Motion In-betweening using Diffusion Models

Silvia Arellano García
Author
silviaag@kth.se

Rajmund Nagy
Supervisor
rajmundn@kth.se

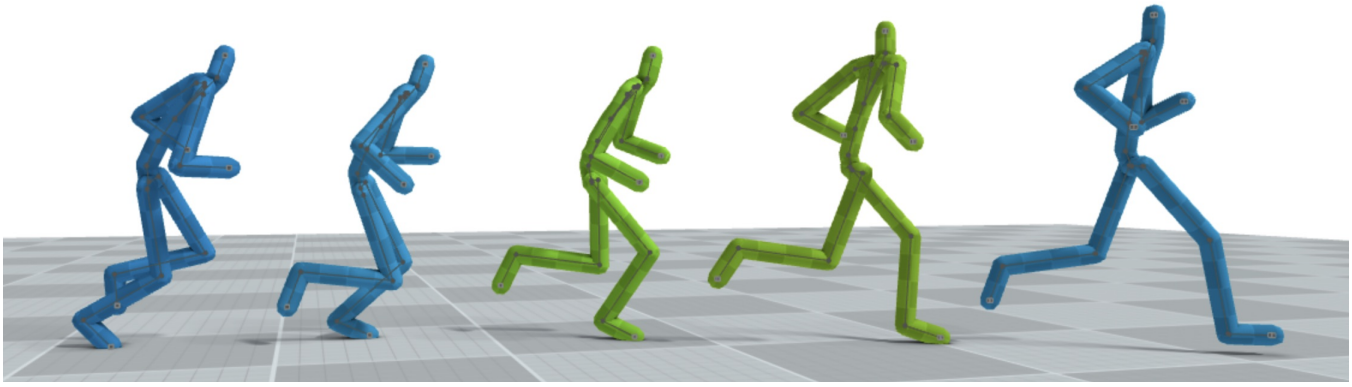

Figure 1: Illustration of the inbetweening task. Given some keyframes (blue skeletons), our model generates plausible intermediate frames (green skeletons) to reconstruct smooth and natural motion transitions.

ABSTRACT

Motion in-betweening involves generating smooth motion sequences that transition between keyframes provided by the user. This technique is especially useful in character animation, where manually editing every frame can be time-consuming and requires significant skill. By automating the in-betweening process, we can significantly reduce the effort needed to create fluid, natural animations. This project aims to implement a deep generative network that generates the missing frames of a sequence, with gap sizes ranging from 1 to 15 frames. Our model generates both the root positions and rotations for the missing frames, producing smooth and realistic motion across a variety of actions. We evaluate the model's performance using the LaFAN1 motion capture dataset, which includes diverse actions performed by different actors. The evaluation includes both quantitative and qualitative comparisons with interpolation methods and related work.

1 INTRODUCTION

Computer-animated characters are becoming more widespread in animated films and video games. Creating smooth and natural movements for these characters often involves generating frames between key poses, a process known as in-betweening. Some animators start from a version where the missing frames are substituted with ones computed using interpolation. However, an animation specialist can take more than 5 hours to adapt 3 transition frames [7] of a single character. To enhance the animator's workflow, several approaches have explored using deep neural networks to generate missing frames from a sparse set of keyframes provided by the animator. This allows the network's output to serve as a more refined starting point compared to interpolation or creating frames from scratch, significantly reducing the time and effort required.

Deep generative networks typically require large amounts of data to produce high-quality outputs. However, the availability of high-quality motion capture datasets is limited, presenting a significant challenge for training such models. Additionally, the nature of motion data itself adds complexity, as it involves both spatial and temporal dimensions. This dual complexity is intensified by the fact that humans are highly sensitive to even minor irregularities in motion, making the task of generating realistic sequences particularly demanding.

This project aims to develop a diffusion-based model capable of generating smooth and realistic motion to fill gaps of varying lengths and address diverse types of actions. A key objective of this work is to gain a deeper understanding of diffusion models by implementing the approach almost entirely from scratch. Furthermore, we seek to evaluate the results both quantitatively and qualitatively, comparing the model's performance against established baselines.

The report is structured as follows. Section 2 introduces related work to provide context and highlight the inspirations behind this project. Next, Section 3 presents a detailed description of the studied problem and the models used to address the motion in-betweening task. In Section 4, we describe the dataset utilized and examine the details of the network training process. Then, in Section 5, we evaluate the performance of our method both quantitatively and qualitatively. Section 6 explores additional experiments to provide a deeper understanding of our model. Finally, Section 7 discusses the limitations of our method and outlines directions for future work, while Section 8 summarizes our findings.

2 RELATED WORK

This section reviews several recent motion in-betweening methods. One of the first approaches to tackle this task using neural networks was proposed by Harvey et al. [5]. They introduced an RNN-based

motion predictor with two notable features: a time-to-arrival embedding and an additive scheduled target noise vector to enhance robustness. However, this approach struggles with generating long sequences. Additionally, they presented the LaFAN1 dataset, which is utilized in our project.

In recent years, diffusion models have gained popularity in generative applications. Some works have successfully applied these models to motion synthesis, achieving impressive results not only in locomotion but also in domains like dancing and co-speech gesture generation [1, 2, 9]. Diffusion models have also been used for in-betweening tasks, as demonstrated by Cohan et al. [3], who proposed a diffusion model conditioned on keyframes and text prompts. Inspired by their high-quality results, we decided to experiment with this architecture in our project.

Other inspiring approaches include the work of Oreshkin et al. [6], who propose a transformer-based architecture that computes motion differences between the linear interpolation and the reference motion sequence for the missing gaps. This approach motivated us to explore different data representations, as we hypothesize that the choice of data representation can significantly influence results. Another innovative method is proposed by Starke et al. [7], whose in-betweening system uses phase variables learned by a periodic autoencoder. Their approach excels in producing realistic movements and transitions that are rare in existing datasets and successfully synthesizes motion for keyframes with long missing-frame gaps.

3 MOTION IN-BETWEENING FRAMEWORK

Our motion in-betweening framework predicts the character's pose for a set of missing frames simultaneously. However, we divide this task into two parts. The first involves generating the rotations for every joint in the missing frames, while the second focuses on generating the root positions for those frames. Once both components are generated, the outputs are combined. In this section, we provide a more detailed explanation of the problem and further insights into the models used.

3.1 Problem definition and motion representation

One way of storing motion data is through BVH files, where motion is represented as a hierarchical structure of joints with associated positional and rotational data. A BVH file consists of two main sections: the hierarchy, which defines the skeletal structure, including joint connections and offsets, and the motion section, which contains the frame-by-frame rotational transformations for each joint.

In a similar manner, we store the offsets in a vector $\mathbf{X} \in \mathbb{R}^{B\times N\times J\times 3}$, where B is the batch size, N is the total number of frames in the sequence, and J is the number of joints in the skeleton. This vector includes the global root position for the root joint and the offsets for the remaining joints. The rotational information is stored using ortho6D [10], resulting in a vector $\mathbf{Q} \in \mathbb{R}^{B\times N\times J\times 6}$. It is worth noting that $\mathbf{X}$ and $\mathbf{Q}$ include both known and unknown frames, with the unknown frames initialized to zeros. The objective of our model is to predict these masked frames, ensuring that the complete motion is both realistic and coherent.

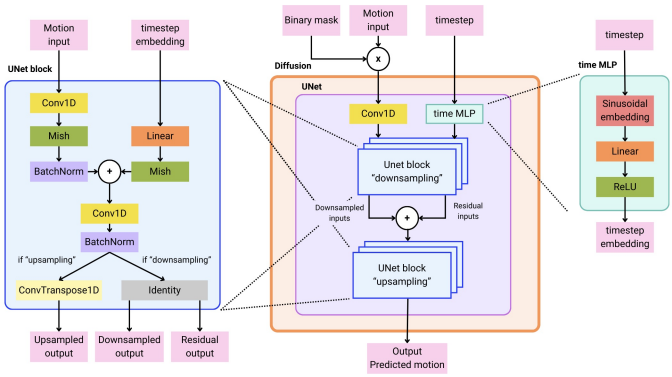

Figure 2: Diagram of the diffusion model with a UNet and 1D convolutional layers used for generating the joint rotations.

3.2 Joint-Rotation generation

For this part of the network, we implemented a diffusion-based model with a UNet architecture and 1D convolutional layers, as illustrated in Figure 2. The model operates within the diffusion framework, where noise is gradually added to the input data during training, and the network learns to reverse this process to predict clean, realistic outputs.

The input to the model is the vector with the joint rotation information $\mathbf{Q}$. A binary mask is applied to distinguish known frames from missing ones, and a timestep embedding, processed through a time MLP, provides temporal context for the diffusion process. The UNet architecture consists of downsampling and upsampling operations, which extract and reconstruct motion features at different resolutions. The final output of the network is the predicted joint rotations, denoted as $\hat{\mathbf{Q}}$.

3.3 Root position generation

To generate the root position information $\hat{\mathbf{X}}$, we use an LSTM network with 3 layers, each containing 256 hidden units. The network processes the root positions from the input motion sequences, learning the temporal dependencies between frames. It generates the root positions frame-by-frame for the missing frames, using $\mathbf{X}$ as input. The final output layer produces the predicted root position for each frame in the sequence. The model computes the loss using the mean squared error (MSE), comparing the predictions with the reference sequence.

4 NETWORK TRAINING

This section outlines the training process of the model and the data preprocessing steps. We use the full LaFAN1 dataset [5] from Ubisoft to train our model. Following the approach of previous works [5–8], we use the captures from Subject 5 as the test set, use Subjects 1 to 3 to extract features for the training set, and reserve Subject 4 as the validation set. In total, 64 sequences are processed, comprising 399,139 motion frames (approximately 3.69 hours at 30fps). The sequences are split into shorter segments of 50 frames, with a 20-frame offset between them.

Training the rotation network with a batch size of 256 took 9.83 hours and included 968,500 optimization steps. This duration is

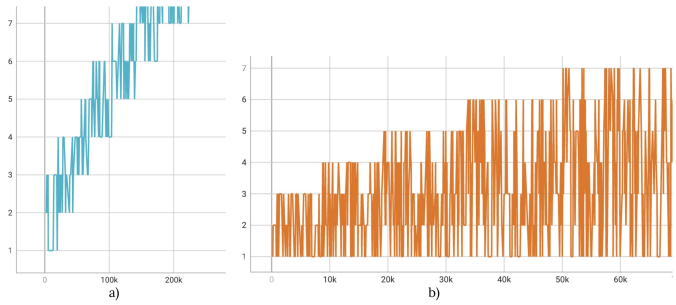

Figure 3: Progression of the number of masked frames over training. a) corresponds to the joint-rotations network, b) corresponds to the root position network.

primarily due to our strategy for handling the number of masked frames. To ensure the network gradually learns more complex tasks, we progressively increase the number of masked frames. Specifically, the gap size is randomly selected from a range of three possible sizes, which increases over time in a sliding window fashion. This approach allows the gap size to shift progressively, covering a larger portion of the sequence, as illustrated in Figure 3a. Toward the end of training, we perform additional training with a fixed gap size of 15 to ensure it receives a comparable amount of training to the other gap sizes.

Training the root position generation network took 4.91 hours, equivalent to 648,448 optimization steps with a batch size of 256. For this task, a different strategy for modifying the gap size proved more effective. Instead of selecting the gap size from a fixed range, we progressively expand the range over time, while still allowing smaller gap sizes to be sampled. This approach is illustrated in Figure 3b.

In both cases, as the gap size increases we allocate additional training time, since larger gaps correspond to a more complex prediction task.

This progressively increasing gap size made the loss curves more challenging to interpret, as they do not exhibit a strictly monotonic decrease. The validation loss for both models is shown in Figure 4. For both cases—the joint-rotation generation model and the root position generation model—the loss decreases within each interval before the range of the gap size is expanded. In the root position model (Figure 4b), the loss consistently converges to a similar value across all intervals, suggesting stable learning. However, this behavior does not hold for the joint-rotation model (Figure 4a), where the loss begins each interval at a higher value and converges less effectively.

It is important to note that the losses are not directly comparable, even though both are computed as the mean squared error (MSE). This is because the quantities being measured differ. Specifically, the root position loss considers only the root joint across 3 dimensions, while the joint-rotation loss accounts for all joints across 6 dimensions. Additionally, the loss is calculated over all dimensions of the masked frames without normalization by the number of masked frames, further complicating direct comparisons.

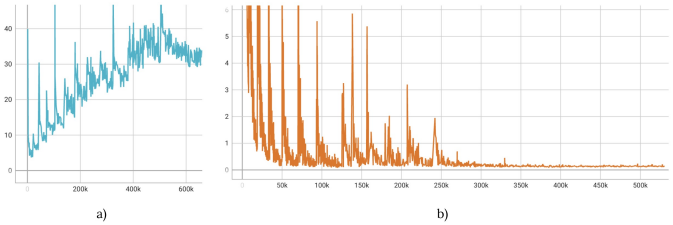

Figure 4: Validation loss of a) to the joint-rotations network and b) the root position network.

5 EVALUATION

We evaluate our results both quantitatively and qualitatively, comparing them to several baselines and related approaches. As metrics, we use the L2Q (the global quaternion squared loss, Eq. 1), L2P (the global position squared loss, Eq. 2) and NPSS (the normalized power spectrum similarity score, Eq. 5) [4].

$$\textbf{L2Q} = \frac{1}{|\mathcal{D}|}\frac{1}{\mathcal{T}}\sum_{s\in\mathcal{D}}\sum_{t\in\mathcal{T}}\left\|\hat{\mathbf{q}}_t^s - \mathbf{q}_t^s\right\|_2, \tag{1}$$

$$\textbf{L2P} = \frac{1}{|\mathcal{D}|}\frac{1}{\mathcal{T}}\sum_{s\in\mathcal{D}}\sum_{t\in\mathcal{T}}\left\|\hat{\mathbf{p}}_t^s - \mathbf{p}_t^s\right\|_2, \tag{2}$$

In these equations, $|\mathcal{D}|$ represents the number of test sequences considered, and $\mathcal{T}$ denotes the gap size that the model aims to reconstruct. For this evaluation, we consider $\mathcal{T} \in 5, 10, 15$. The term $\mathbf{q}_t \in \mathbb{R}^{J\times 6}$ corresponds to the 6-dimensional vector representing the global rotations of all skeletal joints at timestep t, while $\mathbf{p}_t \in \mathbb{R}^{J\times 3}$ corresponds to the global position of all skeletal joints.

To define the NPSS metric, let $X_{i,j}[f]$ and $Y_{i,j}[f]$ denote the squared magnitude spectra of the discrete Fourier transform coefficients for the ground-truth and predicted motion data, respectively, for the j-th feature of the i-th sequence. These are normalized as:

$$X_{i,j}^{\text{norm}}[f] = \frac{X_{i,j}[f]}{\sum_f X_{i,j}[f]}, \quad Y_{i,j}^{\text{norm}}[f] = \frac{Y_{i,j}[f]}{\sum_f Y_{i,j}[f]}. \tag{3}$$

The Earth Mover's Distance (EMD) between these normalized spectra is then computed as:

$$\text{emd}_{i,j} = \|X_{i,j}^{\text{norm}}[f] - Y_{i,j}^{\text{norm}}[f]\|_1. \tag{4}$$

Finally, the NPSS score aggregates these distances using a power-weighted average:

$$\text{NPSS} = \frac{\sum_i\sum_j p_{i,j}\cdot \text{emd}_{i,j}}{\sum_i\sum_j p_{i,j}}, \quad p_{i,j} = \sum_f X_{i,j}^{\text{norm}}[f]. \tag{5}$$

Here, $p_{i,j}$ represents the total power of the j-th feature in the i-th sequence. This metric captures differences in the distribution of the power spectra, providing a robust measure of motion similarity.

Table 1 compares the quantitative metrics of interpolation and our approach. We observe that as the number of frames increases, our model performs significantly better. We hypothesize that this happens for several reasons. First, the way we progressively increased the gap sizes during training might have caused the model

Table 1: Comparison of Interpolation and our approach. Lower score is better.

Metric	Gap size	Interpolation	Ours
L2Q	5	**0.223**	0.330
	10	**0.450**	0.493
	15	0.637	**0.471**
L2P	5	**0.335**	0.513
	10	**0.742**	0.858
	15	1.043	**0.894**
NPSS	5	**0.023**	0.032
	10	**0.084**	0.086
	15	0.158	**0.105**

to focus more on solving the harder problems with large gaps, at the cost of slightly neglecting the simpler, shorter ones. Another reason is that interpolation performs better for short gap sizes because the motion remains smooth and predictable. For longer gap sizes, however, the motion becomes more complex and harder to capture with simple linear methods. It is also worth noting that the quantitative metrics are computed with respect to the reference frames. However, there are other plausible solutions that, while different from the reference, could still be valid even if they give a higher error.

The qualitative differences are more noticeable, especially for larger gap sizes. Our model reconstructs a plausible action that aligns with the overall sequence, whereas interpolation results in a fake-looking transition that often does not correspond to the expected action. This is particularly evident in sequences with quick movements, where interpolation struggles to produce realistic motion.

However, our model is not perfect. One of the most notable issues is that the root position drifts slightly below the floor level. In these cases, the skeleton performs the correct action, but its position appears lower than expected. This occurs because the root position generation and joint-rotations generation are handled independently. If the root position does not align perfectly with the joint rotations, the skeleton's position can shift downward. Interpolation avoids this issue by ensuring that the starting and target positions of all joints always remain above the floor. Another issue is foot sliding, where the feet move when they should remain fixed. While this artifact is much more pronounced in interpolation, it can still appear in our model during complex actions.

In summary, interpolation works well for short gap sizes, but its limitations become evident as the gaps grow larger. Our model produces more realistic and plausible transitions overall, despite minor artifacts like root sinking and foot sliding.

6 ADDITIONAL EXPERIMENTS

6.1 Analysis of the models separately

In this section, we will explore different modifications of the model evaluated in Section 5, which we will refer to as "$\hat{\mathbf{Q}}$ and $\hat{\mathbf{X}}$" or "final model" for the rest of the section. First, we will analyze a model where only the joint rotations are generated, and the root positions are taken from the original dataset. We will call this "$\hat{\mathbf{Q}}$ and $\mathbf{X}$". Next, we will examine a variation where only the joint rotations are generated but conditioned on the root positions. This model will be referred to as "$\mathbf{Q}$ and $\hat{\mathbf{X}}$". Finally, we will evaluate a model that generates the root position while using the original dataset's joint rotations. We will refer to this model as "$\mathbf{Q}$ and $\hat{\mathbf{X}}$". The goal of this section is to assess the performance of each model individually.

Table 2: Comparison of our final model with models that only generate the joint rotations.

Metric	Gap size	$\hat{\mathbf{Q}}$ and $\hat{\mathbf{X}}$	$\mathbf{Q}_{interp}$ and X	$\hat{\mathbf{Q}}$ and X	$\hat{\mathbf{Q}}_{cond}$ and X
L2Q	5	0.330	**0.223**	0.320	0.332
	10	0.493	**0.450**	0.479	0.505
	15	0.471	0.637	**0.460**	0.523
L2P	5	0.513	**0.215**	0.341	0.346
	10	0.858	**0.483**	0.573	0.596
	15	0.894	0.712	**0.546**	0.612
NPSS	5	0.032	**0.023**	0.029	0.033
	10	0.086	0.084	**0.079**	0.085
	15	0.105	0.158	**0.099**	0.120

First, in Table 2, we examine the results of different approaches that generate only the joint rotation values using the root position from the original dataset. For this, we use the model described in Section 3.2 and vary the input. We consider two approaches: "$\hat{\mathbf{Q}}$ and $\mathbf{X}$" and "$\hat{\mathbf{Q}}_{cond}$ and $\mathbf{X}$".

Both approaches show similar performance, with better results for larger gaps compared to smaller ones, and both outperform interpolation for a gap size of 15. We hypothesize that this occurs for similar reasons as discussed in Section 5. Interestingly, the approach "$\hat{\mathbf{Q}}_{cond}$ and $\mathbf{X}$" performs slightly worse than "$\hat{\mathbf{Q}}$ and $\mathbf{X}$". This may happen because the model could struggle to effectively handle the complexities introduced by the conditioning.

Next, we analyze the results of generating only the root positions using the joint rotation information from the original data, the "$\mathbf{Q}$ and $\hat{\mathbf{X}}$" model. The results are shown in Table 3. We observe that the differences are more pronounced in this case, as the model tends to overfit to the task of generating larger gap sizes, forgetting what it learned in earlier stages of training. This is evident from the L2P value, which decreases even if the number of frames to predict increases. Qualitatively, the results show plausible root positions with no apparent issues. In the root plot, small differences can sometimes be observed along the X-Y axes (e.g., the hips may appear slightly higher or lower relative to the floor compared to the reference frame), as illustrated in Figure 5. However, in most cases, the generated solutions are equally valid. When analyzing the plot, it is important to consider the difference in scale on the Y-axis.

6.2 Change the way of increasing the gap size

The way we increase the gap size during training affects how the model handles more complex problems. Our goal is to teach the model to solve easier tasks first and progressively move on to more difficult ones. However, once training is complete, we want the model to be able to handle problems across the entire range of

Table 3: Comparison of our final model with models that only generate the the root position.

Metric	Gap size	$\hat{Q}$ and $\hat{X}$	Q and X_{interp}	Q and $\hat{X}$
L2P	5	0.513	**0.236**	0.645
	10	0.858	**0.513**	0.575
	15	0.894	0.704	**0.350**

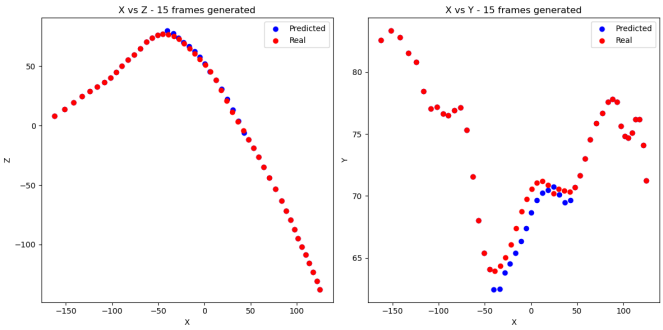

Figure 5: 2D root trajectory plots from a top view (left) and a side view (right). The sequence corresponds to a 90 degree turn, with 15 middle frames generated by the model.

gap sizes seen during training with consistent quality. Therefore, in addition to the approach presented in Figure 3a, we explored two other methods for adapting the gap size during training. This study was conducted using the model that generates joint rotations for the missing frames, conditioned on the root positions of the sequence, as introduced in Section 6.1.

The first method we tried involved uniformly sampling a gap size from the entire range of possible sizes (1 to 15) throughout the training process, as shown in Figure 6a. This posed a significant challenge to the model, which required much more time to reduce the loss. After 12 hours of training, the model still did not generate acceptable motion. Despite the loss not having fully converged, we decided to stop the training, as continuing would have been much more computationally expensive compared to other approaches.

The second method, illustrated in Figure 6b, involved progressively increasing the gap size during training. The model would first focus on generating a single missing frame, then gradually move to larger gaps (e.g., two missing frames, and so on). We call this the "step increase strategy". This approach yielded better results than the previous one, but it was more challenging for the model to perform well compared to the method in Figure 3a, where the gap size was sampled from a range of three values. The comparison between the two methods is shown in Table 4. This outcome is reasonable because, in the progressive approach, the model focuses on a new task at each stage, rather than maintaining continuity by reinforcing what it learned in earlier stages.

6.3 Larger gap size

In-betweening models are designed to alleviate the workload of animators. Therefore, the more frames they can generate, the better.

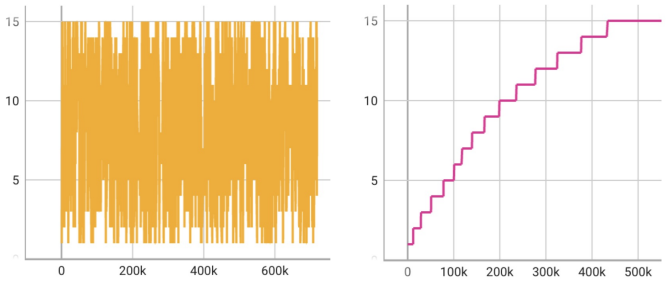

Figure 6: Approaches explored for varying the gap size during training.

Table 4: Quantitative comparison of gap-size scheduling strategies, applied to the "$\hat{Q}_{cond}$ and X" model, described in 6.1. "Step increase" refers to the approach described in Section 6.2. A lower score indicates better performance.

Metric	Gap size	Final model	Step increase
L2Q	5	**0.332**	0.355
	10	**0.505**	0.551
	15	**0.523**	0.561
L2P	5	**0.346**	0.395
	10	**0.596**	0.670
	15	**0.612**	0.697
NPSS	5	**0.033**	0.034
	10	**0.085**	0.095
	15	**0.120**	0.126

Until now, our model has been able to handle a maximum gap size of 15 frames, producing completely incorrect or invalid outputs for longer gaps. One possible solution to this issue is increasing the kernel size, as it allows the model to capture longer-range dependencies. However, this approach may come at the cost of losing finer details in the motion.

To test this, we used the "$\hat{Q}_{cond}$ and **X**" model, and compared its performance to outputs where the joint rotations were generated using interpolation. As before, we observed that the differences are more pronounced for smaller gaps, indicating that the model performs better on tasks encountered during the later stages of training. To evaluate its performance for larger gap sizes, we trained the model until it started to converge for a gap size of 25. This required a longer training period for the larger gap size compared to others.

We hypothesize that dedicating less time to training for a specific gap size and distributing training time more evenly across all gap sizes could compromise performance for the most challenging case (gap size of 25). However, this approach might improve results for smaller gap sizes, as it would reduce the risk of overfitting to a single task. As shown in Table 5, interpolation outperforms the model in terms of quantitative metrics. Nevertheless, the model's outputs correspond to plausible motions, whereas the interpolation

Table 5: Comparison of only generating the joint rotations ($\hat{\mathbf{Q}}$), while the root position (X) is taken from the reference samples. A lower score indicates better performance.

Metric	Gap size	$\mathbf{Q}_{interp}$ and X	$\hat{\mathbf{Q}}_{cond\ K7}$ and X
L2Q	5	**0.223**	0.321
	10	**0.450**	0.512
	15	**0.637**	0.671
	20	0.782	**0.761**
	25	0.899	**0.792**
L2P	5	**0.215**	0.387
	10	**0.483**	0.697
	15	**0.712**	0.940
	20	**0.889**	1.073
	25	**1.016**	1.115
NPSS	5	**0.023**	0.030
	10	**0.084**	0.088
	15	0.158	**0.151**
	20	0.229	**0.206**
	25	0.304	**0.238**

results deviate significantly from expected behavior. The generated motions are not clean, with movements appearing shaky and requiring post-processing. Despite this, the model provides a reasonable approximation of how the motion might look within the missing gap, partially fulfilling the objective of in-betweening.

Other alternatives to address this problem and handle larger gap sizes include using dilated convolutions, which provide a wider receptive field without increasing kernel size, or exploring other architectures, such as transformers, which can efficiently capture long-range dependencies through their self-attention mechanism.

6.4 Other neural networks as baseline

We also wanted to test whether simpler architectures could perform well on our problem, to better understand how diffusion models were affecting performance. For this purpose, we selected the task where the model generates the missing joint rotations conditioned on the root position.

First, we tested an LSTM with 3 layers and a hidden size of 512. This architecture was chosen because it had previously performed well for generating root positions, and we wanted to evaluate its performance with more complex data. Quantitatively, the metrics are significantly worse (by more than 2 units) compared to the results presented in Section 5, both for our model and the interpolation baseline. Qualitatively, the generated sequences resemble interpolation outputs: they often solve the problem using a motion artifact where joints appear to glide, rather than following natural skeletal constraints. However, the transitions between the known and missing frames are abrupt, and the outputs frequently contain sudden changes that make the motion unrealistic. As a result, the outputs of this network fail to meet the requirements for being a viable solution to the motion in-betweening task.

Table 6: Comparison of only generating the joint rotations ($\hat{\mathbf{Q}}$), while the root position (X) is taken from the reference samples. A lower score indicates better performance.

Metric	Gap size	$\hat{\mathbf{Q}}_{cond}$ and X	$\hat{\mathbf{Q}}_{LSTM}$ and X	$\hat{\mathbf{Q}}_{UNet}$ and X
L2Q	5	**0.332**	3.852	0.824
	10	**0.505**	3.195	2.006
	15	**0.523**	1.387	1.002
L2P	5	**0.346**	5.188	1.073
	10	**0.596**	4.430	1.822
	15	**0.612**	1.912	1.117
NPSS	5	**0.033**	3.135	0.091
	10	**0.085**	2.636	0.877
	15	**0.120**	0.407	0.309

Next, we tested how a UNet with 1D convolutional layers performed. The architecture used here was the same as the one described in Section 3.3, but without incorporating diffusion models. Quantitatively, the metrics were approximately one unit worse than those reported in Section 5. Qualitatively, the results were more realistic than those from the LSTM but still exhibited more artifacts compared to the outputs of the diffusion model. While the UNet generated plausible motions, the outputs often contained sudden, erratic changes that were not present in the diffusion model's results. These abrupt changes are likely the main reason for the lower scores in the objective metrics.

The comparison between the different architectures are presented in Table 6.

6.5 Conditioning the rotations on the generated root positions

One of the limitations of the model, as discussed in Section 5, was the mismatch in the generated motion due to generating the root positions and joint rotations separately. One potential solution to address this problem involves a two-step approach. First, the root positions would be generated using the same model described in Section 4. The output from this model would then be fed into the "$\hat{\mathbf{Q}}_{cond}$ and $\mathbf{X}$" model, predicting the joint rotations conditioned on the generated root positions.

However, this modification resulted in slightly worse performance than the "$\hat{\mathbf{Q}}$ and $\hat{\mathbf{X}}$" model, with the L2P and L2Q metrics increasing by a range of 0.005 to 0.05. This suggests that this approach is not the best way to integrate information from the root positions and joint rotations. It would be beneficial to repeat these experiments with alternative architectures to determine whether this behavior persists. Another possible avenue for improvement would be to train both models simultaneously to encourage better coordination between the root position and joint rotation predictions.

7 LIMITATIONS AND FUTURE WORK

While our model generates plausible motions to fill in the gaps between keyframes, offering a valid solution to the motion inbetweening problem, it still has some limitations that could be addressed to enhance the quality of the results. One key limitation is that the

model does not connect the root position and joint rotations during training. The lack of coordination between these two aspects can lead to artifacts, such as characters appearing to float slightly below the floor. We believe that integrating the root position with joint rotations could improve the stability and naturalness of the generated motions. A potential solution would be to train the models for root position and joint rotations simultaneously, allowing the model to learn their interdependencies more effectively, leading to cleaner and more consistent results.

Another limitation of the current model is its inability to generate more than 15 adjacent frames. Expanding this capability to produce longer sequences would be particularly valuable, especially when only a few surrounding reference frames are available. Additionally, the model's performance varies depending on the size of the gap, and we aim to improve its robustness to ensure consistent performance, whether the gap is small or large. To address this, we plan to explore architectural modifications, such as integrating models designed for long-range temporal dependencies, like transformers or dilated convolutions. These models could help the network capture broader temporal context while preserving the finer details of local motion, ultimately improving performance across different gap sizes.

In future work, we would like to explore the impact of different input representations on model performance. One potential improvement is to shift from using global data to using local data relative to the last known frame before the gap. This approach may improve training stability by reducing the complexity of learning over long sequences. Additionally, we are interested in exploring other techniques to further enhance the model's capabilities. For instance, we could investigate the approach proposed by Oreshkin et al. [6], where the model predicts the difference between linear interpolation and the reference frame.

Lastly, we plan to experiment with generating sparse keyframes rather than generating a continuous set of frames concentrated in the middle of the sequence. This approach could allow for more flexibility in the animation process and potentially lead to more efficient motion generation by focusing on the most critical moments of the motion.

8 CONCLUSION AND SELF ASSESSMENT

We have successfully implemented a motion in-betweening model from scratch, outperforming the interpolation baseline for a gap size of 15 frames across various action types. Our model generates realistic motion that is coherent with the surrounding known frames. Additionally, we have explored the strengths and weaknesses of applying different architectures and examined alternatives to improve the model's performance. We also conducted both quantitative and qualitative evaluations of the results, analyzing the outputs in detail.

This project has provided me with valuable insights into both the theoretical and practical aspects of diffusion models, as well as the challenges of working with data with information in the spatio-temporal domain. Furthermore, it has taught me how to structure and manage a large project from scratch, understanding the necessary steps from start to finish.

REFERENCES

[1] Simon Alexanderson, Rajmund Nagy, Jonas Beskow, and Gustav Eje Henter. 2023. Listen, denoise, action! audio-driven motion synthesis with diffusion models. *ACM Transactions on Graphics (TOG)* 42, 4 (2023), 1–20.

[2] Tenglong Ao, Zeyi Zhang, and Libin Liu. 2023. Gesturediffuclip: Gesture diffusion model with clip latents. *ACM Transactions on Graphics (TOG)* 42, 4 (2023), 1–18.

[3] Setareh Cohan, Guy Tevet, Daniele Reda, Xue Bin Peng, and Michiel van de Panne. 2024. Flexible motion in-betweening with diffusion models. In *ACM SIGGRAPH 2024 Conference Papers*. 1–9.

[4] Anand Gopalakrishnan, Ankur Mali, Dan Kifer, Lee Giles, and Alexander G Ororbia. 2019. A neural temporal model for human motion prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 12116–12125.

[5] Félix G Harvey, Mike Yurick, Derek Nowrouzezahrai, and Christopher Pal. 2020. Robust motion in-betweening. *ACM Transactions on Graphics (TOG)* 39, 4 (2020), 60–1.

[6] Boris N Oreshkin, Antonios Valkanas, Félix G Harvey, Louis-Simon Ménard, Florent Bocquelet, and Mark J Coates. 2023. Motion In-Betweening via Deep Δ-Interpolator. *IEEE Transactions on Visualization and Computer Graphics* (2023).

[7] Paul Starke, Sebastian Starke, Taku Komura, and Frank Steinicke. 2023. Motion in-betweening with phase manifolds. *Proceedings of the ACM on Computer Graphics and Interactive Techniques* 6, 3 (2023), 1–17.

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

[9] Guy Tevet, Sigal Raab, Brian Gordon, Yoni Shafir, Daniel Cohen-or, and Amit Haim Bermano. 2023. Human Motion Diffusion Model. In *The Eleventh International Conference on Learning Representations*. https://openreview.net/forum?id=SJ1kSyO2jwu

[10] Yi Zhou, Connelly Barnes, Jingwan Lu, Jimei Yang, and Hao Li. 2019. On the continuity of rotation representations in neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 5745–5753.