintain a separate persistent sequence of states,  $\mathbf{x}_{1:T}$ , updating these with reference to the states decoded from the latent variables. We can take into account the decoded states,  $\mathbf{x}'_{1:T}$ , in several manners (using optimisation techniques or direct updates):

1. Copy over
  - Directly replace the persistent sequence of  $\mathbf{x}_{1:T}$  with the decoded sequence  $\mathbf{x}'_{1:T}$ .
2. Blend
  - Perform a weighted addition of the persistent  $\mathbf{x}_{1:T}$  and decoded  $\mathbf{x}'_{1:T}$ .
3. Loss term
  - Add a loss term on the difference between the persistent  $\mathbf{x}_{1:T}$  and decoded  $\mathbf{x}'_{1:T}$ .

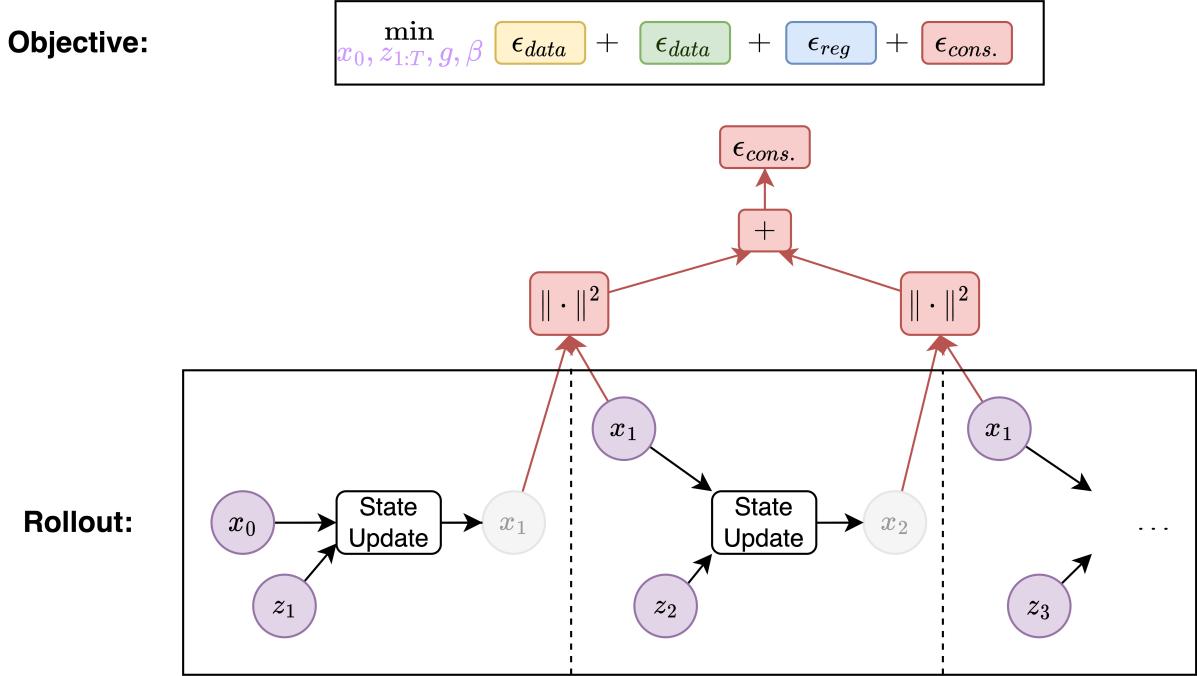
In Figure ?? we can see the example of adding a loss term on the difference between the persistent  $\mathbf{x}_{1:T}$  and decoded  $\mathbf{x}'_{1:T}$ .

### 3. HuMoR



**Figure 3.7.: TestOps Stage 3 Computation Graph**

The optimised variables are highlighted in pink,  $x$  being the state (root, joint positions etc.), and  $z$  the latent transitions. The gray  $x$ 's indicate the states that are calculated during the rollout (c.f Section 3.1.2) and the losses that are applied after the rollout are color coded.  $\epsilon_{motion}$  is the loss due to the HuMoR prior (and initial state likelihood GMM briefly mentioned in Section 3.1.2).  $\epsilon_{data}$  is the projection loss comparing the projected state to the OpenPose predictions.  $\epsilon_{reg}$  are the regularisation losses, smoothness, foot contacts, etc (and are also the losses responsible for the ground plane  $g$  and the shape parameters  $\beta$  briefly mentioned in Section 3.1.2). The 'State Update' node represents the application of the HuMoR decoder to the previous state and latent transition to obtain a pose update, and the application of this pose update to obtain a new pose.



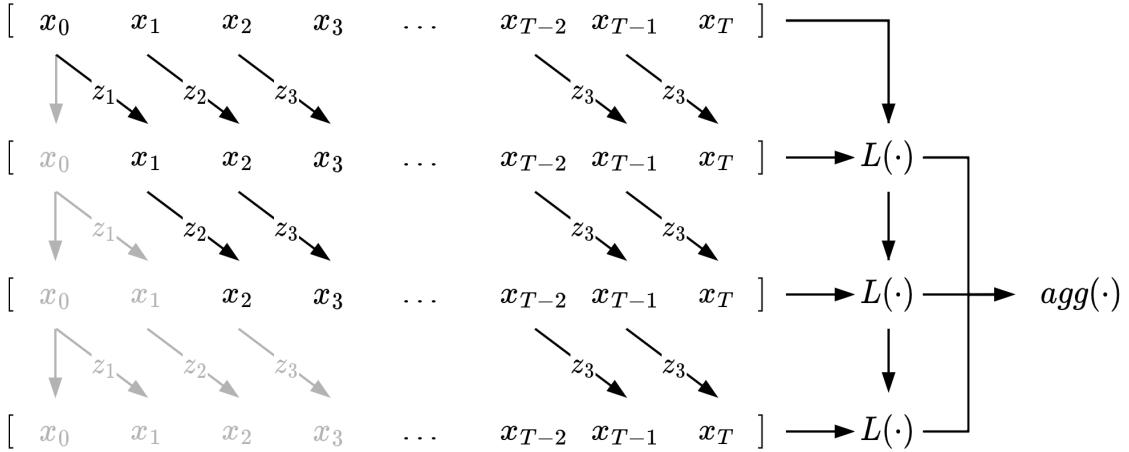
**Figure 3.8.: Decoupled Computation Graph**

The computation graph is similar to that presented in Figure ???. This time however we rollout one step, maintain a persistent sequence of states,  $x_{1:T}$ , and add a loss term,  $\epsilon_{consistency}$  on the difference between the persistent  $x_{1:T}$  and decoded  $x'_{1:T}$ . This theoretically allows for parallelized decoding of the latent variables, and thus a much faster computation time, by avoiding the rollout and large computation graph.

The motivation behind this change is the idea that all decoding steps can now be performed in parallel, rather than sequentially, which greatly reduces the compute time. The maintaining of a persistent sequence of states,  $x_{1:T}$ , allows for the information to 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  $x_0$  and thus the chain is unbroken from beginning to end of the sequence every single rollout (though as discussed in Section 3.3.1 we don't believe this to be necessary). In the HuMoR TestOps information can flow backwards from any future frame  $x_t$ , as the computation graph links all the states through the auto-regression, in the new method however, the information flow will be much more limited, as the rollouts will be significantly shorter, though the information can still flow through the iterations of the optimiser.

This new formulation also allows us to experiment with different degrees 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 ???, where each next line is a newly decoded sequence derived from the previous.

### 3. HuMoR



**Figure 3.9.: Decoded sequences**

The formulation of the optimisation problem shown in Figure ?? can be extended by subsequent rollout. Each time we can reapply the latent variables  $\mathbf{z}_{1:T}$  to obtain a new sequence. In applying losses to these sequences, we can update the latent variables taking into account all the decoded sequences, thus can more quickly propagate information backward through the introduction of a limited rollout. Note that the grayed out variables are redundant and hence ignored.

### 3.3.2. 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 decoder (as it operates on said reference frame), and then back to world reference frame for the optimiser, the Axis-Angle vector flipped direction. Though the flipped vector still represented the same rotation, the numerical values of the axis angle representations in the persistent sequence of  $\mathbf{x}_{0:T}$  and the decoded sequence of  $\mathbf{x}_{0:T}$  were now different, hence incorporating the information from the decoded sequence in the persistent sequence caused the angle to find a middle point between the flipped values which was no longer a sensible angle representation, thus resulting in strange rotations and artifacts in the final sequence. We mitigated this issue by optimising in 6d rotation representation.

Secondly, an issue in the SMPL model was found. When using a 2d reprojection loss (comparing the projected 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 seemed dubious as the losses were only seeing the arm and face joints, hence no gradients should have flowed to the legs. It was eventually found that the skinning weights of the SMPL model [BKL<sup>+</sup>16] 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 example the vertex 332 of the SMPL mesh [SMP], representing the nose joint which we need for a comparison to the OpenPose skeleton, is skinned 99.8% by the 'head' joint, but also spuriously 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.

### 3.3.3. Experiments

To begin with, we looked into the initial  $\mathbf{z}_{1:T}$  that was obtained from projecting the Stage 2 results through the HuMoR encoder. We found that in occluded situations  $\mathbf{z}_{1:T}$  already encodes a sitting motion without any optimisation as seen in Figure 3.10, that simply the projection leads them to encode the necessary occluded motion, hence represent a meaningful starting point. However we note that when autoregressively rolled out over a longer sequence the results, in Figure 3.12, deviate largely from the initial sequence of  $\mathbf{x}_{0:T}$  seen in Figure 3.11. This shows us that though the initial  $\mathbf{z}_{1:T}$  encode some sensible motion locally, and can be used to accurately recover the next frame, when they are chained together small errors accumulate to create an overall divergence. For example if the arm is moved slightly too much by a single  $\mathbf{z}_t$ , then all the future frames will have the arm in slightly the wrong position, and if multiple latent variables  $\mathbf{z}_{t:t'}$  are wrong in the same direction, then the arm will raise up. This diverging sequence is the initial starting rollout in the Stage 3 of the HuMoR TestOps in which the rollout is optimised, and it seems unfortunate to us that from a sensible starting sequence  $\mathbf{x}_{0:T}$  obtained from Stage 2 we get such a bad sequence through the autoregressive rollout. We do however note that while this starting point represents a sequence that diverges heavily, the  $\mathbf{z}_{1:T}$  might not need to move so far to avoid such accumulation effects. The intuition that motivates our formulation is the fact that we are not completely throwing out the sensible starting point  $\mathbf{x}_{0:T}$  and  $\mathbf{z}_{1:T}$  obtained from Stage 2, and therefore that it should be possible to refine the  $\mathbf{x}_{0:T}$  by take into account the HuMoR model through the  $\mathbf{z}_{1:T}$  whilst updating the  $\mathbf{z}_{1:T}$  in a more efficient manner than simply discarding these  $\mathbf{x}_{0:T}$  through an initial rollout.

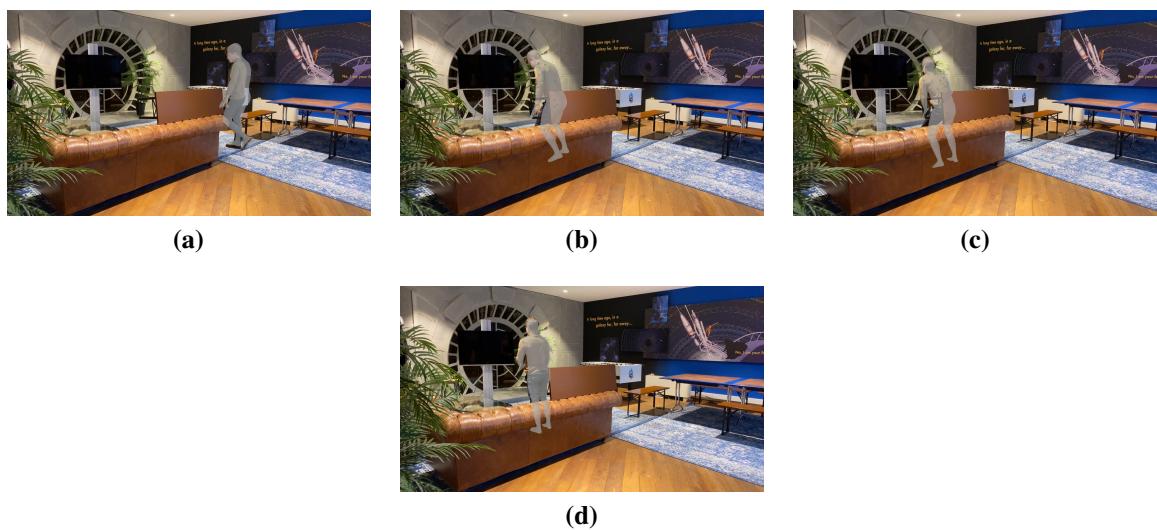
With some preliminary experimentation in which we began optimising  $\mathbf{x}_{0:T}$  and  $\mathbf{z}_{1:T}$ , we found that the most successful approach to blending the decoded and persistent  $\mathbf{x}_{1:T}$ , among those described in Section 3.3.1, was to have an L1 loss on the difference between them. We saw that when simply copying over, the propagation of small errors forward was too strong (as described in Figure 3.12) and the system failed to recover as the long range gradients present in the HuMoR TestOps method that could tell early states that their errors resulted in a large divergence 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 resistance.

Next we experimented with jointly optimising  $\mathbf{x}_{0:T}$  and  $\mathbf{z}_{1:T}$  with a larger variety of losses. It was noted that the sitting down motion was not achieved consistently. However as previously mentioned, the initial  $\mathbf{z}_{1:T}$  encode a sitting motion, thus we conject that the  $\mathbf{x}_{0:T}$  and  $\mathbf{z}_{1:T}$  were fighting and this sitting motion was lost, that is to say the  $\mathbf{z}_{1:T}$  move to match the initial  $\mathbf{x}_{0:T}$

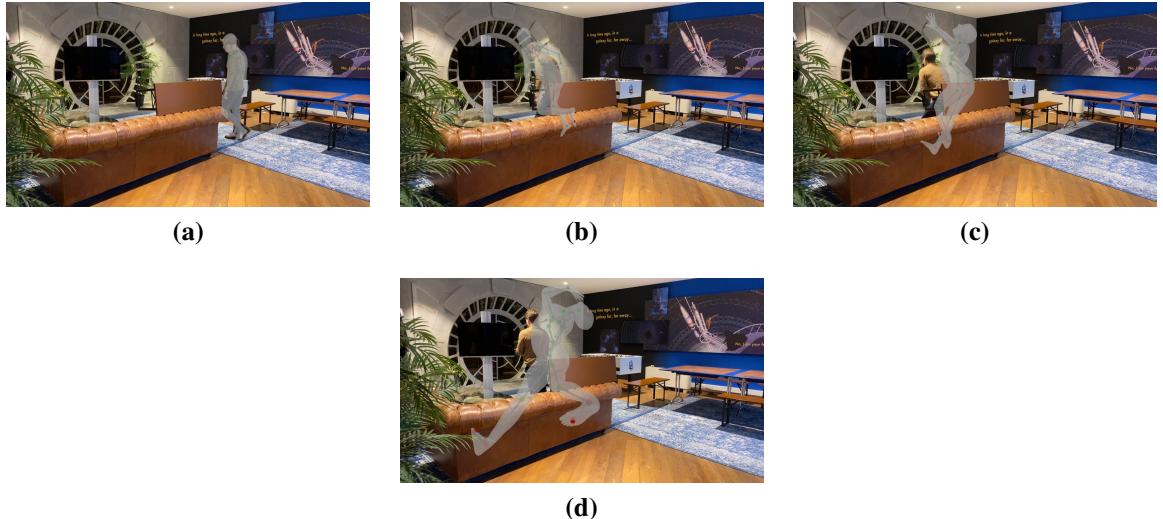
### 3. HuMoR



**Figure 3.10.:** Stage 2  $\mathbf{z}_{1:T}$  encode a sitting motion



**Figure 3.11.:** Stage 2  $\mathbf{x}_{0:T}$



**Figure 3.12.**: Deviation due to rollout of Stage 2  $\mathbf{z}_{1:T}$

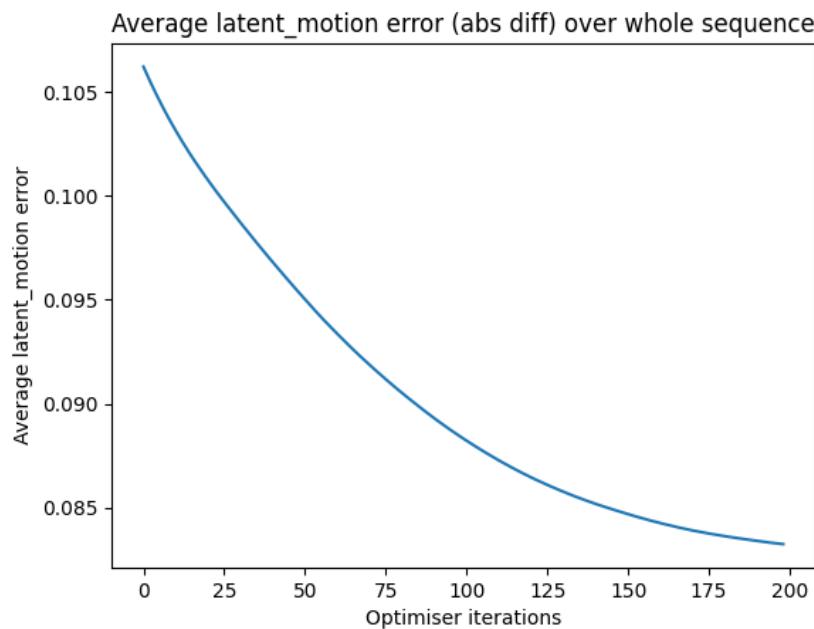
motion in which the skeleton simply clips into the floor without bending the legs, thus the  $\mathbf{z}_{1:T}$  begin encoding less leg bending. To avoid this issue in future experiments, the  $\mathbf{z}_{1:T}$  were detached from the consistency loss that encourages the persistent  $\mathbf{x}_{0:T}$  match the decoded  $\mathbf{x}_{0:T}$ .

This led us on to some experiments in which the goal was either to optimise  $\mathbf{x}_{0:T}$  whilst keeping  $\mathbf{z}_{1:T}$  fixed, or vice versa, to avoid the competing and see if information was passed between latent transitions and poses sensibly. With the  $\mathbf{z}_{1:T}$  fixed at the stage 2 results, we managed to achieve 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 due to its parallelizable nature). In the inverse case consisting of fixing  $\mathbf{x}_{0:T}$  to the final HuMoR optimised states and optimising  $\mathbf{z}_{1:T}$ , we noted that we consistently converge in the direction of the final HuMoR  $\mathbf{z}_{1:T}$  states, which indicates that we are optimising in the right direction, note that this was achieved with a rollout of 5 decoding steps as described in Figure ??.

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

These experiments suggest that the next direction to pursue would be an optimisation scheme in which we flip flop optimising  $\mathbf{z}_{1:T}$  then  $\mathbf{x}_{0:T}$ , as when jointly optimising we found competing goals and a difficult to balance optimiser, but when separated we found that the information was able to propagate as we desired. However considering our difficulty in achieving consistent results for the optimisation of the  $\mathbf{x}_{0:T}$  with fixed  $\mathbf{z}_{1:T}$ , 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. HuMoR



**Figure 3.13.:** Optimising  $\mathbf{z}_{1:T}$  with fixed  $\mathbf{x}_{0:T}$

With  $\mathbf{x}_{0:T}$  fixed to be the final HuMoR TestOps solution, we found that the  $\mathbf{z}_{1:T}$  converges in the direction of the  $\mathbf{z}_{1:T}^{(HuMoR)}$  found by the HuMoR TestOps ('latent\_motion error' is the average absolute difference between these quantities). This indicates that our method also considers the HuMoR solution to be sensible, and gives us hope that it is possible to arrive at the same solution with a faster method.

### 3.3.4. Improvement Conclusion

We found that jointly optimising the  $\mathbf{x}_{0:T}$  and  $\mathbf{z}_{1:T}$  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  $\mathbf{x}$  and  $\mathbf{z}$  at the same time
- the coupling of the  $\mathbf{x}_{0:T}$  and  $\mathbf{z}_{1:T}$  in the HuMoR decoder is an important issue. We found it to be at the very least a minor issue as it was necessary to decouple the  $\mathbf{z}_{1:T}$  from the consistency loss (matching the persistent  $\mathbf{x}_{1:T}s$  to the decoded  $\mathbf{x}'_{1:T}$ ) to avoid the  $\mathbf{z}_{1:T}$  being modified in such a way as to fit the unoptimised  $\mathbf{x}_{0:T}$  at the beginning of the optimisation schedule
- the optimiser is badly balanced

This suggests that a new decoder that does not couple  $\mathbf{x}_{0:T}$  and  $\mathbf{z}_{1:T}$  would be useful and worth investigating if a similar method is pursued in the future.

In the investigations we found it possible to achieve sensible results on a given clip when optimising either the  $\mathbf{x}_{0:T}$  or the  $\mathbf{z}_{1:T}$  separately. For example fixing the unoptimised initial  $\mathbf{z}_{1:T}$ , we produce a plausible sitting motion in a short occluded sequence. However we find that same settings for the optimiser failed to achieve 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 tricky one in the formulation we were investigating. While no exiting concrete results were produced, we did manage to improve our intuition, understand and fix several issues, and gain more insight into 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 autoregression in mind, and trying to parallelise it, proved more difficult than expected. The experiments lead us to believe 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 necessary. We therefore conclude that a method in which we model longer motion sequences may be more fruitful.



# Motion Diffusion Models

In Chapter 3., we came to the conclusion that the next sensible step would be an investigation of a model that operates directly on motion sequences. We also desired to have a general model that can be used for many different tasks, such that we might get a better overview of the uses of motion modelling. We therefore decided, due to promising literature presented in Section 2.2, that diffusion models were worth investigating.

## 4.1. Method

### 4.1.1. Diffusion Models

The diffusion framework that we use is defined theoretically as follows [HJA20, ND21, TCL22]. Given  $x_0 \sim p(x_0)$  some data distribution, diffusion is a Markov noising process over latent variables  $x_t, t = 1, \dots, T$ , in which the process of adding noise (the 'forward noising process') is defined through the distributions

$$\begin{aligned} q(x_t|x_{t-1}) &:= \mathcal{N}(\sqrt{1-\beta_t}x_{t-1}, \beta_t\mathbf{I}) \\ q(x_T|x_0) &:= \prod_{t=1}^T q(x_t|x_{t-1}) \end{aligned} \tag{4.1}$$

where with a sensible  $\beta$  schedule **TODO: Describe the beta schedule better** and a large  $T$ , we approximate a standard normal distribution as we approach  $t = T$ . This entails that the first latent variable  $x_T$  is approximately gaussian noise. The 'reverse noising process' is non tractable without the full data distribution, which we don't have access to, hence we approximate it using some parametrized function

$$p_\theta(x_{t-1}|x_t) = \mathcal{N}(\mu_\theta(x_t, t), \Sigma_\theta(x_t, t)) \tag{4.2}$$

#### 4. Motion Diffusion Models

We find with  $p$  and  $q$  that this system forms a VAE that can be trained through the Variational Lower Bound (VLB) [KW22].

**TODO:** Put the classic figure of the noising/denoising process from [HJA20] in here

In practice however we note some modification to this definition to ease our implementation and to improve performance. Firstly we note that the forward noising process can be written as

$$q(x_t|x_0) \sim \mathcal{N}(\sqrt{\bar{\alpha}_t}x_0, (1 - \bar{\alpha}_t)\mathbf{I}) \quad (4.3)$$

with  $\bar{\alpha}_t = \prod_{s=1}^t \alpha_s$  and  $\alpha_t = 1 - \beta_t$ . This is simply a mixing of the data with gaussian noise according to a monotonically decreasing schedule  $\bar{\alpha}_t$  and allows us to sample any latent at any timestep directly from the clean signal, which will aid our training procedure. We note that we take the following definition for the  $\bar{\alpha}_t$  schedule, as in [ND21],

$$\begin{aligned} \bar{\alpha}_t &= \frac{f(t)}{f(0)} \\ f(t) &= \cos\left(\frac{t/T + s}{1+s} \cdot \frac{\pi}{2}\right) \end{aligned} \quad (4.4)$$

with the a small offset  $s$  (we take  $s = 0.008$  as in [ND21]) to avoid  $\beta_t$  from being too small near  $t = 0$  which can reduce performance according to the authors.

We can also present a more simple reverse noising process, where rather than predicting the mean and variance of the noise for a single step, we directly try to predict the denoised signal [RDN<sup>+</sup>22],

$$\hat{x}_\theta(x_t, t) \approx x_0 \quad (4.5)$$

which allows us to describe the denoising process as follows

$$p_\theta(x_{t-1}|x_t) = q(x_{t-1}|\hat{x}_\theta(x_t, t)) \quad (4.6)$$

**TODO:** check this definition of the denoising process against the code, I feel something is off  
This direct prediction of the denoising signal allows us to use a the simplified MSE objective [HJA20, RDN<sup>+</sup>22] during training

$$\mathcal{L}_{simple} = \mathbb{E}_{x,t} [\|x - \hat{x}_\theta(x_t, t)\|_2^2] \quad (4.7)$$

which can still be interpreted as a form of Variational Lower Bound without the variance terms.

**TODO:** missing some note about the fixed small variance from the code?

#### 4.1.2. Disney Motion Diffusion (DMD)

Armed with this understanding of diffusion, we define our Disney Motion Diffusion (DMD) model, strongly inspired by the work of Tevet. et. al [TRG<sup>+</sup>22].

#### State Representation

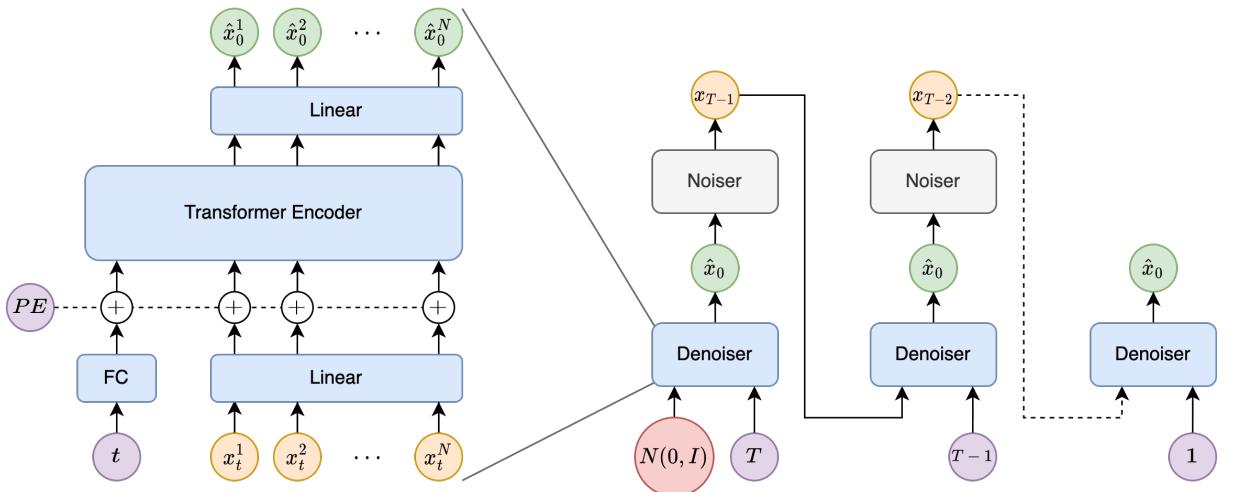
Our state represents a motion sequence of length  $N$ , and includes primarily a root translation,  $r \in \mathbb{R}^{N \times 3}$ , and 6d [ZBL<sup>+</sup>18] joint orientations in body frame,  $o \in \mathbb{R}^{N \times 6}$ . The state can be

extended with some optional extras depending on the data, of foot ground contacts  $b \in \mathbb{R}^{N \times 1}$ , and explicit velocities for the root translation and joint orientations. Again, depending on the data, we can represent the root translation, and root orientations (a subset of the joint orientations) either in world space, where each motion sequence will have a distinct starting point and starting orientation, or as what we call a 'stuck root' reference frame, where the first pose is set to the positions  $(0, 0)$  and identity orientation and all subsequent poses are described with respect to this first frame, with the view being that this formulation leads to a more general and smaller space that the model needs to understand.

## Architecture

We follow closely [TRG<sup>+</sup>22] in our architecture, though without the text conditioning. We use a simple transformer [VSP<sup>+</sup>17] encoder as the denoising network, conditioned on the diffusion timestep so that the network can learn how much noise is present that needs removing. This is used with the as described denoising process which can be seen in Figure 4.1.

**TODO:** Link to appendix with more details on the architecture choices, number of hidden layers, etc



**Figure 4.1.:** DMD Architecture & Denoising Process

## 4. Motion Diffusion Models

### Training

In addition to the simple MSE loss described in section 4.1.1, we can include a number of other losses to aid the training procedure.

$$\begin{aligned}\mathcal{L}_{fk} &= \frac{1}{N} \sum_{i=1}^N \|x_0^{(pos)} - FK(\hat{x}_0^{(i)})\|_2^2 \\ \mathcal{L}_{vel} &= \frac{1}{N-1} \sum_{i=1}^{N-1} \|(x_0^{(i+1)} - x_0^{(i)}) - (\hat{x}_0^{(i+1)} - \hat{x}_0^{(i)})\|_2^2 \\ \mathcal{L}_{contact} &= \frac{1}{N-1} \sum_{i=1}^{N-1} \left\| \left( FK(\hat{x}_0^{(i+1)}) - FK(\hat{x}_0^{(i)}) \right) \cdot \hat{b}^{(i)} \right\|_2^2\end{aligned}\quad (4.8)$$

with  $x_0^{(i)}$  representing the  $i$ th pose of the motion sequence  $x_0$ . The  $\mathcal{L}_{fk}$  loss is a forward kinematics loss, i.e that where the positions are calculated from the orientations, and compared to the ground truth positions that we have access to,  $x_0^{(pos)}$ , and which provides another mechanism by which the orientations might be adjusted, thus increasing the robustness of the predictions. The  $\mathcal{L}_{vel}$  loss is a calculated velocity loss, in which the first order finite differences are used to estimate the velocities of the state (orientations and root translation only in this case), and which encourages the model to learn a smooth motion sequence. The  $\mathcal{L}_{contact}$  loss is a foot contact loss, where the velocities of the ground contacting feet joints, indicated by the predicted foot contacts  $\hat{b}^{(i)}$ , are minimised so as to discourage foot sliding. We note that the  $\mathcal{L}_{contact}$  loss is only used when the ground contacts are included in the state representation.

### Inpainting

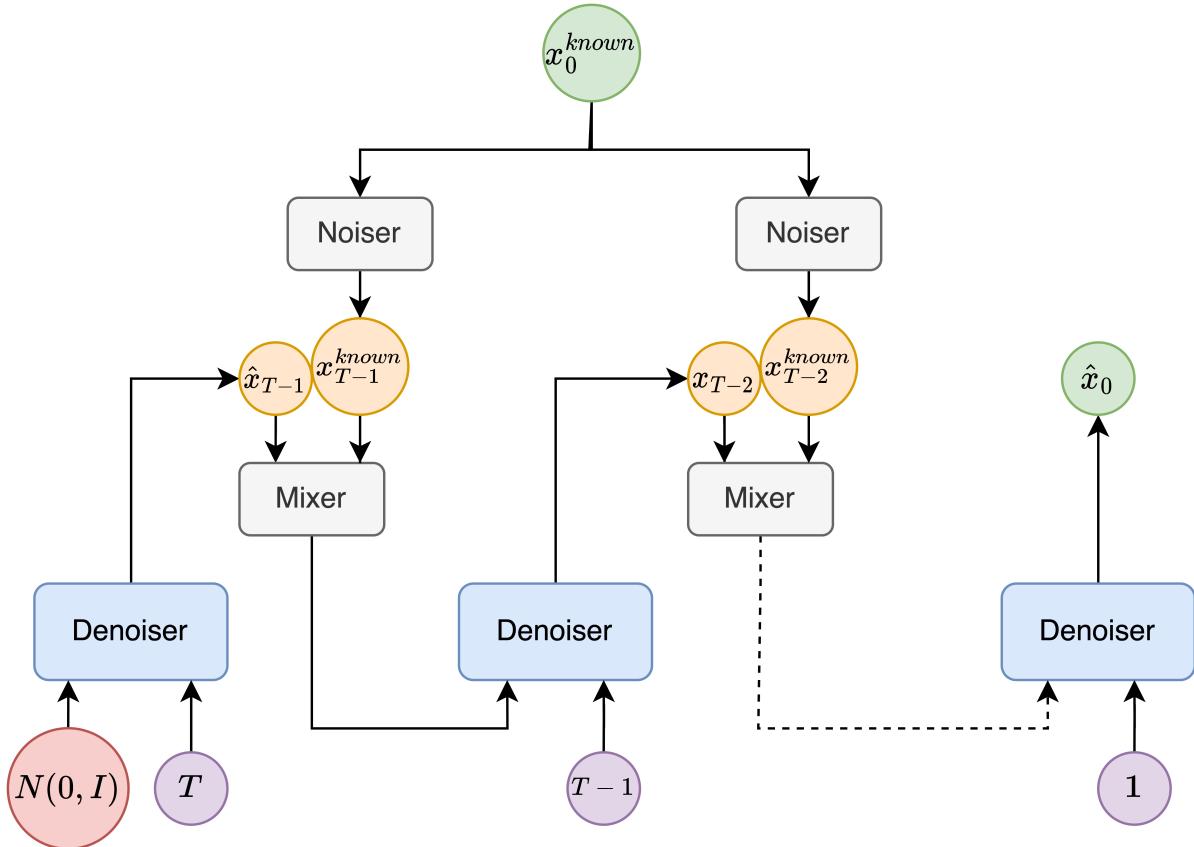
An important tool in the arsenal of the diffusion model literature is that of inpainting [LDR<sup>+</sup>22]. Inpainting is the process of keeping a portion of the state constant throughout the diffusion process. An example of the use of this technique would be that of changing the motion of the lower body while maintaining the motion of the upper body, though there are many other use cases that are explored in more detail in Section 4.2. The inpainting procedure works as follows; at each denoising step we mix a portion of the latent variable state  $\hat{x}_t$ , indicated through the binary mask  $m$ , with the complementary portion of some known state  $x^{(known)}$ , indicated through  $1 - m$ ,

$$\hat{x}_t = m \odot \hat{x}_t + (1 - m) \odot q(x^{(known)} | t - 1) \quad (4.9)$$

Graphically this process can be seen in Figure 4.2. This procedure is particularly attractive at it requires no special training procedure, and can be employed readily with any diffusion model.

### Data

At Disney Research|Studios we have access to professionally capture motion capture data that encompasses a wide range of possible motion for a number of subjects. In total we have TODO: X motion sequences, varying in length from TODO: X to TODO: X seconds long, thus in total

**Figure 4.2.: Inpainting Procedure**

we have **TODO: X** minutes of motion capture data. This data is normalised elementwise and retargeted to a standard skeleton. A number of motion sequences are held out for testing.

**TODO:** Describe dataset 1 (fixed root, not contacts, etc) and 2 (moving root, stuck root, contacts, how to get contacts, etc.)

## 4.2. Experiments

Our main goal in this investigation into diffusion models was to get a better understanding of their performance on a wide range of tasks. We keep in mind the original task of improving an existing motion sequence, but are aware of the other potentials of the model and so do not limit ourselves in scope.

### 4.2.1. Test Cases

We present here a variety of different test cases that on which the models performance can be evaluated. The general structure of the evaluation is to generate a sufficiently large number of sequences for each given task, then to import them into blender and visually inspect their quality and how well they achieve each task. We have decided to take a qualitative approach

## 4. Motion Diffusion Models

as we are attempting to get a better understanding of the models capabilities, and so this was the path of least resistance to obtain such an understanding in the time that was left after the HuMoR investigation. We note that in the future, if a given task is deemed worth investigation further, or if a significantly larger number of models are being tested, more rigorous evaluation metrics would be greatly beneficial.

### Sequence Generation

To ensure that the model has learned a sensible variety of human motion, we generate a number of sequences from gaussian noise, and check that they contain sufficient diversity, considering that a main indicator of an overfit diffusion model is that of a lack of diversity.

### Denoising

**TODO:** Come up with a better test for this

### Occluded Legs

An important situation in which the current Disney Research|Studios model does not perform optimally is that of occluded sequences. To test the ability of the diffusion models to handle this situation, we evaluate the models capabilities to generate the leg motion of an entire motion sequence. We achieve this using the inpainting framework described in Section 4.1.2, keeping the upper body joint rotations and the root translation fixed, and allowing the model free reign over the rest of the state.

### Missing State

To further test the ability of the models to handle occlusions, a test case where we only keep a random percent of the state during inpainting. This can either be

1. a random subset of each frame in a motion sequence
2. a random subset that is constant across the whole sequence

with the first testing the models ability to handle arbitrarily occurring missing data, and the second testing the models ability to recover from longer term occlusions.

### Inbetweening

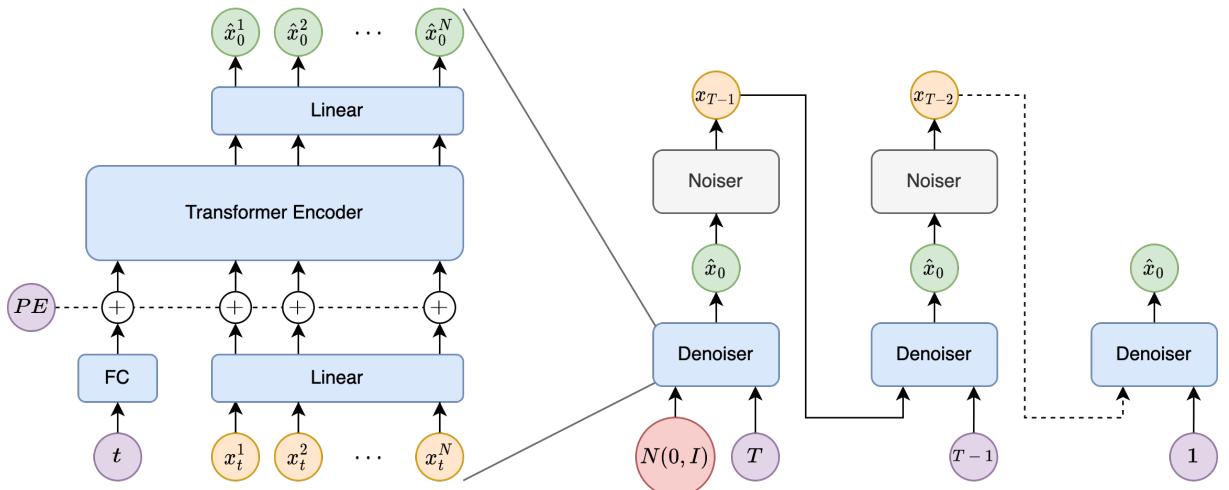
Another important area of motion generation research is that of Motion Inbetweening, in which the goal is to fill in a middle part of a motion sequence in which a number of frames at the beginning and end of the sequence are given. Again we can use the inpainting framework from Section 4.1.2 and keep the edges of the motion as static.

## 4.2.2. Baseline Model

The baseline model was trained with the Disney Research!Studios dataset that does not, and cannot (due to a lossy rendering of the mocap), contain foot contacts. The state includes the root translation, joint orientations, and their respective velocities explicitly. It was trained for TODO: X epochs. **TODO: More details?**

### Sequence Generation

We found that the model generates a diverse set of plausible motion, a number of frames are shown in Figure 4.3. A few shortcomings however can be seen. Most notably we find that the model produces a fair bit of foot sliding **TODO: Is this true??**, where the foot seem to be in contact with the ground but has a non zero velocity in some directions. This effect can be mitigated through the use of the foot sliding loss coupled with data with containing labeled foot contacts.



**Figure 4.3.: Motion Generation - Baseline model**

### Denoising

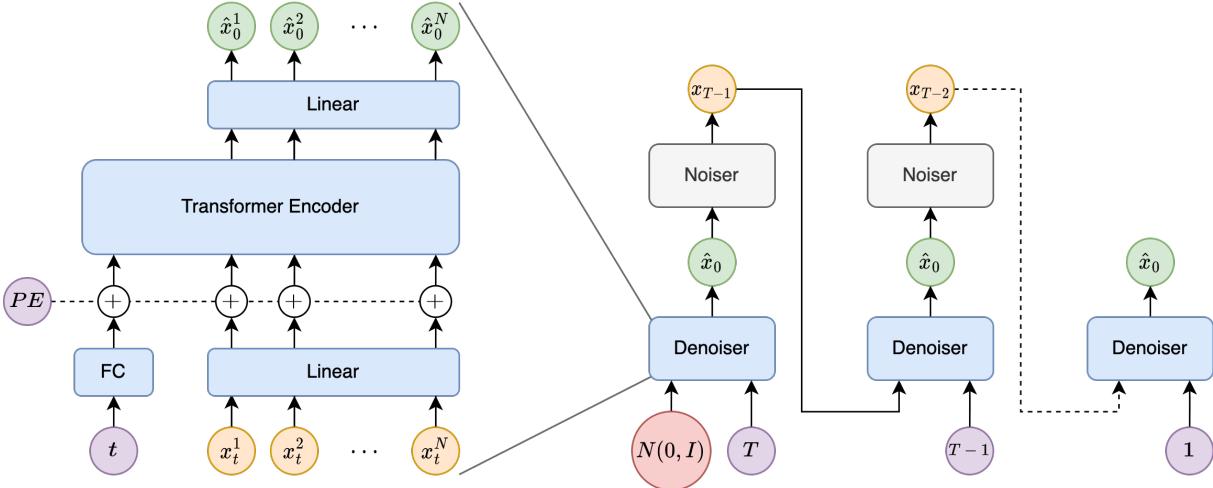
**TODO: Come up with a better test for this**

### Occluded Legs

The model shows a strong performance on the task of inpainting missing legs in a motion sequence, with the result being of leg motion that is often both smooth and well adhering to the upper body motion. A few examples frames are shown in Figure 4.5. The most notable failure case is that sometimes the model reverses the direction of motion. We postulate that this is due to the fact that the data for the baseline model has a root that does not move in the forward/reverse direction, therefore the pendulating rotation motion of the hips, without the

#### 4. Motion Diffusion Models

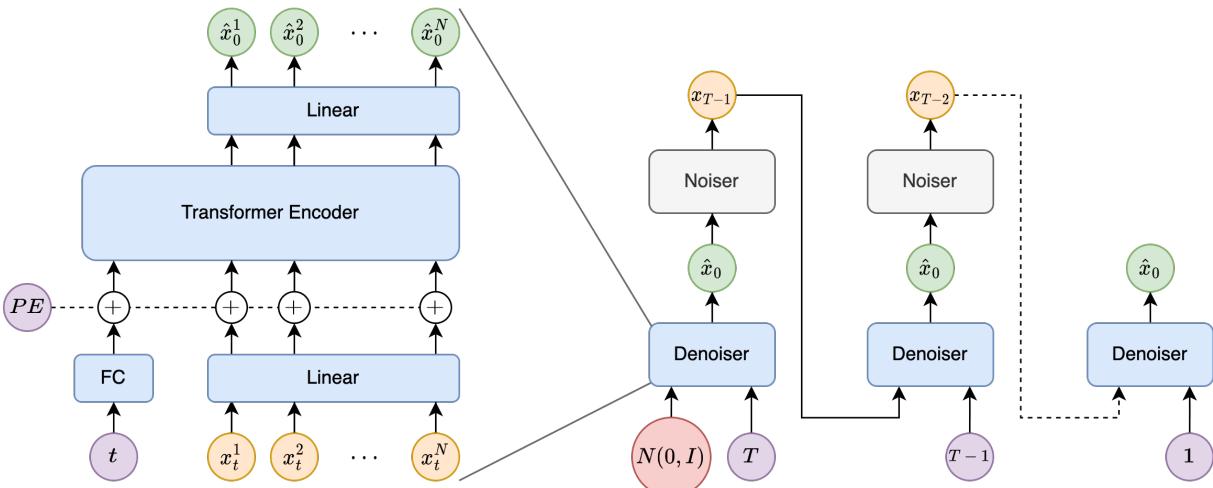
additional context of a specific direction, can often be both interpreted as both forward and backward motion.



**Figure 4.4.: Occluded Legs Inpainting - Baseline model**

#### Missing State

**Random each frame** When removing a new random subset of the joints of each pose every frame in the motion sequence, we find that the model can very accurately reconstruct the missing state, even when only 50% of the state is given each frame, and surprisingly it still performs reasonably well when only 25% of the state is given each frame. A few example frames are shown in Figure ???. This shows that the model has learned well the concept of temporal consistency, as in this experiment it is likely that a missing joint will be surrounded (temporally) by given joints, hence it simply needs to interpolate between the given joints.



**Figure 4.5.: Occluded Legs Inpainting - Baseline model**

**Random for the whole sequence** **TODO:** Describe better

**Inbetweening****4.2.3. Model with Foot Contacts****Sequence Generation****Denoising****Occluded Legs****Missing State****Inbetweening****4.2.4. Autocompletion**

Another project that is running at Disney Research|Studios is that of pose autocompletion. This is the task of taking a set of *handles*, a subset of joints, and predicting the position of the rest of the joints. The goal here is to again help to improve the animation pipeline, by reducing the amount of work an animator has to do to get a sensible pose. As a side note to the previous experiments, we trained some diffusion models for the described task, where we enforce a sequence length of 1 so that we are only predicting single pose, and where we modify the inpainting to allow for a specific set of joints to be deemed *handles* and to be kept fixed.

**Baseline**

The model is as described in Section 4.2.2 however with a sequence length of 1.

We find that while the model does present plausible poses, and that on average these poses have handle joints close to the desired positions of the handles (with a few exceptions), it does not satisfy the task properly as can be seen in **TODO: include images of position of handles vs. position of actual joints**. An ideal solution would not move the handles at all, and would simply modify the rest of the joints.

**TODO: include images**

**Handle Conditioning**

To mitigate this issue of the handles being modified, we propose a handles conditioning, in which a one hot embedding of the handles is run through a linear layer, then concatenated to the input of the network. A loss is then applied to the output that penalises any deviation in position of the handles that are output as compared to those input. In theory this should force the network to learn to consider the handles as immutable, and to create a pose that satisfies the handles.

#### 4. Motion Diffusion Models

Though the implementation was largely completed, we unfortunately did not have time to finish debugging some of the finer details, and so cannot provide concrete results. However we deemed the idea interesting enough to present in this thesis, and hope that it can be of use in future research directions.

### 4.3. Conclusion

The DMD diffusion framework described in Section 4.1.2 proves to represent an exciting research direction. The baseline model shows good performance on a variety of tasks, as seen in Section 4.2.2, demonstrating that the diffusion framework is able to learn a general purpose motion prior model without task specific fine tuning or architecture design. This gives us hope for future research directions in which the specific task to solve is more precisely described, where more fine grained architecture, data and training choices could be made that could further improve the performance of the models.

The implementation of the diffusion framework within the Disney Research|Studios codebase paves the way for further investigation into its uses in the context of general animation tasks.

**TODO:** Longer conclusion?

### 4.4. Future Work

#### 4.4.1. Model

Some of the shortcomings of the model, seen in some of the Inpainting tasks and most notably in the Inbetweening task, is that of inconsistency between the specified known motion and the motion subsequently generated during inpainting by the model. For inbetweening, if we avoid overwriting the generated motion with the known motion in the last denoising step, the motion is very smooth **TODO: image**, however does not quite match the known motion, as when we override the generated motion with the known motion, we find that there is a discontinuity between the generated and known motion, i.e the motion jumps. This effect might be mitigated with a number of strategies. Firstly we might condition the model with an indication of where a known area is specified and penalise it during training for deviating from this known motion, as described in Section 4.2.4. Another method would simply be that of a postprocessing step, in which the known motion and the generated motion is mixed for a number of frames, thus achieving a smooth transition between the two.

The model training could also be improved. The stopping criteria is currently taken as the point where the loss visually seems to have converged on the loss graphs. This however is not a robust method as it does not take into account the possibility of overfitting or the possibility of decreasing performance. This can be mitigated through the use of validation metrics as described in Section 4.4.3, or by a method such as that presented in [TRG<sup>+</sup>22], in which they retrospectively choose the checkpoint that minimizes the FID score.

#### 4.4.2. Model use

- Inpainting - We notice that the target changes often, even in the last steps - Maybe change the schedule to repeat the last steps a number of times to allow more time for the process to converge - Autocomplete

#### 4.4.3. Model evaluation

- We presented an entirely qualitative evaluation of the models - We should make more effort to report on the metrics - Metrics - Generation - FID score? (the one comparing the distribution to the a realistic dist of data) - Diversity - Checking it's not just regurgitating the training data
- Inbetweening - Smoothness around the borders - Denoising/missing joints - Difference to the ground truth - [TCL22] - Physical foot contact score - Model comparison - We should compare the models to some baseline models - For example the missing state evaluation where a new subset of the joints is removed each frame might well be accurately reconstructed by simple linear interpolation between the missing frames, so the models performance might not be so impressive afterall.

#### 4.4.4. Data

**TODO: Cite some paper that says data is more important than anything** We note that our dataset, with **TODO: X** minutes of motion spanning **TODO: X** subjects is significantly smaller than the AMASS [MGT<sup>+</sup>19] dataset that contains more than 40hours of motion data spanning 300 subjects. We therefore expect that our model will generalise less well for than if it were trained on the AMASS dataset. We also note however that this is not possible due to the licensing restrictions on the AMASS dataset limiting it to non-commercial use, and so if any of the models we explored in this thesis or in subsequent follow up work are to be productionized, it would be prudent to obtain a larger dataset of motion, especially considering that the use cases in which such a system would be most helpful to an animator might well be those that are furthest from the data distribution we currently have, as the more obscure and intricate the motion sequence the harder it would be to animate.



# **Conclusion and Outlook**

TODO: Chpt 1



# **Appendix**

Appendix intro

## **A.1. Section**

**TODO:**



# Bibliography

- [BKG21] Dor Bank, Noam Koenigstein, and Raja Giryes. Autoencoders, 2021.
- [BKL<sup>+</sup>16] Federica Bogo, Angjoo Kanazawa, Christoph Lassner, Peter Gehler, Javier Romero, and Michael J. Black. Keep it smpl: Automatic estimation of 3d human pose and shape from a single image, 2016.
- [CHS<sup>+</sup>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.
- [CKP<sup>+</sup>21] Kyungmin Cho, Chaelin Kim, Jungjin Park, Joonkyu Park, and Junyong Noh. Motion recommendation for online character control. *ACM Trans. Graph.*, 40(6), dec 2021.
- [Cla] Simon Clavet. Motion matching and the road to next-gen animation.
- [CSY<sup>+</sup>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<sup>+</sup>23] Yuming Du, Robin Kips, Albert Pumarola, Sebastian Starke, Ali Thabet, and Artsiom Sanakeyeu. Avatars grow legs: Generating smooth human motion from sparse tracking inputs with diffusion model, 2023.
- [DN21] Prafulla Dhariwal and Alex Nichol. Diffusion models beat gans on image synthesis. *CoRR*, abs/2105.05233, 2021.
- [HJA20] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *CoRR*, abs/2006.11239, 2020.
- [HKPP20] Daniel Holden, Oussama Kanoun, Maksym Perepichka, and Tiberiu Popa. Learned motion matching. *ACM Trans. Graph.*, 39(4), aug 2020.

## Bibliography

- [HSK16a] Daniel Holden, Jun Saito, and Taku Komura. A deep learning framework for character motion synthesis and editing. *ACM Trans. Graph.*, 35(4), jul 2016.
- [HSK16b] 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.
- [HYNP20] Félix G. Harvey, Mike Yurick, Derek Nowrouzezahrai, and Christopher Pal. Robust motion in-betweening. *ACM Transactions on Graphics*, 39(4), aug 2020.
- [KW22] Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2022.
- [LDR<sup>+</sup>22] Andreas Lugmayr, Martin Danelljan, Andres Romero, Fisher Yu, Radu Timofte, and Luc Van Gool. Repaint: Inpainting using denoising diffusion probabilistic models, 2022.
- [LGK20] Zhengyi Luo, S. Alireza Golestaneh, and Kris M. Kitani. 3d human motion estimation via motion compression and refinement. *CoRR*, abs/2008.03789, 2020.
- [LVC<sup>+</sup>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.
- [MGT<sup>+</sup>19] Naureen Mahmood, Nima Ghorbani, Nikolaus F. Troje, Gerard Pons-Moll, and Michael J. Black. Amass: Archive of motion capture as surface shapes, 2019.
- [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.
- [ND21] Alex Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models, 2021.
- [OBB20] Ahmed A. A. Osman, Timo Bolkart, and Michael J. Black. STAR: sparse trained articulated human body regressor. *CoRR*, abs/2008.08535, 2020.
- [PCG<sup>+</sup>19] Georgios Pavlakos, Vasileios Choutas, Nima Ghorbani, Timo Bolkart, Ahmed A. A. Osman, Dimitrios Tzionas, and Michael J. Black. Expressive body capture: 3d hands, face, and body from a single image. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.
- [RBH<sup>+</sup>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.
- [RBL<sup>+</sup>21] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models, 2021.

- [RDN<sup>+</sup>22] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents, 2022.
- [SLY15] Kihyuk Sohn, Honglak Lee, and Xinchen Yan. Learning structured output representation using deep conditional generative models. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015.
- [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.
- [TRG<sup>+</sup>22] Guy Tevet, Sigal Raab, Brian Gordon, Yonatan Shafir, Daniel Cohen-Or, and Amit H. Bermano. Human motion diffusion model, 2022.
- [TWH<sup>+</sup>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.
- [VSP<sup>+</sup>17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017.
- [WH97] D. J. Wiley and J. K. Hahn. Interpolation synthesis of articulated figure motion. *IEEE Computer Graphics and Applications*, 17(06):39–45, nov 1997.
- [YSI<sup>+</sup>22] Ye Yuan, Jiaming Song, Umar Iqbal, Arash Vahdat, and Jan Kautz. Physdiff: Physics-guided human motion diffusion model, 2022.
- [YZS<sup>+</sup>23] Ling Yang, Zhilong Zhang, Yang Song, Shenda Hong, Runsheng Xu, Yue Zhao, Wentao Zhang, Bin Cui, and Ming-Hsuan Yang. Diffusion models: A comprehensive survey of methods and applications, 2023.
- [ZBL<sup>+</sup>18] Yi Zhou, Connelly Barnes, Jingwan Lu, Jimei Yang, and Hao Li. On the continuity of rotation representations in neural networks, 2018.