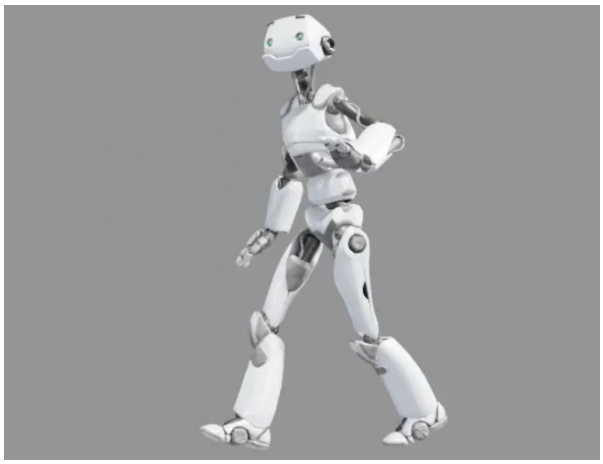
Motion Priors for Pose Estimation and Animation Workflows



Luke Smith

Master Thesis
May 2023

Prof. Dr. Robert W. Sumner
Dr. Jakob Buhmann

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Disney RESEARCH STUDIOS

cgl
computer graphics laboratory

Abstract

Motion capture, the process of automatically recording the motion of a person or object, has firmly established itself as a key technology in modern animation pipelines since 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 represent a high barrier not only for individual artists but also for small teams. One way to overcome this barrier is with a system that can capture motion directly from RGB video. Such systems commonly perform pose estimation on the individual video frames and run the result through an optimizer to clean up the motion. While promising, such systems occasionally struggle with certain artifacts and with some occluded motions. The goal of this thesis is therefore to consider how we might train a general human motion model that could improve upon these shortcomings. To begin with, we take a state-of-the-art approach based on Variational Auto Encoders and investigate how to improve upon its issues to make it usable for our purposes. Next, we consider a relatively new class of models, diffusion models, which are proving to be powerful tools in many domains. We investigate the application of such models in the context of motion modeling and implement and evaluate a baseline model in order to conclude the feasibility of such models for our purposes.

Zusammenfassung

Motion Capture, die automatische Aufzeichnung der Bewegung einer Person oder eines Objekts, hat sich als Schlüsseltechnologie in modernen Animationspipelines etabliert, da sie qualitativ hochwertige, physikalisch plausible Ergebnisse in Echtzeit liefert. Allerdings stellen die Vorabkosten für maßgeschneiderte Software und Hardware, umständliche Motion-Capture-Anzüge und andere Probleme nicht nur für einzelne Künstler, sondern auch für kleine Teams eine hohe Hürde dar. Eine Möglichkeit, diese Hürde zu überwinden, ist ein System, das Bewegungen direkt aus RGB-Videos erfassen kann. Solche Systeme führen in der Regel eine Posenschätzung für die einzelnen Videobilder durch und lassen das Ergebnis durch einen Optimierer laufen, um die Bewegung zu bereinigen. Solche Systeme sind zwar vielversprechend, haben aber gelegentlich mit bestimmten Artefakten und verdeckten Bewegungen zu kämpfen. Ziel dieser Arbeit ist es daher, zu untersuchen, wie wir ein allgemeines menschliches Bewegungsmodell trainieren können, das diese Unzulänglichkeiten beheben kann. Zunächst nehmen wir einen modernen Ansatz, der auf Variational Auto Encoders basiert, und untersuchen, wie wir seine Probleme verbessern können, um ihn für unsere Zwecke nutzbar zu machen. Als Nächstes befassen wir uns mit einer relativ neuen Klasse von Modellen, den Diffusionsmodellen, die sich in vielen Bereichen als leistungsfähige Werkzeuge erweisen. Wir untersuchen die Anwendung solcher Modelle im Zusammenhang mit der Bewegungsmodellierung und implementieren und bewerten ein Basismodell, um die Durchführbarkeit solcher Modelle für unsere Zwecke zu prüfen.

Contents

List of Figures

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 its inherent costs, notably

- upfront costs for tailored software and hardware
- cumbersome motion capture suits
- lengthy setup times and 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/animators. 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?** We consider an approach that aims to capture motion directly from RGB video. The system

currently follows the following steps:

1. record a video of a human motion sequence
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 shows great promise. However, it has several drawbacks, the most notable being that the per-frame pose estimation system does not produce temporally consistent results. The predictions are made independently per frame, hence the resultant 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 achieved, it does not fix all issues. The optimiser struggles to handle occluded motion sequences, does not always deal with limb flips (where the left/right leg/arm predictions are horizontally flipped between frames), 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 training 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 explores such motion models with the primary goal of establishing the most promising model architecture and the most effective manner in which such a model might be used. Initially, we take 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. As this does not prove as fruitful as hoped we next try an emerging class of powerful models called diffusion models in Chapter 4. We explore the use of these models in the context of motion modeling in Section 4.1.2 and finally evaluate them on a variety of tasks with a view to better understanding their capabilities in Section 4.2. Having demonstrated their potential, we then look ahead to indicate avenues in which this promise might be realised in Section 4.3 and Section 4.4, before concluding in Chapter 5.

Related Work

The study of synthesizing 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 is performed between similar motions from a database of captured motion. This, however, has a high memory footprint, though efforts have been made to introduce learned aspects to compress the data into faster-to-query networks, such as [HKPP20].

More recent branches of motion modeling 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 matching, 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+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.

Holden et al. [HSKJ15] [HSK16b] present simple CNN-based autoencoder architectures that operate 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+21]. [CSY+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. Starke et. al present DeepPhase [SMK22], an autoencoder with a latent space enforced to match sinusoidal functions that represent periodic motion. Contrary to a common trope in sequence-level models, the authors of [TWH+22] and of [LZCvdP21] operate in a frame-to-frame regime, predicting the 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+21, TWH+22, LZCvdP21, MWFD21].

As we can see, the literature is rich and diverse, but we found ourselves drawn to the HuMoR model [RBH+21], due to its state-of-the-art performance and its 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+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+21], diffusion models have begun to spread into many other fields within machine learning [YZS+23], including the field of human motion modeling.

The authors of Avatars grow legs [DKP+23] denoise a sequence of SMPL [BKL+16] parameters conditioned 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 generated 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+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 the 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+22] describe a text conditioned denoising architecture.

As we have seen, diffusion models can be employed in the field of motion modeling for a wide variety of tasks. Motion can be generated [TRG+22, TCL22, DKP+23] conditioned on various inputs. Motion sequences can also be edited through inpainting [LDR+22, TRG+22] to change only parts of a sequence, and motion inbetweening can also be achieved similarly [TRG+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+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 ϵ_{data} that can be tailored to the modality as the HuMoR state is information-rich, containing 3D joints (hence can fit 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+21], alongside its use in a problem that directly matches our own (TestOps operating on RGB video) leads us to evaluate and investigate the model in Section 3.2, and subsequently try to extend and improve upon its limitations in Section 3.3.

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 omitted 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

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, all their respective velocities, and additional joint angles. This rich and redundant state representation is a superset of the SMPL models' state [BKL+16] (given some shape parameters β required by SMPL).

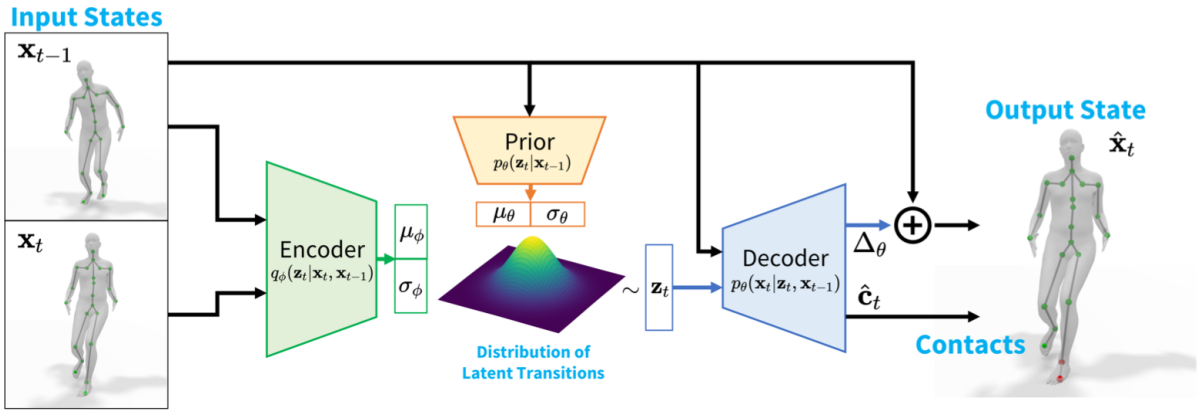


Figure 3.1: *HuMoR C-VAE Architecture [RBH+21]*

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 obtained 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+19] to obtain 2d pose estimates. Next, a three-stage optimisation procedure is proposed. The first two stages are used to obtain 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) are 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+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 3.7.

We refer the reader to the original paper [RBH+21] for details on some extra parameters that are also optimised during this procedure, notably the ground plane and SMPL shape parameters, and for details on a loss on the initial state $\mathbf{x}_0$ using a learned Gaussian mixture model that is applied during Stage 3.

3.2 HuMoR Investigation

3.2.1 Method

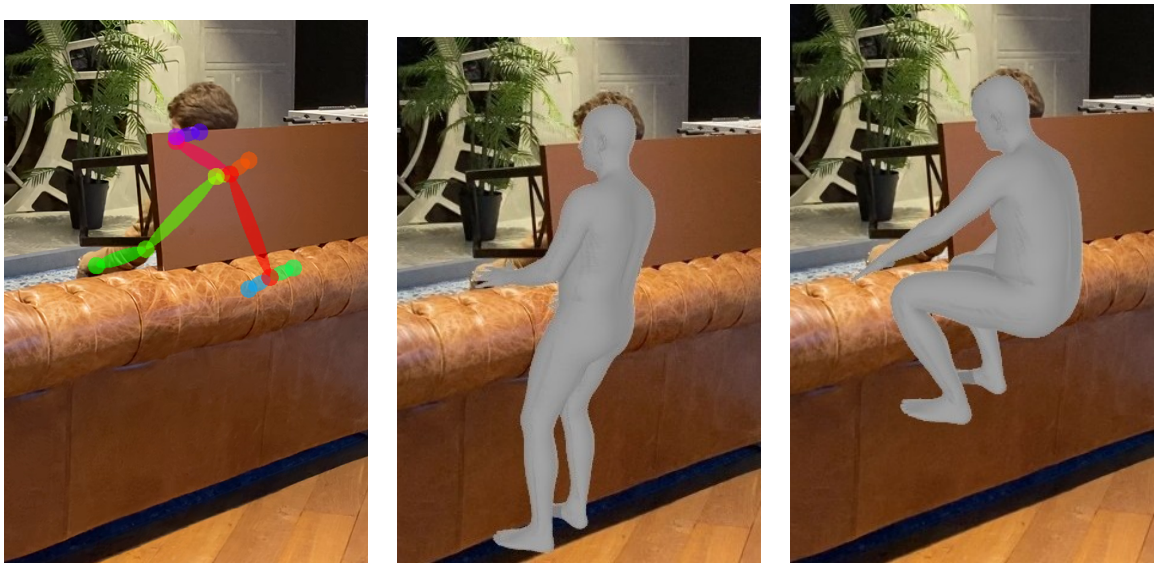
Our investigation began with a largely qualitative evaluation of the HuMoR model which had two main aims. First was to stress test the system, to see where it failed, where it succeeded, and if the mentioned benefits in the HuMoR paper [RBH+21] were as described. The second was to evaluate the model with the defects of the current automated mocap system (described in Chapter 1) in mind to see if it could complement its 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 containing a variety of motions; fast, slow, abnormal, and with occlusions. The HuMoR system was run 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 third makes use of the HuMoR motion model, we can therefore compare the Stage 2 results to the Stage 3 results to see where the model provided 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.



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

Figure 3.2: *HuMoR achieving occluded sitting*

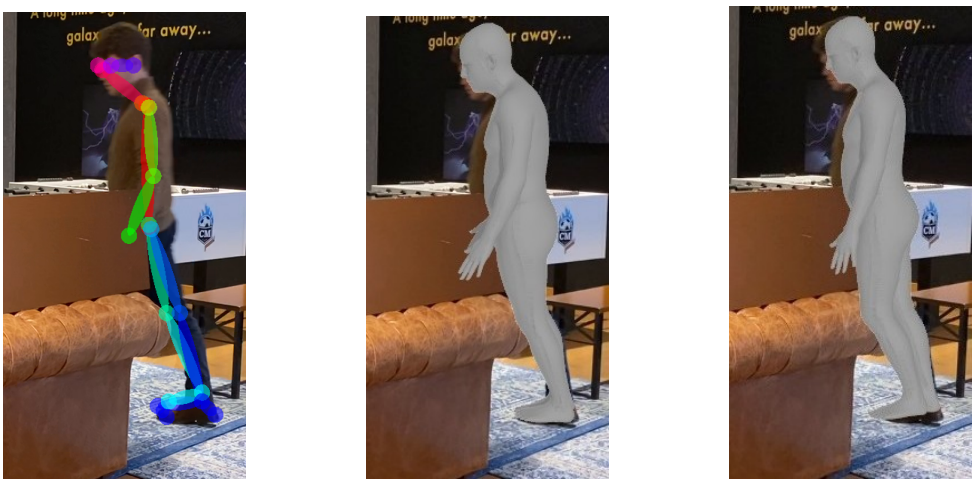
We also note that in many of the less complicated videos HuMoR produces clean motion and

deals with many 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

Occluded walking

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 [CHS+19] 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 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.



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

Figure 3.3: *Occluded walking failure point, 2 legs predicted instead of 1*

Axis-angle representation

We also found that the choice of axis angle representation for various rotations led to the emergence of commonly known issues that arise from the discontinuities present in the representation [ZBL+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.

Over-smoothed motion

Next, we saw overly smoothed motion for certain clips, most notably for particularly stylised motions such as dancing. We also found that if there is a single frame without OpenPose

(a) Raw frames

(b) OpenPose

(c) HuMoR, note the full rotation to the left

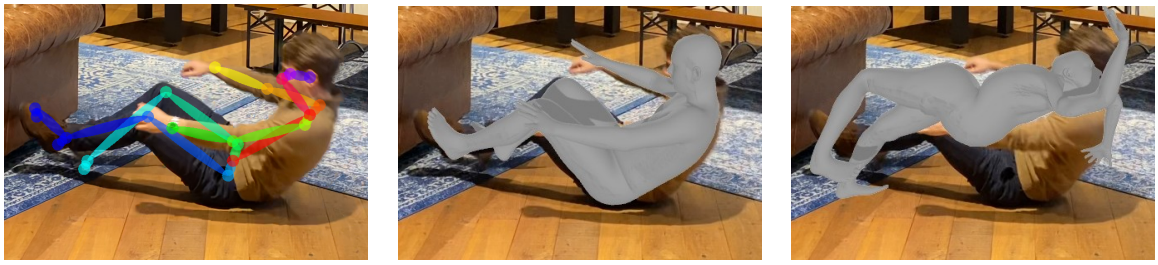
Figure 3.4: *Axis angle issue: dubious rotations*

predictions then from that frame onwards the TestOps fails.

Autoregressive nature

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.

Another issue due to the autoregressive nature was the computational burden and high memory usage. The system splits a full video into multiple overlapping subsequences as it is not com-



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

Figure 3.5: *HuMoR in a tangle*

putationally practical to roll out for the entire sequence. The overlapping sections then have an additional loss applied to them to ensure consistency between the subsequences. While reasonably effective, this demonstrates the heavy computation burden of a long-range autoregression and we find that the shorter the subsequences, the more discontinuities in the optimised motion.

Finally, and most importantly, we found that the TestOps was extremely slow. It took around 20mins per 2s clip, and we could batch at most four 2s clips together, totaling 20mins per 8s on an Nvidia GeoForce 1080ti with 11Gbs memory.

3.2.4 Profiling

To further 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 backward 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 backward step on the enormous computation graph resulting from the rollout.

3.2.5 Investigation Conclusion

Through this investigation, we found that HuMoR demonstrates a number of positive properties. The model handles occluded sitting motions well and often creates clean and plausible motion. It does however also exhibit numerous issues. The choice of axis angle representation, the dependence on OpenPose, and the smoothed motion all had a good chance of being fixed with different optimiser design choices. The glaring issue however was the unresonable amount of time the system took, rendering it unusable for our desired goal of having a close to real-time motion capture system. The next step was therefore to speed up the method, and to do so we needed to focus primarily on improving/removing the rollout of the third stage of the optimiser. In the next section, an attempt to do just this is presented.

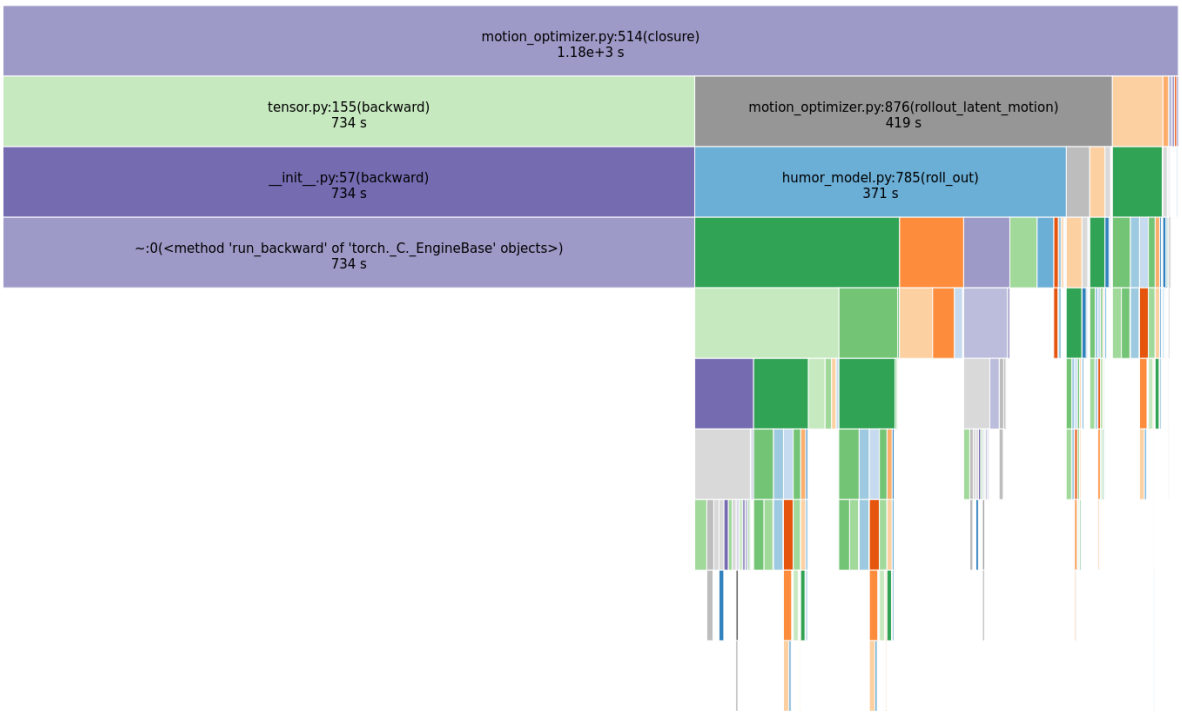


Figure 3.6: *TestOps profiling*

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 time.

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 suggested that there is no need for such a large context window, 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 3.7. 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 could take into account the decoded states, $\mathbf{x}'_{1:T}$, in several manners (using optimisation

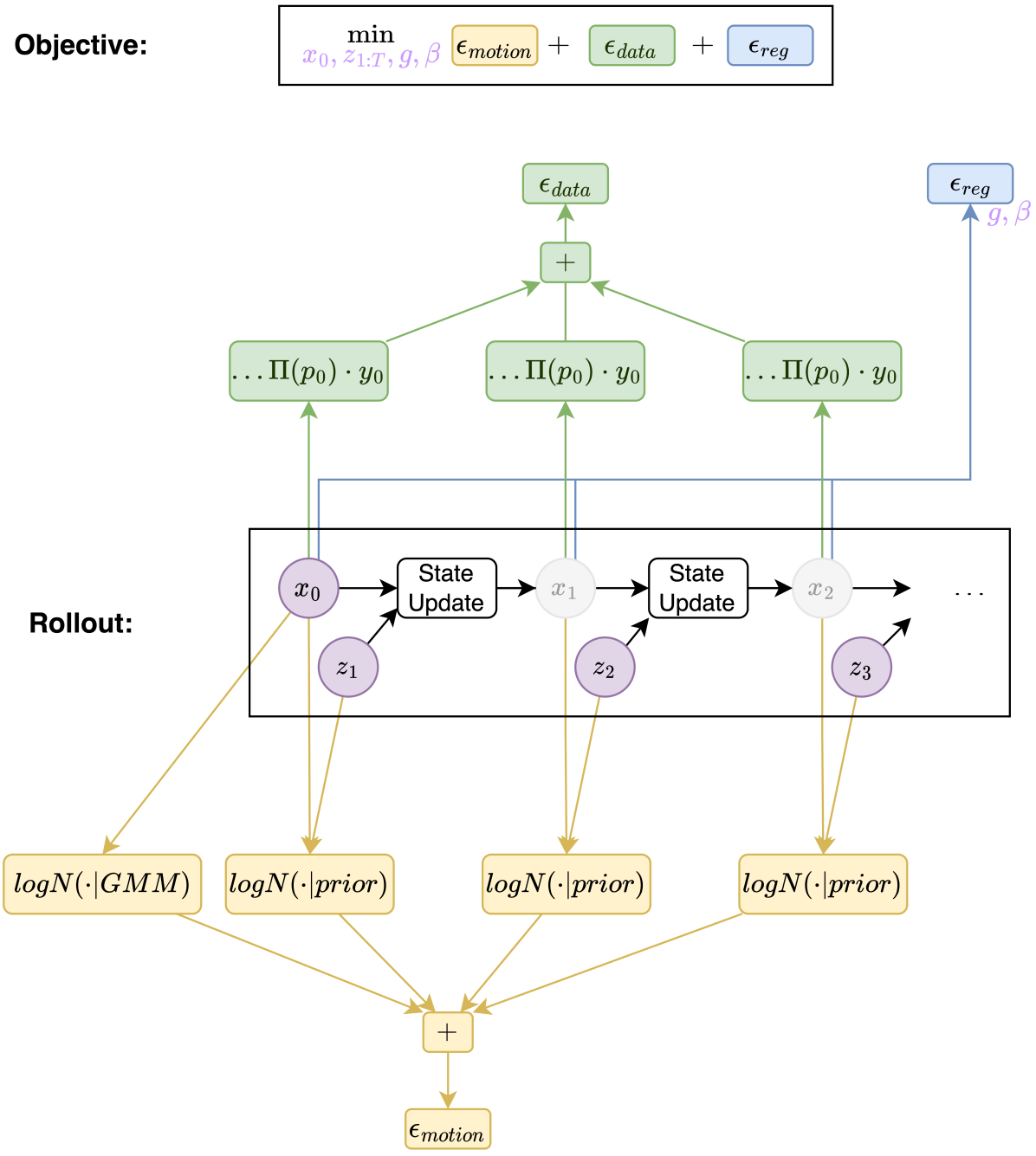


Figure 3.7: *TestOps Stage 3 Computation Graph*

The optimised variables are highlighted in purple, $\mathbf{x}$ being the state (root, joint positions etc.), and $\mathbf{z}$ the latent transitions. The gray $\mathbf{x}$'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. ϵ_{motion} is the loss due to the HuMoR prior (and initial state likelihood GMM briefly mentioned in Section 3.1.2). ϵ_{data} is the projection loss comparing the projected state to the OpenPose predictions. ϵ_{reg} are the regularisation losses, smoothness, foot contacts, etc (and are also the losses responsible for the ground plane g and the shape parameters β 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.

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

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, $\mathbf{x}_{1:T}$, 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 $\mathbf{x}$'s are regressed from the initial $\mathbf{x}_0$ 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 backward from any future frame $\mathbf{x}_t$, 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 $\mathbf{x'}$'s to get $\mathbf{x''}$, 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 3.9, where each next line is a newly decoded sequence derived from the previous.

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

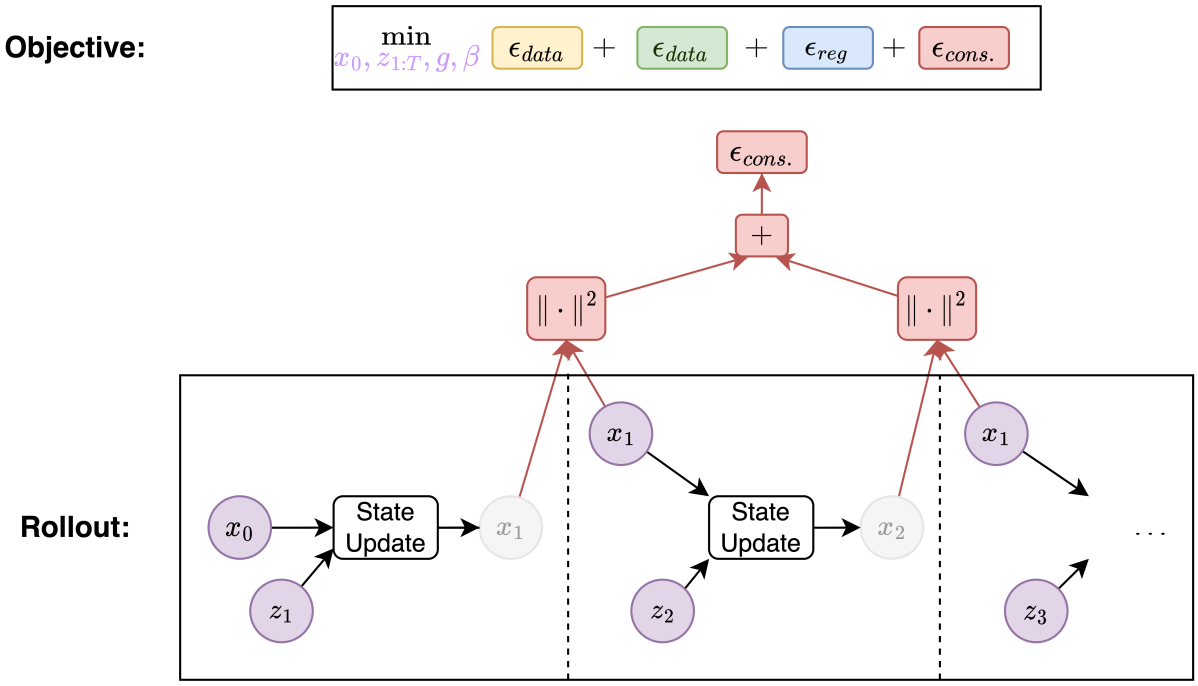


Figure 3.8: *Decoupled Computation Graph*

The computation graph is similar to that presented in Figure 3.7. This time however we roll out one step, maintain a persistent sequence of states, $\mathbf{x}_{1:T}$, and add a loss term, $\epsilon_{consistency}$ on the difference between the persistent $\mathbf{x}_{1:T}$ and decoded $\mathbf{x}'_{1:T}$. This theoretically allows for parallelised decoding of the latent variables, and thus a much faster computation time, by avoiding the rollout and large computation graph.

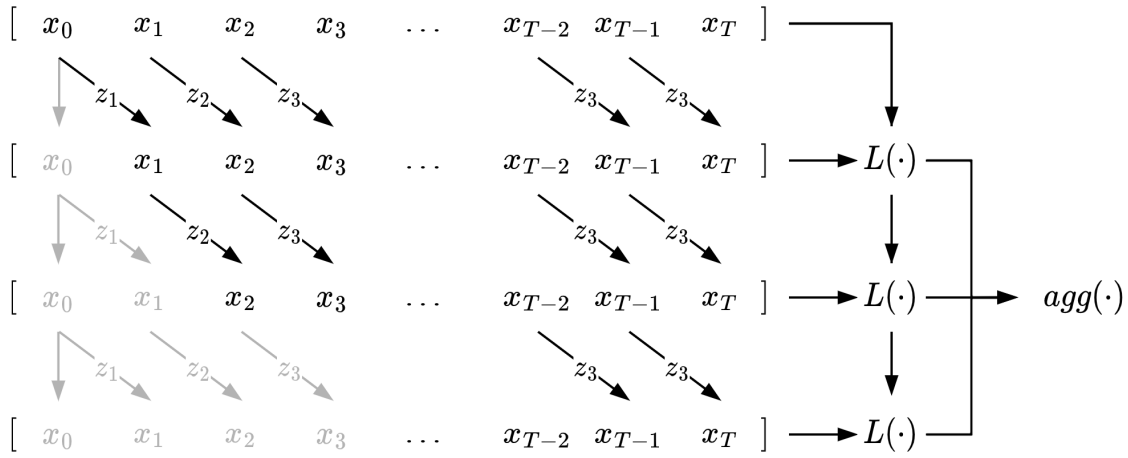


Figure 3.9: *Decoded sequences*

The formulation of the optimisation problem shown in Figure 3.8 can be extended by further 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.

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+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 [TC], 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

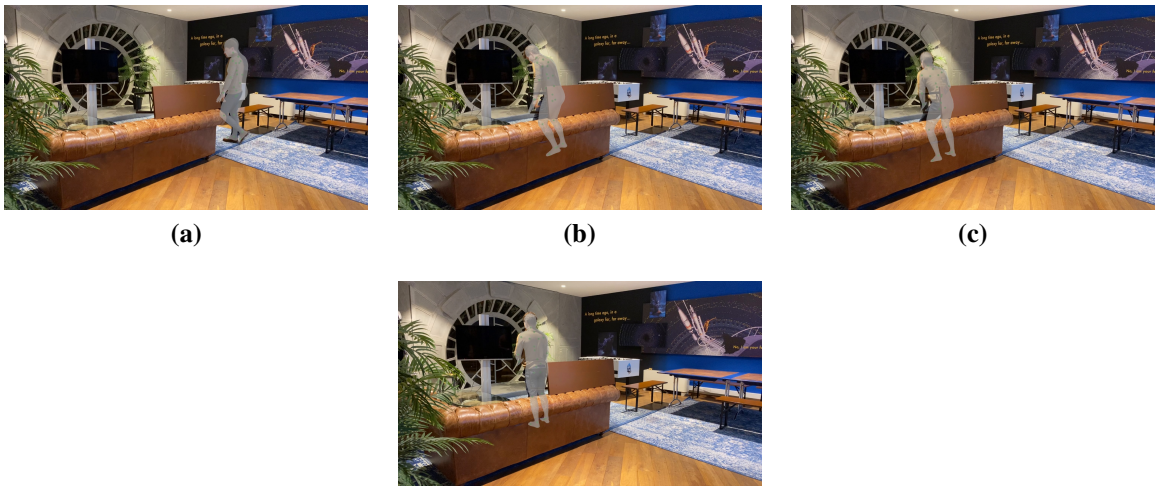
Initial $\mathbf{z}_{1:T}$

To begin with, we looked into the initial $\mathbf{z}_{1:T}$ that was obtained from projecting the Stage 2 $\mathbf{x}_{0:T}$ through the HuMoR encoder. We found that in occluded situations this $\mathbf{z}_{1:T}$ already encodes a sitting motion when locally rolled out, as seen in Figure 3.10, thus $\mathbf{z}_{1:T}$ represents a meaningful starting point for the Stage 3 optimisation. We note however that when we perform a longer rollout the resultant motion sequence, shown in Figure 3.12, deviates largely from the initial sequence $\mathbf{x}_{0:T}$ seen in Figure 3.11. This shows us that though the initial $\mathbf{z}_{1:T}$ encodes some sensible motion locally and though they 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.

With this understanding of the starting point of our optimisation variables in mind, we moved on to attempting to optimise them. We note that the next experiments were performed with varying numbers of decoding steps, 1, 2, 5, and 10 (as described in Figure 3.9), and that we found that longer rollouts produced more stable results but at the cost of longer compute times, as intuition would suggest.

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

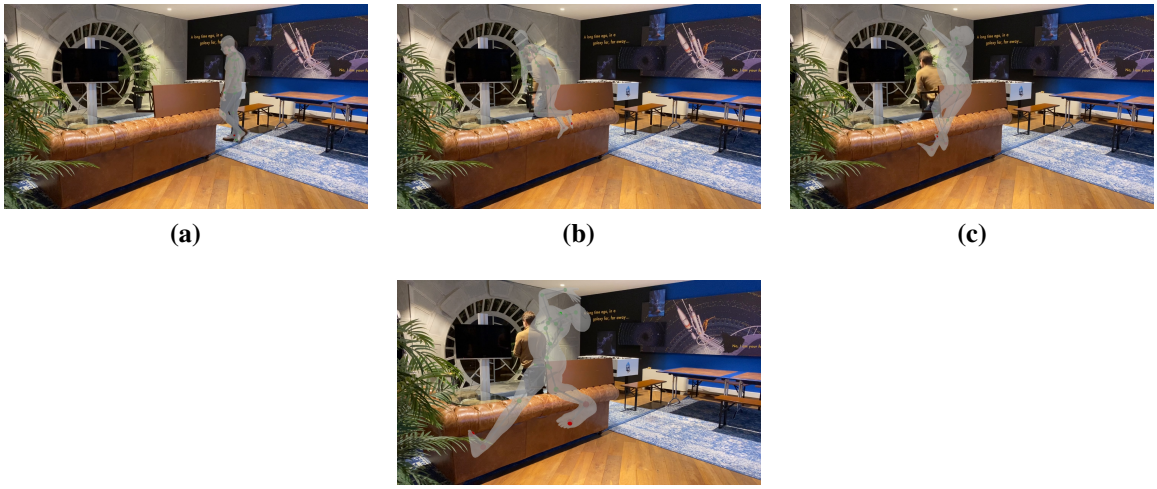
Here we can see that the legs bend in a sitting/squatting motion. The arms raising are due to an accumulation of errors.



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

(d)

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



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

(d)

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

Jointly optimising $\mathbf{x}_{0:T}$ and $\mathbf{z}_{1:T}$

Next, we embarked upon 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 (the effect shown in Figure 3.12) and the system failed to recover as it no longer had access to the long-range gradients present in the HuMoR TestOps method rollout. We also found that 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 best method.

We then 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 $\mathbf{x}_{0:T}$ 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 $\mathbf{x}_{0:T}$.

These experiments allowed us to choose a blending strategy for the decoded and persistent $\mathbf{x}_{0:T}$, and indicated that jointly optimising $\mathbf{x}_{0:T}$ and $\mathbf{z}_{1:T}$ was a difficult task due to divergent goals and difficult balancing, thereby suggesting we try optimising them separately.

Separately optimising $\mathbf{x}_{0:T}$ and $\mathbf{z}_{1:T}$

We next embarked upon some experiments in which the goal was either to optimise $\mathbf{x}_{0:T}$ whilst keeping $\mathbf{z}_{1:T}$ fixed or vice versa, so as to avoid the divergent goals found in Section 3.3.3 and to see if information could be passed between latent transitions and poses sensibly.

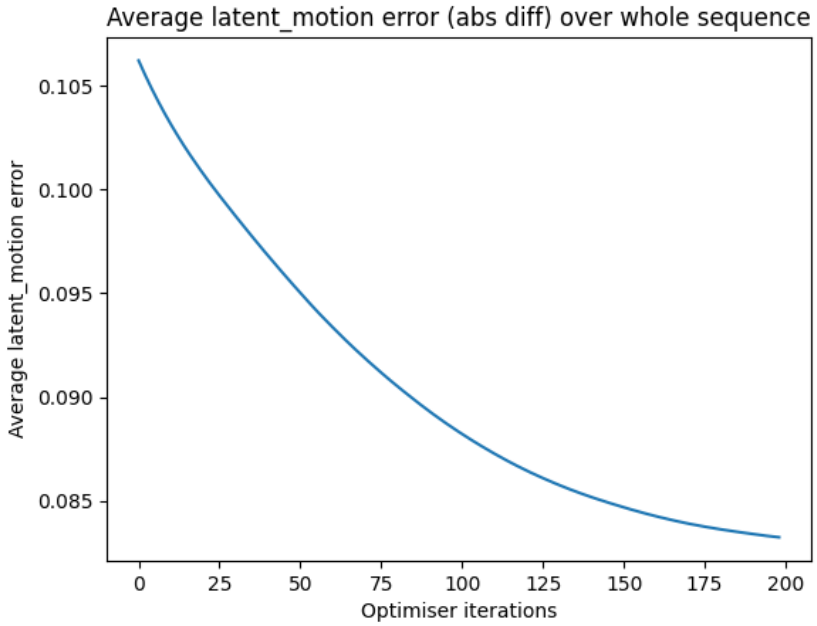


Figure 3.13: *Optimising $\mathbf{z}_{1:T}$ with fixed $\mathbf{x}_{0:T}$*

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.

With the $\mathbf{z}_{1:T}$ fixed at their initial state and after considerable effort to balance the optimiser, we managed to achieve a sensible sitting motion for a clip containing occluded sitting, however 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.

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 a sensible direction.

These experiments showed that the information can be passed sensibly from $\mathbf{x}_{0:T}$ to $\mathbf{z}_{1:T}$ and vice versa, however that this was no easy task.

Next step

The previous experiments all suggest that the next direction to pursue would be an optimisation scheme in which we go back and forth between optimising $\mathbf{z}_{1:T}$ then $\mathbf{x}_{0:T}$. This would avoid the competing goals and difficult to balance optimiser we found when jointly optimising $\mathbf{z}_{1:T}$ and $\mathbf{x}_{0:T}$, and would allow further exploration of the promising results found when optimising $\mathbf{z}_{1:T}$ and $\mathbf{x}_{0:T}$ separately. However considering our difficulty in achieving consistent results for the optimisation of $\mathbf{x}_{0:T}$ with fixed $\mathbf{z}_{1:T}$, our ever growing feeling that this method was not the most efficient formulation of the problem, my desire to try a new direction, and the time constraints,

we deemed it time to move on and investigate a new 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 $\mathbf{x}_{1:T}$ 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 the 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. This investigation did not result in a better model, however allowed us to improve our intuition, understand and fix several issues, and gain more insight into the problem and how best to achieve 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 concluded that the next sensible step would be to investigate a model that operates directly on motion sequences. We wished next to investigate a general-purpose motion generation model that could be readily employed in a wide variety of tasks such that we might gain insight into the broader uses of motion modeling and find unexplored avenues. Due to the promising literature presented in Section 2.2, it was decided that diffusion models operating on motion sequences represented a research direction worth pursuing.

4.1 Method

4.1.1 Diffusion Models

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

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

With a well-chosen variance schedule $\beta_1, ..., \beta_T$ 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 unknown without the full data distribution, which we don't have access to, hence we approximate it using some parametrized function

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

The whole process can be visualized in Figure 4.1 borrowed from the DDPM paper [HJA20].

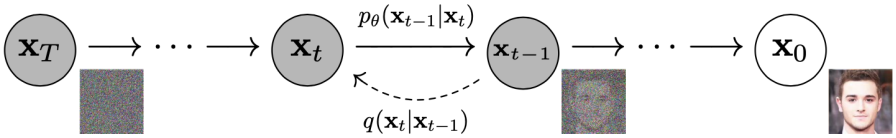


Figure 4.1: *Diffusion Process*

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

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

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

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 variable at any timestep directly from the clean signal, which aids the training procedure described in Section 4.1.2. Next, we also note that we take the following definition for the $\bar{\alpha}_t$ schedule (rather than defining the β schedule), as in [ND21],

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

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

We also use a simplified reverse noising process, where rather than predicting the mean and variance of the noise for a single step as indicated in (4.2), we directly try to predict the denoised signal [RDN+22],

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

which, alongside the choice of a fixed rather than learned variance as in [HJA20], allows us to describe the denoising process as follows, where a tractable expression is provided by Bayes Theorem [ND21],

$$\begin{aligned} p_\theta(x_{t-1}|x_t) &= q\big(x_{t-1}|x_t, \hat{x}_\theta(x_t, t)\big) \\ &= \mathcal{N}(\mu_\theta(x_t, t), \tilde{\beta}_t I) \end{aligned} \tag{4.6}$$

with

$$\begin{aligned} \tilde{\beta}_t &= \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t \\ \mu_\theta(x_t, t) &= \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\hat{x}_\theta(x_t, t) + \frac{\sqrt{\alpha}_t(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}x_t\,. \end{aligned} \tag{4.7}$$

This direct prediction of the denoised signal allows us to use the simplified MSE objective [HJA20, RDN+22] during training

$$\mathcal{L}_{simple} = \mathbb{E}_{x_0,t}\left[\|x_0 - \hat{x}_\theta(x_t, t)\|_2^2\right] \tag{4.8}$$

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

The essential idea of training such a model is to randomly sample a timestep, noise a signal back to this timestep with (4.3), predict the denoised signal with (4.5), and perform a backward step on the loss described in (4.8) alongside some other regularisation losses that suit our problem.

The inference procedure involves sampling random Gaussian noise, then iteratively predicting the denoised signal with (4.5) and sampling from the next timestep with (4.6).

4.1.2 Model Design

Armed with this understanding of diffusion, we move on to the task of designing our model, strongly inspired by the work of Tevet. et. al [TRG+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+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 root orientation and all subsequent poses are described with respect to this first frame, with the idea being that this formulation leads to a smaller space for the model to learn.

Architecture

We follow closely [TRG+22] in our architecture, though without the text conditioning. We use a simple transformer [VSP+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 described denoising process which can be seen in Figure 4.2.

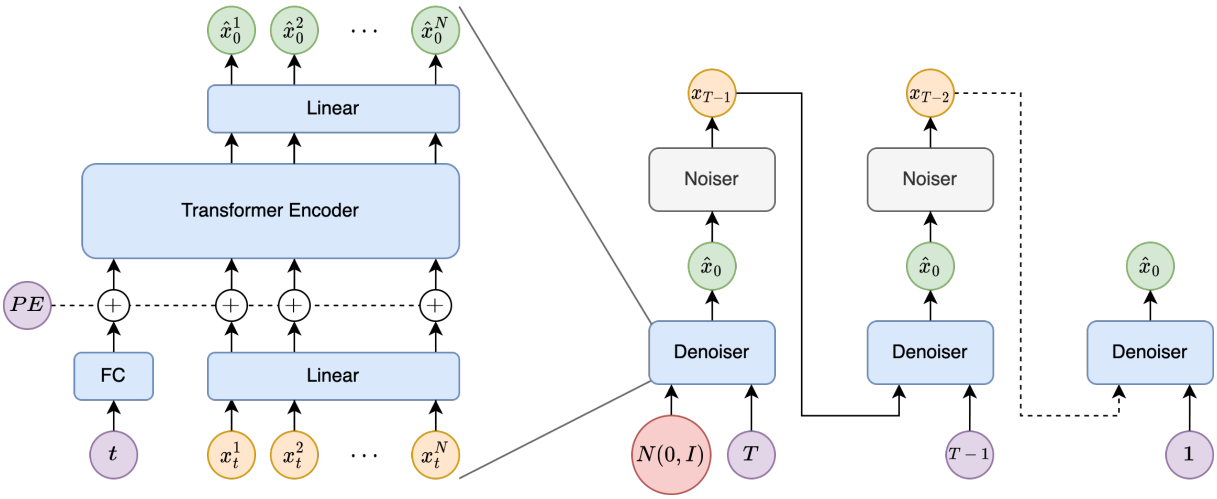


Figure 4.2: *Model Architecture & Denoising Process*

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.

$$\begin{aligned}
\mathcal{L}_{fk} &= \frac{1}{N}\sum_{i=1}^{N} \|x_0^{(pos)} - FK(\hat{x}_0^{(i)})\|_2^2 \\
\mathcal{L}_{vel} &= \frac{1}{N-1}\sum_{i=1}^{N-1} \|(x_0^{(i+1)} - x_0^{(i)}) - (\hat{x}_0^{(i+1)} - \hat{x}_0^{(i)})\|_2^2 \\
\mathcal{L}_{contact} &= \frac{1}{N-1}\sum_{i=1}^{N-1} \| \left(FK(\hat{x}_0^{(i+1)}) - FK(\hat{x}_0^{(i)})\right) \cdot \hat{b}^{(i)}\|_2^2
\end{aligned} \tag{4.9}$$

with $x_0^{(i)}$ representing the ith pose of the motion sequence x_0. The $\mathcal{L}_{fk}$ loss is a forward kinematics loss, i.e a loss on the positions that are calculated from the orientations through a forward kinematic function 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+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 many other use cases 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 $1 - m$,

$$\hat{x}_t = m \odot \hat{x}_t + (1 - m) \odot q(x^{(known)}|t - 1) \tag{4.10}$$

Graphically this process can be seen in the diagram presented in Figure 4.3, and more intuitively in Figure 4.4 where the inpainting progression is shown for the example of replacing leg motion. This procedure is particularly attractive as it requires no special training procedure, and can be employed readily with any diffusion model.

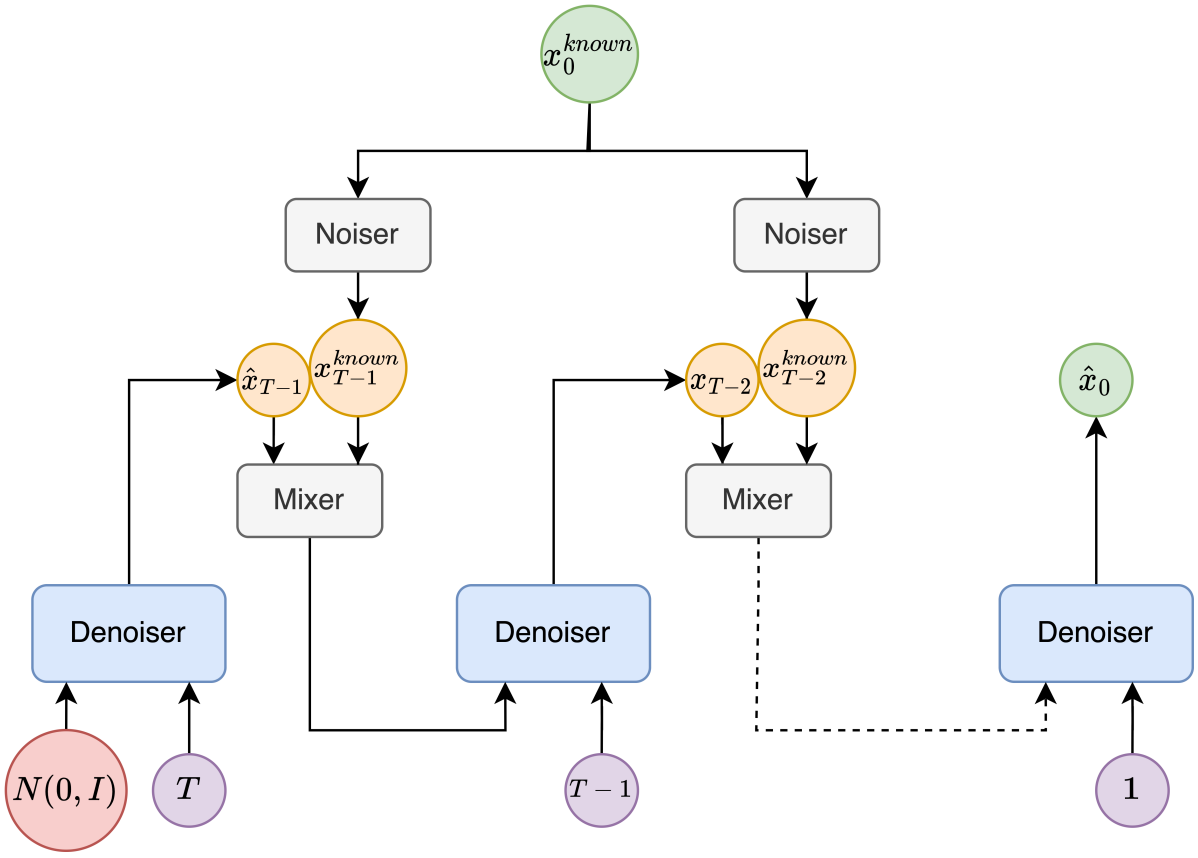


Figure 4.3: *Inpainting Procedure*

Data

We have access to professional motion capture data that encompasses a wide range of possible motion for a number of subjects. In total we have 21 motion sequences on the order of 2 minutes each, thus in total we have around 45 minutes of motion capture data. This data is normalised elementwise and retargeted to a standard skeleton. Several motion sequences are held out for testing. We have one dataset where the root of the skeleton is kept static in the y direction but can move freely in all others, and one version of the dataset that has a freely moving root and for which we have calculated foot contact labels.

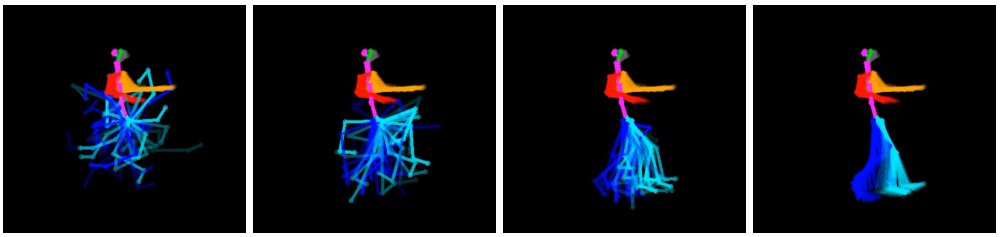


Figure 4.4: *Inpainting: Replacing the legs of a motion sequence*

On the left we can see the first step of the inpainting procedure in which we have replaced the upper body and kept random noise for the lower body. As we proceed in the denoising process, we continually replace the upper body with the known fixed motion whilst denoising the legs.
Note that these images are illustrative and intended to help the reader's intuition. In reality, the upper body is also noised to the corresponding diffusion step but we have shown the denoised version here so that it is obvious what is kept constant.

4.2 Experiments

Or main goal during 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 on which the model's performance can be evaluated. The general structure of the evaluation is to generate a sufficiently large number of sequences for each given task, then import them into the Blender computer graphics software and visually inspect their quality and how well they achieve each task. We have decided to take a qualitative approach as we are attempting to get a better general understanding of the model's capabilities. We note that in the future if a method is deemed worthy of further investigation, more rigorous evaluation metrics would be 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

To test the model's denoising ability, we simply noise a motion sequence back to a given timestep, for example timestep 10, to introduce some amount of gaussian noise, and then denoise this sequence.

Occluded Legs

An important situation in which the existing method does not perform optimally is that of occluded sequences. To test the ability of the diffusion models to handle this situation, we evaluate the model's 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.

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. This process can be visualised in Figure 4.5 where the blue poses represent the given motion, and the gold the motion generated by the model. Again we can use the inpainting framework from Section 4.1.2 and keep the edges of the motion static.

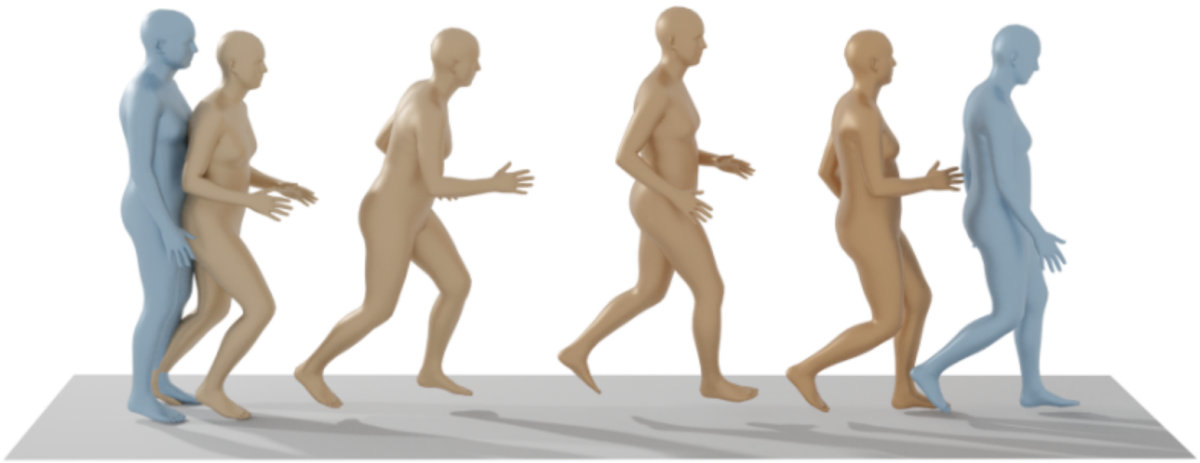


Figure 4.5: *Motion Inbetweening (image from [TRG+22])*

Missing State

To further test the models' ability to handle occlusions, a test case where we only keep a random percent of the state during inpainting is presented. This portion of the state to keep 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 model's ability to recover from longer-term occlusions.

4.2.2 Baseline Model

The baseline model was trained with a dataset that does not, and cannot (due to a lossy rendering of ground truth mocap), contain foot contacts. The state includes the root translation, joint orientations, and their respective velocities explicitly.

Sequence Generation

We found that the model generates a diverse set of plausible motions, as can be seen in Figure 4.6. A few shortcomings were however noted. We found that the model produces a fair bit of foot sliding, where the foot is in contact with the ground and the pose is static, but the feet have a non-zero velocity and are sliding about. This effect usually can be mitigated through the use of the foot sliding loss during training which necessitates access to data containing labeled foot contacts.

Figure 4.6: *Motion Generation - Baseline model*

Denoising

We found that while the model does predict a sensible motion sequence from the noised version, this sequence often is quite different from the original sequence, even if we introduce only a small amount of noise by noising to timestep 10 out of 1000. This may indicate that the model has overfit to the training data, and therefore that it only knows to predict a certain number of motion sequences and thus converges to the nearest one. This could also be due to the effect that we saw of the model drastically changing its predictions during the denoising process, which again may indicate overfitting but may also admit another explanation.

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. An example frame is shown in Figure 4.7. 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/backward directions, therefore the bobbing motion of the upper body, without the additional context of a specific direction, can often be interpreted as both forward and backward walking.

Figure 4.7: *Occluded Legs Inpainting - Baseline model*

The character on the left has the original motion, in the middle the inpainted motion which is derived from the rightmost character, whose leg features are simply Gaussian noise

Inbetweening

The performance on the motion inbetweening task is promising but has some flaws. The model successfully manages to interpolate between the two known motions at the extremities of the sequence and presents plausible motion in the middle, however the generated motion is not

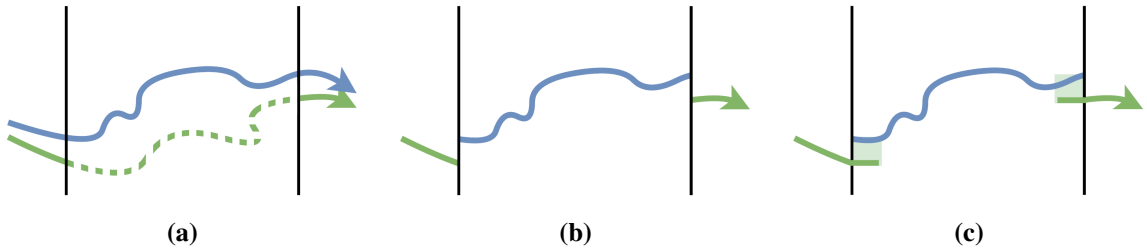


Figure 4.8: *Stage 2* $\mathbf{x}_{0:T}$

The lines represent the trajectories of a motion sequence. The blue line corresponds to the denoised motion, whereas the green line to the ground truth. The black lines represent where the inbetweening begins and ends in the motion sequence. The green known motion line is solid outside the inbetweened part, and this is what we use to overwrite the output of the model at each denoising step.
In (a), the dotted section of the green known motion line represents what a smooth inbetweened sequence would look like. In reality, the denoised motion shown in blue does not perfectly match the known motion at the beginning and end where we will overwrite it. This means that when we do overwrite it we end up with discontinuities, as we can see in (b). The strategy we employ in Section 4.2.5 to try and mitigate this issue involves, at each denoising step, repeating the poses on the edges of the inbetweened motion and blending between the denoised and known motion for these frames, as show in (c).

particularly related to the known motion, often meandering and looking quite distinct from the known motion. Here the text conditioning of [TRG+22] could provide benefits by restricting the possible inbetweened motion. We also note that there is a jump between the known motion and the generated motion. We investigated and found that during the final step of the denoising procedure, the last operation is that of inpainting where the known and generated motions are mixed through a binary mask. This therefore overrides the beginning and end of the generated motion with known motion. If we simply look at the generated motion, it is smooth, however the motion that is overwritten by the known motion does not exactly match the known motion, hence the overwriting introduces a jump, as described graphically in Figure 4.8. This could be mitigated by using a more sophisticated inpainting method as discussed in Section 4.4 and attempted in Section 4.2.5.

Missing State

Random each frame When removing a new random subset of the joints of each pose in 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. 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 known joints, hence it simply needs to interpolate between these known joints.

Random for the whole sequence When choosing a random subset of joints to occlude and occluding them across the entire motion sequence, the model manages to predict a smooth motion sequence that is close to the original, even when only 25% of the joint rotations are

given. However we note the same issue arises that we encountered during inbetweening in Section 4.2.2 in which the inpainting procedure introduces discontinuities that are present even at the end of the denoising process. Again the denoised motion does not perfectly match the generated motion, hence when we overwrite part of the motion with the known motion, this results in discontinuities which produce in this case very jittery motion. Further investigation is necessary to mitigate this issue if we desire that the known state be maintained perfectly.

4.2.3 Model with Foot Contacts - World Space

The second model was trained in a similar manner, but with a different dataset in which the foot contact labels were given, and therefore where an extra foot contacts loss was used. We first tested this model with the motion in world space, as described in Section 4.1.2, meaning that the start point and orientation of each motion are different. We found generally that the model still learned some reasonable motion, however that the root motion was rather disconnected from the rest of the body, resulting in foot sliding. We postulate that this is because we have now massively enlarged the state space, considering the extra dimensions of root motion added and the fact that this is unhomogenised (i.e two identical motions that start in different positions are completely distinct from the model's perspective), and that combined with our limited dataset the model struggles to learn the correlation between root motion and the rest of the body.

4.2.4 Autocompletion

Another potential use of such models 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 again to help to improve the animation pipeline, this time by reducing the amount of work an animator has to do to get a sensible pose. As a side note to the previous experiments to understand the generality of these diffusion models, we trained some diffusion models for the described task, where we enforce a sequence length of 1 so that we are only predicting a 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. An ideal solution would not move the handles at all, and would simply modify the rest of the joints. This again is a manifestation of the issue described in Section 4.2.2 in which the inpainting procedure introduces discontinuities.

Handle Conditioning

To mitigate this issue of the handles being modified, we propose 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 the 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.

Though the implementation was largely completed, we unfortunately did not have time to finish debugging some of the finer details, 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.

4.2.5 Other Experiments

Denoising steps

One drawback of diffusion models is that the denoising procedure is iterative, that is to say if we train with 1000 denoising steps, we then denoise with the same number of steps. Thus, while it is a direct prediction model unlike the optimisation procedure presented in Chapter 3, it still does not provide immediate results. Various strategies have been presented to mitigate this issue. A notably succesful strategy for speeding up the denoising procedure for image generation is that of diffusing in some latent encoding space [RBL+21], thereby reducing the dimensionality of the state that the denoising model operates on. Another strategy, presented in [ND21], is simply denoising with less steps. In essence rather than denoising from step 1000 -> 999 -> 998 -> ... you stride the denoising procedure, 1000 -> 950 -> 900 -> ... which only requires a reweighting of the noise schedule.

We tested this latter strategy on our model, denoising with various numbers of steps. We found that denoising with anything down to 100 steps provided visually identical results to denoising with 100 steps, however, below, with 10 steps for example, the quality of the motion deteriorates. This is an exciting result as it suggests that we can achieve interesting results in a reasonable amount of time, thereby indicating that such a model could be employed in production and provide users with results within an acceptable time frame.

Inbetweening Blending

In Section 4.2.2 we noted a discontinuity between the known motion and the generated motion. We hypothesised that this was due to the inpainting procedure, and that a more sophisticated inpainting procedure could mitigate this issue. We therefore implemented a denoising procedure in which we pad the edges of the known regions of motion with the last known pose, and rather than using a binary mask we blend this padded region with the generated motion smoothly, as described in Figure 4.8. With this procedure we aim to encourage the model to generate a motion that is continuous with the known motion by more smoothly introducing the boundaries between these motions. We find that this procedure, when coupled with the model operating

on motion with freely moving root presented in Section 4.2.3, looks rather like a simple linear interpolation between the two motion. We postulate that this is due to the shortcoming of the model, notably that the root motion is not well learned and so is often very far from the known root motion. However, for the model operating on motion with fixed root presented in Section 4.2.2, we find that this leads to a more correlated inbetweened motion.

4.3 Conclusion

The 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. It is capable of inpainting the legs of a motion sequence, of plausibly filling in randomly occluded states, of inbetweening motion sequences, and of generating new motion sequences from scratch. The model also shows promise for other tasks such as pose autocompletion. This demonstrates that the diffusion framework can learn a general-purpose motion prior model without task-specific fine-tuning or architecture design. We are positive that future research directions in which the specific tasks to solve are more precisely described, and where more fine-grained architecture, data and training choices could be made, would prove fruitful avenues with lots of opportunity. The implementation of the diffusion framework paves the way for further investigation into its uses in the context of general animation tasks and proves itself as another useful tool that can be drawn upon in the future.

4.4 Future Work

We have demonstrated that diffusion models represent an exciting research avenue and have laid a foundation of code for future research. The future directions that could build upon this work are numerous and potentially rich in their rewards. We will discuss some of the most promising avenues for future work in this section, and will also discuss some of the shortcomings of the work presented in this thesis.

4.4.1 Model

Some of the shortcomings of the model seen in Inpainting tasks, most notably in the Inbetweening task, are due to an inconsistency between the specified known motion and the motion subsequently generated during inpainting by the model. For the inbetweening task, if we avoid overwriting the generated motion with the known motion in the last denoising step, the motion is smooth over the whole sequence and the transition between known and generated motion contains no jump. The generated and known motion however do not match perfectly even at the last denoising step, that is to say, the part in the generated motion which will be overwritten by the known motion is not exactly the same as the known motion, and so when we override the generated motion with the known motion for the last time, we lose the smoothness of the motion and introduce a discontinuity between the generated and known motion. This effect

was not just present in inbetweening, however also in the autocomplete and in replacing random missing joints over a whole sequence, and might be mitigated through several strategies. Firstly we might condition the model with an indication of where a known area is specified and penalize it during training for deviating from this known motion (as described in Section 4.2.4 but for sequence-level motion). Another method would simply be that of a postprocessing step, in which the known motion and the generated motion could be mixed for several frames at the transition point, thus achieving a smooth motion between the two. We must also note that this effect might be present in overfit models, as the model would be less free in the motion it is capable of generation, thus literature search into the presence of this effect in other models and more generally into modification to the inpainting procedure should be conducted.

There are many more exciting avenues for creating more task-specific models, for example, an inbetweening model could include a timestep embedding indicating the time till completion of the inbetweened motion, with the idea that the model can better blend the known and generated motion if has access to explicit information indicating at what point the motion should transition between the two. More generally, for any inpainting task, it could be interesting to condition explicitly on the known data and to penalize deviations from this known data during training.

The model training could also be improved. The stopping criteria are 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+22], in which they retrospectively choose the checkpoint that minimizes the FID score.

Finally, more work could be directed toward the question of tuning the diffusion framework to motion modeling. Much of the literature [TCL22, TRG+22, YSI+22] bases itself primarily on the diffusion framework presented in [HJA20] and satisfies itself in extending the DDPM implementation of diffusion. Therefore important, diffusion-specific hyperparameters such as the number of noising steps are not seemingly well explored in the context of motion modelling.

4.4.2 Model use

We noted through our studies that during the denoising procedure, the denoised x_0 often changed drastically throughout the process. At the beginning of the procedure when the network is predicting a denoised x_0 from Gaussian noise, it often resorts to predicting some mean motion. As this is incorporated into the latents the predicted x_0 motion shows more and more diversity, what we noted however is that this motion can change drastically, and can do so late in the denoising process. Intuition would suggest that as we become closer to the end of the denoising process, where there is less and less noise present in the latent variables we are denoising, the model should become more and more certain of its predicted motion and should not drastically change it. However, it sometimes did exactly that. This phenomenon is worth investigation, as it may indicate a flaw in the network, or perhaps represents an area for improvement.

We could also consider more test cases for the model to be evaluated on. Motion switching, which is a specific form of inbetweening where we attempt to join two distinct motion sequences (e.g from performing karate moves to sitting down) could also prove an interesting test. Many

of the tests that were presented above did not require the model to inpaint a root translation, thus more tests where this is necessary will be an important avenue.

We could also consider test cases containing more structured noise, such as we encountered during the automated motion capture investigation in Chapter 3 where legs are randomly swapped left to right and in which joints can be arbitrarily sticking out due to bad 2d pose estimates. We might approach this problem by modifying the inpainting procedure to deal with confidence estimates for the joints, performing the mixing between known and generated motion not through a binary mask, but a confidence-based mask.

4.4.3 Model evaluation

We presented an entirely qualitative evaluation of the models. As discussed previously, this was motivated by a shorter available time and a desire to get a better understanding of the models before investing time in a more rigorous evaluation. This however does not result in publishable or definite work, but rather a stepping stone toward such work indicating that it might be fruitful. Future work should therefore make more effort to investigate the models quantitatively. Relevant metrics should be explored for the tasks at hand (FID, KID, precision/recall, diversity [RLL+22]). It would also be sensible to compare generated motion to the dataset, to see if the network has simply learned to memorise the dataset or if it is generating novel motion. Alternative baseline models should be discussed as well and comparisons should be made. For example, the missing state evaluation where a new subset of the joints is removed from each frame might well be accurately reconstructed by simple linear interpolation between the missing frames, so the model's performance might not be so impressive after all. It must also be noted that for the data with no forward/backward motion, the visualisations effectively hide/minimise foot sliding, as there is no global root translation a foot sliding while walking looks very similar to a foot that doesn't slide. As we saw with the data with a global root translation, the foot sliding is a serious issue and so this issue must be addressed.

4.4.4 Data

We note that our dataset, with approximately 45 minutes of motion spanning 4 subjects is significantly smaller than the AMASS [MGT+19] dataset used by many other papers that contains more than 40 hours of motion data spanning 300 subjects. We, therefore, expect that our model would generalise better 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. If any of the models we explored in this thesis or subsequent work are to be productized, 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 dealing with motion far from the data distribution we currently have, as the more obscure and intricate the motion sequence the harder it would be to animate and thus more value our system could provide.

Conclusion and Outlook

We embarked upon the work presented in this thesis to explore motion modeling in the context of a project to capture motion data direct from RGB videos. To begin with, we explored the relevant literature in Chapter 1, gaining a broader understanding of the field of motion modeling, and placing emphasis on the common Autoencoder and VAE architectures and the promising field of diffusion models. Next, we selected the HuMoR method [RBH$^+$21] as a base for an investigation, with the view that the model's strong performance and use for rectifying motion sequences obtained from 2d pose estimates could lead to an important improvement over the existing automated motion capture system. We investigated the model in Chapter 3, finding that the model provided promising results however at a huge computational cost. We set out to improve this speed issue in Section 3.3, presenting a novel strategy for optimisation that subdivided the autoregressive nature of the HuMoR method at the heart of the performance issue. Considerable effort was made to obtain comparable results, however the conclusion was reached that the results were not nearing the required quality and that the desired speedup seemed tricky to achieve. With the lessons of this experience on hand we formulated a new research direction, choosing to investigate a sequence-level model, rather than the local pose-to-pose model of HuMoR due to the large potential speedup, and moved to the novel diffusion framework for which a promising literature is emerging. We explored in detail the diffusion framework in Chapter 4 and formulated a baseline architecture upon which to base our investigations. We evaluated on a wide range of different tasks in Section 4.2, finding that the model can successfully replace missing legs and random missing joints, that it shows promise for motion inbetweening, that it generates diverse plausible motion, that it can be used for other tasks such as pose autocompletion, that the diffusion denoising procedure is flexible and can be sped up, and that the inpainting procedure can be modified to improve performance. Finally, we discussed the many exciting future research directions in Section 4.4. We are hopeful for the future of diffusion models in motion modeling and are thrilled to have provided a notion of their potential and a building block upon which to base future research.

Bibliography

[BKG21] Dor Bank, Noam Koenigstein, and Raja Giryes. Autoencoders, 2021.

[BKL+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+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+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+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.

[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.

[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+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+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+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+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+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+21] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models, 2021.

[RDN+22] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents, 2022.

[RLL+22] Sigal Raab, Inbal Leibovitch, Peizhuo Li, Kfir Aberman, Olga Sorkine-Hornung, and Daniel Cohen-Or. Modi: Unconditional motion synthesis from diverse data, 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.

[TC] Taheri and Choutas. Vertex ids file of the smpl-x project. `https://github.com/vchoutas/smplx/blob/main/smplx/vertex_ids.py`. Accessed: 2023-05-17.

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

[TRG+22] Guy Tevet, Sigal Raab, Brian Gordon, Yonatan Shafir, Daniel Cohen-Or, and Amit H. Bermano. Human motion diffusion model, 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.

[VSP+17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz 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+22] Ye Yuan, Jiaming Song, Umar Iqbal, Arash Vahdat, and Jan Kautz. Physdiff: Physics-guided human motion diffusion model, 2022.

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

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Declaration of originality

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

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

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

Title of work (in block letters):

MOTION PRIORS FOR POSE ESTIMATION AND ANIMATION WORKFLOWS

Authored by (in block letters):

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

Name(s):	**First name(s):**
SMITH	LUKE ANDREW

With my signature I confirm that

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

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

Place, date	**Signature(s)**
Zürich, 19.05.2023	

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