

Motion In-betweening via Two-stage Transformers

JIA QIN, State Key Lab of CAD&CG, Zhejiang University and Netease Fuxi AI Lab

YOUYI ZHENG*, State Key Lab of CAD&CG, Zhejiang University

KUN ZHOU*, State Key Lab of CAD&CG, Zhejiang University



Fig. 1. Three motion transitions generated by the proposed system. For simplicity, the poses are rendered every five frames. The opaque poses are the keyframes, while the semi-transparent poses are the generated transitions and context frames.

We present a deep learning-based framework to synthesize motion in-betweening in a two-stage manner. Given some context frames and a target frame, the system can generate plausible transitions with variable lengths in a non-autoregressive fashion. The framework consists of two Transformer Encoder-based networks operating in two stages: in the first stage a Context Transformer is designed to generate rough transitions based on the context and in the second stage a Detail Transformer is employed to refine motion details. Compared to existing Transformer-based methods which either use a complete Transformer Encoder-Decoder architecture or additional 1D convolutions to generate motion transitions, our framework achieves superior performance with less trainable parameters by only leveraging the Transformer Encoder and masked self-attention mechanism. To enhance the generalization of our transformer-based framework, we further introduce Keyframe Positional Encoding and Learned Relative Positional Encoding to make our method robust in synthesizing longer transitions exceeding the maximum transition length during training. Our framework is also artist-friendly by supporting full and partial pose constraints within the transition, giving artists fine control over the synthesized results. We benchmark our framework on the LAFAN1 dataset, and experiments show that our method outperforms the current state-of-the-art methods at a large margin (an

*corresponding authors

Authors' addresses: Jia Qin, State Key Lab of CAD&CG, Zhejiang University, Hangzhou, 310058, qinjia@zju.edu.cn, Netease Fuxi AI Lab; Youyi Zheng, State Key Lab of CAD&CG, Zhejiang University, Hangzhou, youyizheng@zju.edu.cn; Kun Zhou, State Key Lab of CAD&CG, Zhejiang University, Hangzhou, kunzhou@acm.org.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

© 2022 Copyright held by the owner/author(s). Publication rights licensed to ACM. 0730-0301/2022/12-ART184 \$15.00
<https://doi.org/10.1145/3550454.3555454>

average of 16% for normal-length sequences and 55% for excessive-length sequences). Our method trains faster than the RNN-based method and achieves a four-time speedup during inference. We implement our framework into a production-ready tool inside an animation authoring software and conduct a pilot study to validate the practical value of our method.

CCS Concepts: • Computing methodologies → Motion capture; Neural networks.

Additional Key Words and Phrases: animation, transition generation, motion synthesis, deep learning, transformer

ACM Reference Format:

Jia Qin, Youyi Zheng, and Kun Zhou. 2022. Motion In-betweening via Two-stage Transformers. *ACM Trans. Graph.* 41, 6, Article 184 (December 2022), 16 pages. <https://doi.org/10.1145/3550454.3555454>

1 INTRODUCTION

Traditional handcrafted animation often heavily relies on creating keyframes while the in-betweening is automatically generated through spline-based interpolation. Animators have to manually adjust spline tangents and add extra keyframes to get the desired transition. Despite the recently proposed method [Ciccone et al. 2019] can simplify the tweaking of spline tangents through optimization, adding extra keyframes is still necessary to enhance the details of the interpolated transition, making it still time-consuming for animators. In contrast, deep learning-based methods [Duan et al. 2021; Harvey et al. 2020] can generate more natural transitions conditioned on keyframes and tackle the scalability issue that traditional data-driven methods [Chai and Hodgins 2007; Lehrmann et al. 2014; Wang et al. 2007] have.

Pioneer learning-based work [Harvey and Pal 2018; Harvey et al. 2020] use an augmented LSTM model to synthesize each transition frame based on its previous frame, target frame, and offset

from the target. However, the training and inference of RNN-based methods are relatively slow as they process each frame sequentially on the temporal dimension. CNN-based methods such as [Hernandez et al. 2019; Kaufmann et al. 2020; Zhou et al. 2020] use a fully-convolutional autoencoder architecture to fill missing frames between keyframes. While they leverage the parallel processing power of modern hardware, their ability to model long-term dependencies between frames is limited by the sizes of their receptive fields. Besides, the input size of these networks is often powers of two so that the output size remains the same. Paddings must be added to handle variable-length input, either wasting computation or making the output quality unstable. Compared with CNN and RNN-based methods, Transformer’s global attention is more suitable for modeling long-term dependencies and thus performs better when generating longer transitions [Duan et al. 2021; Kim et al. 2022; Oreshkin et al. 2022].

In this paper, we introduce a framework that synthesizes variable-length motion in-betweening in a two-stage manner through two Transformer Encoder-based networks: a Context Transformer for generating rough transitions based on the context (stage I) and a Detail Transformer for refining motion details (stage II). Compared to existing Transformer-based methods which either use a complete Transformer Encoder-Decoder architecture [Oreshkin et al. 2022] or a Transformer Encoder with additional 1D convolution [Duan et al. 2021] to generate motion in-betweenings, our framework can achieve better performance with less trainable parameters by only leveraging the Transformer Encoder and masked self-attention mechanism. To enhance the generalization of our transformer-based framework when dealing with longer transitions, we further introduce Keyframe Positional Encoding and Learned Relative Positional Encoding, making our method more robust in synthesizing longer transitions exceeding the maximum transition length seen during training.

To make our framework artist-friendly, our system supports full and partial pose constraints within the transition, giving artists fine control over the synthesized results. We implemented our method into a production-ready tool inside an animation authoring software to generate high-quality in-between motions. Our framework is benchmarked against the LAFAN1 dataset, and the results show that our framework outperforms existing RNN, CNN, and Transformer-based methods at a large margin. Meanwhile, our method trains faster than the RNN-based method and achieves a four-time speedup during inference. We also integrate our method into a production-ready tool inside an animation authoring software and conduct a small-scale user study to show the practical value of our tool.

In summary, our contributions are: 1) We demonstrate the effectiveness of a Transformer Encoder-based framework that robustly synthesizes variable length in-betweenings in a two-stage manner with our proposed Keyframe Positional Encoding and Learned Relative Positional encoding. 2) We report state-of-the-art motion in-between benchmark results on LAFAN1, outperforming existing RNN, CNN and Transformer-based methods.

2 RELATED WORK

2.1 Human Motion Synthesis

Human motion synthesis algorithms have been long-investigated in computer graphics. Many classical methods fall into the category of physics-based approaches [Faloutsos et al. 2001; Fang and Pollard 2003; Hodgins et al. 1995] where motion is generated following physical laws, and data-driven approaches where motion is synthesized based on some existing datasets. Two types of data-driven approaches are related to this work: motion control and motion prediction.

Motion control is a conditional motion synthesis task based on high-level control signals. Graph-based methods [Arikan and Forsyth 2002; Beaudoin et al. 2008; Kovar et al. 2002; Lee et al. 2002; Min and Chai 2012; Safanova and Hodgins 2007] are widely adopted where motion graphs are built by inserting transitions at similar frames in a large unstructured motion capture dataset. The algorithms traverse the graph at runtime to produce motions satisfying certain goals. [Lee et al. 2010] extend the graph structure into a continuous vector field. Motion matching [Büttner and Clavet 2015] is a brute-force technique that searches for the best matching segment in a large dataset to match the current character pose and trajectory. As motion graph construction is often hard to control and maintain, some statistical methods are proposed to make the animation data easier to manage, such as linear methods leveraging the Principal Component Analysis [Chai and Hodgins 2005; Tautges et al. 2011], and kernel-based methods using Radial-basis functions and Gaussian Process [Grochow et al. 2004; Kovar and Gleicher 2004; Mukai and Kuriyama 2005; Park et al. 2002; Rose et al. 1998; Wang et al. 2007]. However, all these methods suffer from scalability issues when applied to a vast dataset. Recent neural-network-based methods tackle this problem and provide a fixed computational and memory cost at runtime. [Holden et al. 2016, 2015] presented a motion synthesis framework based on high-level parameters using a Convolutional Neural Network. For synthesizing motions with real-time control, phase-aware [Holden et al. 2017] and mode-aware [Zhang et al. 2018] networks are proposed for humanoid and quadruped characters. They use gating networks to mix the weights of expert networks, producing smooth transitions between different types of motion. [Starke et al. 2019, 2020] further extend these methods to support non-periodic motions and motion interactions. Reinforcement learning based methods such as [Ling et al. 2020; Wang et al. 2020] are also being investigated. In addition, some neural network-based methods are introduced to improve existing game animation workflows, such as learned motion matching [Holden et al. 2020] and neural animation layering [Starke et al. 2021].

Motion prediction is another type of conditional motion synthesis problem where new motions are produced based on some past seed motions. Early exploration such as [Taylor et al. 2006] uses Conditional Restricted Boltzmann Machines to predict the next frame conditioned on the current hidden state and previous frames. Recently, methods based on the Recurrent Neural Network (RNN) have dominated this area. [Fragkiadaki et al. 2015] proposes the Encoder-Recurrent-Decoder (ERD) network, which incorporates nonlinear encoder and decoder networks before and after recurrent layers.

[Jain et al. 2016] combines spatio-temporal graphs with RNN to model human motion and conduct predictions. [Martinez et al. 2017] uses a single-layer residual recurrent network predicting offset from the current pose. Other RNN-based methods [Barsoum et al. 2018; Chiu et al. 2019; Gopalakrishnan et al. 2019; Gui et al. 2018; Pavlo et al. 2018] investigate different architecture, data representation and loss functions. Besides RNN, other network architectures are also being explored: Convolutional Neural Network (CNN) [Liu et al. 2020; Pavlo et al. 2020], where convolution filters are applied along temporal dimension; Graph Convolution Network (GCN) [Dang et al. 2021], which better model the movement relationship among joints along skeleton hierarchy; Transformers [Aksan et al. 2021; Cai et al. 2020], which can directly capture the long-range temporal dependencies within a motion sequence.

2.2 Motion In-betweening

Motion in-betweening is derived from the motion prediction problem, where the resulting motion is constrained on both past and future frames. The most trivial solution is to interpolate keyframes based on splines such as Bezier curves, widely adopted by animation authoring software [Autodesk 2022; Blender Foundation 2022]. While it gives animators fine control over the transitions, tweaking result is often very labor-intensive. [Ciccone et al. 2019] proposes a method to optimize the tangents of interpolation curves based on easily controlled trajectories. However, keyframe interpolation can only create smooth transitions without much motion details. [Ngo and Marks 1993; Rose et al. 1996; Witkin and Kass 1988] use a physically-based approach to produce motion between keyframes by solving an optimization problem with spacetime constraints. Statistical models are later adopted in transition synthesis such as Maximum A Posteriori (MAP) [Chai and Hodgins 2007; Min et al. 2009], Gaussian Process [Wang et al. 2007] and Markov models [Lehrmann et al. 2014].

As neural network-based methods grow popular, [Zhang and van de Panne 2018] use an RNN to produce 2D jump animation conditioned on some keyframes. [Harvey and Pal 2018] present Recurrent Transition Networks (RTN) to autoregressively generate transitions through an augmented LSTM model based on previous frames, target frame, and offset from the target. [Harvey et al. 2020] improves upon RTN by introducing a time-to-arrival embedding to produce a smooth transition to the target frame and a scheduled target noise vector to diversify the results. [Geleijn et al. 2021] proposes a light-weighted network reducing arithmetic operations while producing decent results. [Li et al. 2021] use a hierarchical VAE for motion in-betweening and achieve results comparable to Slerp baseline. [Tang et al. 2022] employ CVAE and RNN to produce real-time motion transitions. CNN-based methods such as [Hernandez et al. 2019; Kaufmann et al. 2020; Zhou et al. 2020] use a fully-convolutional autoencoder architecture to fill not only entire missing frames but also partial poses. To address CNN’s limitation on capturing long-range dependencies of distant features, [Cai et al. 2021] introduce a Conditional Variational Autoencoder (CVAE) with a cross-attention mechanism as a uniformed framework to handle different 3D motion synthesis tasks.

For Transformer-based methods, [Duan et al. 2021] use BERT [Devlin et al. 2018] encoder and 1D convolution to generate in-betweenings in a non-autoregressive manner. [Petrovich et al. 2021] uses a hierarchical RNN module as the backbone in combination with two Transformer controllers to synthesize dance motion based on keyframes. [Kim et al. 2022] uses a single Transformer encoder and supports pose constraints. Our method differs from [Duan et al. 2021] and [Kim et al. 2022] as we adopt a two-stage approach, and can better generalize to longer in-betweenings by leveraging our proposed Keyframe Positional Encoding and Learned Relative Positional Encoding.

A concurrent work [Oreshkin et al. 2022] uses a Transformer-based Encoder-Decoder structure to synthesize motion in-betweening. Its output is based on the sum of spherical linear interpolation (Slerp) between keyframes and the delta motion predicted by their model. In contrast, our method directly predicts the output motion. We argue that our method performs better as Slerp can produce unnatural transitions and cannot be corrected by the delta motion predicted by their model. We show some examples of such artifacts in Section 4.3. Moreover, we demonstrate that our method is more robust in generating long transitions in Section 4.2.

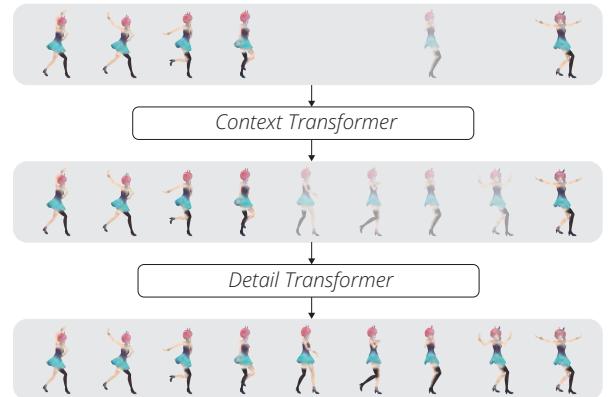


Fig. 2. An overview of our system. Our system accepts context frames, a target frame, and some optional pose constraints as the input. The motion transition is first roughly produced by the Context Transformer and then further refined by the Detail Transformer.

3 METHODOLOGY

3.1 Overview

The objective of our system is to generate plausible in-betweening animation based on several context frames and a target frame. Although generating arbitrary in-betweenings is possible, we focus on generating animations for humanoid skeletons in this work. In order to provide fine control over the synthesized result, animators can specify full or partial pose constraints within the transition. Especially, the partial pose constraints can be very beneficial when the trajectory of an articulated character is predetermined (i.e., position and rotation of root joint are constrained), while the motions of other joints are expected to be generated. Figure 2 shows the overview of our system. From a high-level perspective, our pipeline

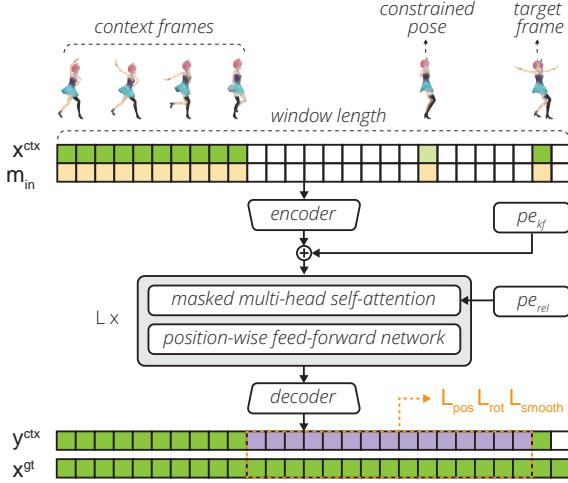


Fig. 3. The Context Transformer network. Ground truth poses are in green. Constrained poses are in light green. The input mask indicates which poses should be kept fully or partially consistent between input and output. The masked frames are in light yellow. Context Transformer's output transition is in purple.

consists of two parts: a Context Transformer network \mathcal{T}_{ctx} and a Detail Transformer network \mathcal{T}_{det} . The Context Transformer network focuses on generating rough transition animation based on the "context" (context frames and target frame). In contrast, the Detail Transformer network refines the output from the Context Transformer. This two-stage approach is based on our empirical observation that using a single network tends to smooth out the generated transition. Adding a second step would enhance motion details and fixes artifacts such as foot sliding.

3.2 Data Representation

In this paper, we use $\mathbf{x} = \{x_0, \dots, x_{t_{ctx}}, \dots, x_{t_{tgt}}, \dots, x_{T-1}\}$ to represent a motion clip with window length T . t_{ctx} is the length of context frames and t_{tgt} is the time index of target frame. Given the context frames $x_{0:t_{ctx}}$ and target frame $x_{t_{ctx}:t_{tgt}}$, our goal is to generate plausible transition $x_{t_{ctx}:t_{tgt}}$. At each frame t , the pose of a humanoid skeleton with J joints is represented by local rotations of all joints $r_t \in \mathbb{R}^{J \times 6}$ and the world position of root joint $p_t \in \mathbb{R}^3$. The local position of each joint relative to its parent is constant, ensuring consistent bone length across frames. We adopt the 6D rotation representation introduced in [Zhou et al. 2019] for r_t . This 6D representation can be viewed as simply taking the first two rows or columns of a 3×3 rotation matrix and can be converted back to a 3×3 rotation matrix through Gram–Schmidt orthogonalization. Compared with many previous works [Harvey et al. 2020; Villegas et al. 2018] that use quaternions as rotation representation, the 6D rotation ensures the uniqueness and continuity of the representation, which is beneficial to the training of deep neural networks. We flatten r_t and p_t into a 1D vector and concatenate them to form \tilde{x}_t . Then apply z-score normalization to get $x_t = (\tilde{x}_t - \mu_x)/\sigma_x$. Finally we get the motion clip $\mathbf{x} \in \mathbb{R}^{T \times D}$, where $D = J \times 6 + 3$.

3.3 Context Transformer

Figure 3 shows the overview of the Context Transformer network \mathcal{T}_{ctx} . It is an encoder-decoder structure with several Transformer layers in the middle. \mathbf{x}^{ctx} is the input motion clip. For missing values in \mathbf{x}^{ctx} , we use linear interpolation to fill the root positions while local joint rotations are set to zero. Between context frames and target frame, there can be arbitrary numbers of constrained poses. These poses can be either fully constrained or partially constrained. For indicating which input values are valid, a binary mask vector is concatenated with \mathbf{x}^{ctx} before inputting into the network. The input mask can be either $\mathbf{m}_{in} \in \mathbb{R}^{T \times 1}$ or $\mathbf{m}_{in} \in \mathbb{R}^{T \times D}$ depending on whether poses are fully or partially constrained.

We first project the input into a d -dimension latent space vector $\tilde{\mathbf{h}}^{ctx} \in \mathbb{R}^{T \times d}$ through an encoder. The encoder \mathcal{E} is a fully-connected feed-forward network with a hidden layer and output layer of 512 units (i.e. $d = 512$):

$$\tilde{\mathbf{h}}^{ctx} = \phi(\phi([\mathbf{x}^{ctx}, \mathbf{m}_{in}] \mathbf{W}_{\mathcal{E}}^{(1)} + \mathbf{b}_{\mathcal{E}}^{(1)}) \mathbf{W}_{\mathcal{E}}^{(2)} + \mathbf{b}_{\mathcal{E}}^{(2)}) \quad (1)$$

Here $\mathbf{W}_{\mathcal{E}}^{(l)}$ and $\mathbf{b}_{\mathcal{E}}^{(l)}$ corresponds to weight matrices and bias vectors at layer l . ϕ is the Parametric Rectified Linear Unit (PReLU) activation function [He et al. 2015].

Since the Transformer does not have recurrence or convolution, we add a positional encoding to $\tilde{\mathbf{h}}^{ctx}$ so that the model can make use of the sequence order. Different from conventional sinusoidal or learned positional encoding, we introduce the Keyframe Positional Encoding $\mathbf{pe}_{kf} \in \mathbb{R}^{T \times d}$ (see Section 3.3.1), representing the position relative to the last context frame and the target frame:

$$\mathbf{h}^{ctx} = \tilde{\mathbf{h}}^{ctx} + \mathbf{pe}_{kf} \quad (2)$$

For the Transformer layers, we follow the encoder layers in [Vaswani et al. 2017] consisting of multi-head self-attention (MHSA) and position-wise feed-forward network (PFFN). Layer normalization [Ba et al. 2016] and residual connection [He et al. 2016] is employed on each sub-layer:

$$\tilde{\mathbf{h}}^l = \mathbf{h}^{l-1} + \text{MHSA}(\text{LayerNorm}(\mathbf{h}^{l-1})) \quad (3)$$

$$\mathbf{h}^l = \tilde{\mathbf{h}}^l + \text{PFFN}(\text{LayerNorm}(\tilde{\mathbf{h}}^l)) \quad (4)$$

where \mathbf{h}^l is the output of l -th Transformer layer, $l = 1, 2, \dots, L$ and $\mathbf{h}^0 = \mathbf{h}^{ctx}$. Note that layer normalization is applied to the input of each sub-layer (pre-norm) rather than after the residual addition (post-norm). This can improve stability and efficiency during training [Chen et al. 2018; Wang et al. 2019].

Compared with vanilla Transformer, the layers we adopt here have two main differences. First, we use masked multi-head self-attention to mask out missing frames. Only context frames, constrained frames, and target frame can attend to each other. Figure 5a shows the attention mask of Context Transformer. Second, we use a learned relative positional encoding \mathbf{pe}_{rel} different from a regular relative positional encoding (see Section 3.3.2) to represent the relative order of the frames. [Shaw et al. 2018] have demonstrated that relative positional encoding can improve performance on tasks such as machine translation. In summary, the masked multi-headed self-attention (MHSA) we adopt can be written as:

$$\text{MHSA}(\mathbf{x}) = \text{concatenate}(\text{head}_1, \text{head}_2, \dots, \text{head}_H) \mathbf{W}^o \quad (5)$$

$$\text{head}_i = \text{Attention}(\mathbf{Q}_i, \mathbf{K}_i, \mathbf{V}_i) \quad (6)$$

$$\mathbf{Q}_i = \mathbf{x}\mathbf{W}_i^Q, \mathbf{K}_i = \mathbf{x}\mathbf{W}_i^K, \mathbf{V}_i = \mathbf{x}\mathbf{W}_i^V \quad (7)$$

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T + \mathbf{S}_{rel}}{\sqrt{d_K}} + \mathbf{m}_{atten}\right)\mathbf{V} \quad (8)$$

d_K is the dimension of the keys. \mathbf{m}_{atten} is the attention mask. \mathbf{S}_{rel} is the matrix introduced in [Dai et al. 2019], which is an efficient way to inject relative positional information into the attention.

Finally, the output of Transformer layers is passed to the 2-layer decoder with hidden layer size of 512. $\mathbf{W}_{\mathcal{D}}^{(l)}$ and $\mathbf{b}_{\mathcal{D}}^{(l)}$ are decoder's weight matrices and bias vectors at layer l :

$$\mathbf{y}^{ctx} = \phi(\mathbf{h}^L \mathbf{W}_{\mathcal{D}}^{(1)} + \mathbf{b}_{\mathcal{D}}^{(1)}) \mathbf{W}_{\mathcal{D}}^{(2)} + \mathbf{b}_{\mathcal{D}}^{(2)} \quad (9)$$

We can then extract the predicted joint rotation $\hat{\mathbf{r}}^{ctx}$ and root joint position $\hat{\mathbf{p}}^{ctx}$ from \mathbf{y}^{ctx} :

$$\hat{\mathbf{r}}^{ctx} = \mathcal{R}(\mathbf{y}^{ctx}), \hat{\mathbf{p}}^{ctx} = \mathcal{P}(\mathbf{y}^{ctx}) \quad (10)$$

where \mathcal{R} and \mathcal{P} are functions to extract joint rotation and root position from \mathbf{y}^{ctx} respectively.

3.3.1 Keyframe Positional Encoding. In vanilla Transformer [Vaswani et al. 2017], a positional encoding based on different frequencies of sine and cosine functions is added to the input, representing the absolute position of each item within the sequence. For transition generation tasks, an interesting observation is that a missing frame is more likely to resemble the known frames when they are close on the temporal dimension and less correlated when they are far apart. Therefore, the position of any missing frame relative to the known frames is crucial for generating plausible in-betweenings. Unfortunately, it cannot be directly inferred from the absolute positional encoding used in vanilla Transformer. One possible solution is to use the concatenation of two absolute positional encoding starting from both ends of the transition. However, in our early experiments, the quality of generated transition quickly deteriorates when the transition length exceeding the maximum training length by a few frames. Moreover, the ideal positional encoding should only relates to the relative distance to the known frames. We would like to obtain it through learning. We do not adopt the conventional learning approach [Gehring et al. 2017] through learning a fixed-size lookup table of positional encodings. The size of lookup table would limit the length of generated transition during inference. Instead, our proposed Keyframe Positional Encoding uses a multi-layer perceptron (MLP) with a single hidden layer of 512 units to learn a mapping between relative position \mathbf{p}_{kf} and the Keyframe Positional Encoding \mathbf{pe}_{kf} :

$$\mathbf{pe}_{kf} = \phi(\mathbf{p}_{kf} \mathbf{W}_{kf}^{(1)} + \mathbf{b}_{kf}^{(1)}) \mathbf{W}_{kf}^{(2)} + \mathbf{b}_{kf}^{(2)}, \quad (11)$$

where $\mathbf{p}_{kf} \in \mathbb{R}^{T \times 2}$ is an integer-valued tensor describing the position of every frame relative to both the last context frame and the target frame. In Section 4.4 we will empirically show that the Keyframe Positional Encoding can enhance our model's generalization.

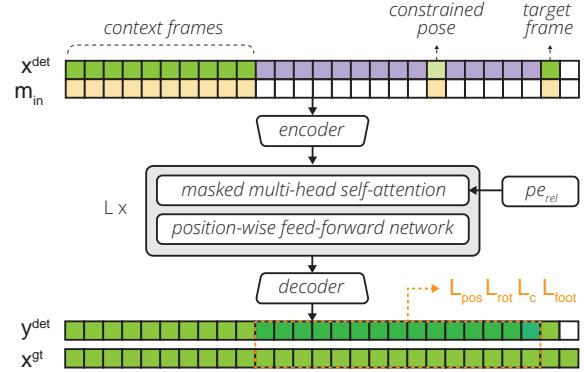


Fig. 4. The Detail Transformer network. The output transition from Context Transformer (in purple) becomes the input of Detail Transformer.

3.3.2 Learned Relative Positional Encoding. Relative positional encoding captures the pairwise distance between any two items within a sequence. It can improve Transformer's performance on sequence-to-sequence tasks such as machine translation [Shaw et al. 2018] and music generation [Huang et al. 2018]. Similarly, we adopt it in our Context and Detail Transformers to improve models' generalization to longer transitions. When generating longer transitions with absolute positional encoding, the position encoding of some frames might be unknown to the model, generating bad results at those frames. When using relative positional encoding, the relative position between any two frames is likely to have appeared during training, causing less artifacts empirically. We do not adopt the conventional approach of learning a fixed-size lookup table with relative positional encodings [Huang et al. 2018; Shaw et al. 2018] for the same reason as the \mathbf{pe}_{kf} . Similarly, our Learned Relative Positional Encoding \mathbf{pe}_{rel} uses a two-layer MLP to learn a mapping between relative distances between two frames $\mathbf{p}_{rel} \in \mathbb{R}^{(2T-1) \times 1}$ and a variable-size lookup table $\mathbf{E}_{rel} \in \mathbb{R}^{(2T-1) \times d_K}$ including all possible relative position encoding for a clip with window length T . Empirical results in Section 4.4 shows the \mathbf{pe}_{rel} can enhance the model's generalization to longer transitions. In summary, the matrix \mathbf{S}_{rel} in equation (8) can be formulated as:

$$\mathbf{S}_{rel} = \text{skew}(\mathbf{Q}\mathbf{E}_{rel}^T) \quad (12)$$

where $\text{skew}(\cdot)$ is the skewing mechanism introduced in [Huang et al. 2018] aiming to reduce memory footprint. \mathbf{E}_{rel} is the output of an MLP with 512 hidden units:

$$\mathbf{E}_{rel} = \phi(\mathbf{p}_{rel} \mathbf{W}_{rel}^{(1)} + \mathbf{b}_{rel}^{(1)}) \mathbf{W}_{rel}^{(2)} + \mathbf{b}_{rel}^{(2)} \quad (13)$$

3.4 Detail Transformer

Detail Transformer \mathcal{T}_{det} refines the output of Context Transformer. Figure 4 is the overview of Detail Transformer. The Detail Transformer has a similar network structure to the Context Transformer. As the purpose of Detail Transformer is to refine motion rather than generate brand new transition motion, the position relative to context and target frames is no longer that relevant. Thus, there is no Keyframe Positional Encoding in the Detail Transformer. Similar

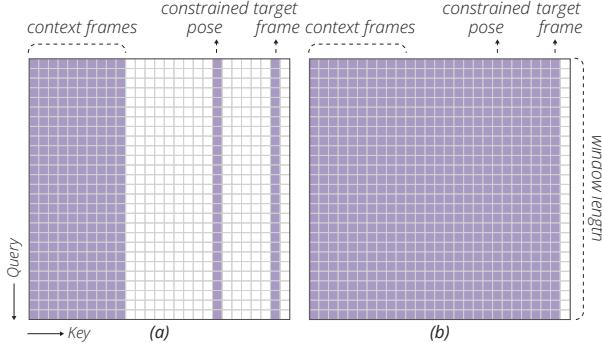


Fig. 5. The attention mask \mathbf{m}_{atten} of Context Transformer (a) and Detail Transformer (b). The purple region is where frames can attend to each other. Purple represents zero and white represents minus infinity.

to the Context Transformer, the input of Detail Transformer consists of motion clip \mathbf{x}^{det} and mask \mathbf{m}_{in} . The input mask \mathbf{m}_{in} is the same as in Context Transformer. For the input motion clip \mathbf{x}^{det} , the main difference is that its values within the transition frames are copied from \mathbf{y}^{ctx} . Note that if any constrained pose exists within the transition, we only copy values of the unconstrained joints while leaving the constrained values to ground truth. Figure 5b shows the attention mask of Detail Transformer. Since there are no placeholder values in the input motion, context frames, transition frames, and the target frame can attend to each other freely. The output of Detail Transformer \mathbf{y}^{det} includes joint rotation $\hat{\mathbf{r}}^{det}$ and root joint position $\hat{\mathbf{p}}^{det}$. We can use the same \mathcal{R} and \mathcal{P} function in equation (10) to extract them:

$$\hat{\mathbf{r}}^{det} = \mathcal{R}(\mathbf{y}^{det}), \hat{\mathbf{p}}^{det} = \mathcal{P}(\mathbf{y}^{det}) \quad (14)$$

Moreover, the Detail Transformer outputs contact information of both feet and toes, denoted by $\hat{\mathbf{c}} \in \mathbb{R}^{T \times 4}$, where ground contact is represented by one otherwise zero. A humanoid rig can utilize this contact information to eliminate foot-sliding artifacts through inverse kinematics. We use C to represent the function which extracts foot contact dimensions from Detail Transformer's output. A sigmoid nonlinearity is applied to get the final foot contact $\hat{\mathbf{c}}$:

$$\hat{\mathbf{c}} = \text{sigmoid}(C(\mathbf{y}^{det})) \quad (15)$$

3.5 Loss Functions

Our Context Transformer and Detail Transformer are trained with multiple loss terms.

3.5.1 Local Rotation Loss. We calculate L1 loss between all joint's predicted local rotations $\hat{\mathbf{r}}$ and ground truth local rotations \mathbf{r} . It's evaluated over transition frames only. The length of transition is $\tilde{T} = t_{tgt} - t_{ctx}$. The local rotation loss can be formulated as:

$$L_{rot} = \frac{1}{\tilde{T}} \sum_{t=t_{ctx}}^{t_{tgt}-1} \|\mathbf{r}_t - \hat{\mathbf{r}}_t\|_1 \quad (16)$$

3.5.2 Global Position Loss. As joints in humanoid skeleton form a hierarchical structure, the errors on joints along the hierarchy should be weighted differently. For instance, the root joint's error

would cause the predicted pose to deviate more from ground truth compared to errors on leaf joints. Therefore, we evaluate L1 loss on joints' global positions, which helps to implicitly weight joint's orientation [Pavllo et al. 2020]:

$$L_{pos} = \frac{1}{\tilde{T}} \sum_{t=t_{ctx}}^{t_{tgt}-1} \|\mathbf{g}_t - \hat{\mathbf{g}}_t\|_1, \quad \mathbf{g}_t = FK(\mathbf{p}_t, \mathbf{r}_t) \quad (17)$$

where $\mathbf{g}_t \in \mathbb{R}^{J \times 3}$ is the global position of all joints at time t . $FK(\cdot)$ represents forward kinematics for calculating global joint positions based on root position and local joint rotations.

3.5.3 Smoothness Loss. To maintain the smoothness of the generated motion, especially at both ends of the transition, we introduce the smoothness term to capture the differences of global joint positions between adjacent frames.

$$L_{smooth} = \frac{1}{\tilde{T} + 1} \sum_{t=t_{ctx}}^{t_{tgt}} \|\hat{\mathbf{g}}_t - \hat{\mathbf{g}}_{t-1}\|_1 \quad (18)$$

3.5.4 Contact Loss. The contact loss is a reconstruction loss for predicting contact between foot and ground. By using the predicted contact information $\hat{\mathbf{c}}$ and inverse kinematics, we can apply optional post-processing to fix foot sliding inside animation authoring software.

$$L_c = \frac{1}{\tilde{T}} \sum_{t=t_{ctx}}^{t_{tgt}-1} \|\mathbf{c}_t - \hat{\mathbf{c}}_t\|_1 \quad (19)$$

3.5.5 Foot Sliding Loss. This term helps to alleviate foot sliding artifact in the output motion. When a ground contact occurs, the joint velocity of toes and feet should be very close to zero. We multiply the predicted contact information with the global velocity of toes and feet and apply the L1 norm to their product:

$$L_{foot} = \frac{1}{\tilde{T}} \sum_{t=t_{ctx}}^{t_{tgt}-1} \|\hat{\mathbf{c}}_t \hat{\mathbf{v}}_t\|_1 \quad (20)$$

where $\hat{\mathbf{c}}_t, \hat{\mathbf{v}}_t \in \mathbb{R}^4$ are the contact vector and global foot velocity vector at time t . Note that when calculating foot sliding loss, we must sever the gradient back-propagation on $\hat{\mathbf{c}}$. Otherwise, optimizing the foot sliding loss will make the predicted contact information $\hat{\mathbf{c}}$ always be zero.

In summary, since the Context Transformer focus on predicting transition motion on a high-level scale, the final loss L_{ctx} for Context Transform consists of L_{rot} and L_{pos} to guarantee motion realism and L_{smooth} to ensure transition smoothness. For the Detail Transformer, besides L_{rot} , L_{pos} , the final loss L_{det} also includes L_c and L_{foot} . All these terms serves Detail Transformer's purpose of motion refinement. α and β are the coefficients:

$$L_{ctx} = \alpha_{pos} L_{pos} + \alpha_{rot} L_{rot} + \alpha_{smooth} L_{smooth} \quad (21)$$

$$L_{det} = \beta_{pos} L_{pos} + \beta_{rot} L_{rot} + \beta_c L_c + \beta_{foot} L_{foot} \quad (22)$$

3.6 Implementation Details

In our implementation of Context Transformer and Detail Transformer, the hidden layer size of both encoder and decoder is 512. The number of Transformer layers L is set to 6. For each layer, the input size is 512, and the number of attention heads H is 8. We tried

various combinations of these hyperparameters. The values above give the best result while keeping the trainable parameters low. A comparison of performance between different hyperparameters can be found in Appendix D.

3.6.1 Training. We train both Context Transformer and Detail Transformer with a mini-batch of 32 motion clips. Each clip is retrieved by sliding a window on longer clips from a motion dataset. The window offset is one frame and window length is $T = t_{ctx} + t_{max_trans} + 2$, where t_{max_trans} is the maximum transition length during training. For the extra two frames, one is reserved for the target frame when transition reaches its maximum length, and the other is for calculating discrete curvature at the target frame, further explained in the following Section 3.6.2.

To reduce the learning complexity, we preprocess each clip so that the root joint faces the X^+ axis and its XZ coordinates are zeroed out at the last context frame. This will make every transition starts with the same root XZ-position and orientation. Z-score normalization is applied to the network’s input to handle values in dramatically different ranges (joint rotation \mathbf{r} vs. root joint position \mathbf{p}). The normalization statistics μ_x and σ_x are calculated based on a sliding window of 50 frames with an offset of 20 frames.

We first train the Context Transformer and freeze its parameters after training is done. Then we train the Detail Transformer by inputting the prediction of the Context Transformer. For both transformers, we randomly sample transition length and constrained frames at each training iteration. Two hyperparameters determine the constrained frames: constraint frame ratio q and probability p . We first sample $q \times (t_{tgt} - t_{ctx})$ candidates and roll the dice for them each to decide the final constrained frames based on the probability p . p and q are set to 0.5 and 0.1 throughout our experiments. We use Adam [Kingma and Ba 2014] optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$. Following [Vaswani et al. 2017], we use Noam learning rate scheduler with warm-up step set to 8000 iterations. Normally, dropout is applied in the training of Transformer as the regularization scheme. However, from our experiments, applying dropout during the training of our Context Transformer and Detail Transformer does not improve the performance. We will discuss this later in Section 5. Throughout our experiments, the loss weights for Context Transformer are $\alpha_{pos} = 0.01$, $\alpha_{rot} = 1.0$ and $\alpha_{smooth} = 0.005$. For Detail Transformer, the loss weights are $\beta_{pos} = 0.02$, $\beta_{rot} = 1.0$, $\beta_c = 0.05$ and $\beta_{foot} = 0.05$. We train the models for maximum 200 epochs with early stopping to prevent overfitting.

3.6.2 Post-processing. For the output of the Context Transformer, all but transition frames are set back to the ground truth before being further processed by the Detail Transformer. Despite having the L_{smooth} term in training loss to reduce motion discontinuity, the reset to ground truth may still introduce small gaps at both transition ends. Therefore, we introduce a post-processing $\psi(\mathbf{y}, \boldsymbol{\delta}^*)$ to alleviate this artifact:

$$\psi(\mathbf{y}, \boldsymbol{\delta}) = \begin{cases} \mathbf{y} + \boldsymbol{\delta}, & t \in [t_{ctx}, t_{tgt}] \\ \mathbf{x}, & \text{otherwise} \end{cases} \quad (23)$$

$$\boldsymbol{\delta}^* = \arg \min_{\boldsymbol{\delta} \in \mathbb{R}^D} \|\kappa(\psi(\mathbf{y}, \boldsymbol{\delta}), t_{ctx} - 1) + \kappa(\psi(\mathbf{y}, \boldsymbol{\delta}), t_{tgt})\|_2$$

where $\boldsymbol{\delta}^* \in \mathbb{R}^D$ is a per-channel offset added to the entire transition, so that the curvature of motion curves are minimized at the last context frame and the target frame. \mathbf{x} is the ground truth corresponding to the predicted transition. $\kappa(\mathbf{x}, t)$ is the curvature of \mathbf{x} at frame t and is approximated by the angle between adjacent line segments formed by the neighboring keyframes in our implementation. Note that this post-processing is only applied during inference of the Context Transformer. Besides, the extra frame for calculating $\kappa(\mathbf{x}, t)$ at the target frame is not manually input by the user. It can be obtained by interpolating or extrapolating existing keyframes.

4 EXPERIMENTS

4.1 Dataset and Metrics

We conduct our experiments on the LAFAN1 dataset [Harvey et al. 2020] from Ubisoft. LAFAN1 contains 496,672 motion frames sampled at 30Hz captured in a production-grade motion capture system. It contains actions performed by five subjects covering various motion types such as walking, running, climbing, dancing, etc. The dataset deliberately excluded random short motions, which are extremely hard for predicting the missing in-betweening based on ambiguous contexts. This makes LAFAN1 a dataset well-suited for transition generation tasks.

In order to precisely evaluate the performance of different methods, we adopt the transition benchmark introduced in [Harvey et al. 2020]. The training set is extracted on Subject 1-4, with window length and offset set to 42 frames and 1 frame, resulting in 404195 clips. The test set contains 2232 clips sampled with a window of 65, offset by 40 frames on Subject 5. All models are trained with context length $t_{ctx} = 10$ frames, maximum transition $t_{max_trans} = 30$ frames (one second) and evaluated on 5, 15, 30 and 45 frames. Three metrics are evaluated: L2P, L2Q, and NPSS. The L2P and L2Q measure the average L2 distance of the global joint position and rotation (in quaternion) per joint per frame. The Normalized Power Spectrum Similarity (NPSS), proposed by [Gopalakrishnan et al. 2019], evaluates angular differences between predicted motion and ground truth on the frequency domain. The mean and standard deviation of rotation and position representations are extracted to make these metrics comparable between datasets with different angle and distance units. The motion data is normalized before calculating the benchmark metrics.

4.2 Quantitative Comparison With Previous Work

We compare our work with [Harvey et al. 2020] (ERD-QV), [Duan et al. 2021] (MC-Trans) and [Oreshkin et al. 2022] (Δ -Interp) which report the transition benchmark results on the LAFAN1 dataset and share the same experiment settings. The ERD-QV is an Encoder-Recurrent-Decoder network with quaternion and root joint velocity as input. The MC-Trans is a recently proposed motion completion method that leverages 1D convolution and Transformer network architecture to generate transition. The Δ -Interp is a concurrent work that adopts Transformer Encoder-Decoder architecture to predict the delta motion on top of the linear interpolated motion between keyframes. Besides RNN and Transformer based methods, we also compare our work with two baselines. One is a naive implementation using keyframe linear interpolation, where root

Table 1. Transition benchmark results on LAFAN1 dataset. All models are trained with maximum transition length of 30 frames. The Δ denotes result deterioration compared with our method presented in the last row.

Lengths (frames)	L2P				L2Q				NPSS			
	5	15	30	45	5	15	30	45	5	15	30	45
Interp.	0.37 Δ 270%	1.25 Δ 221%	2.32 Δ 161%	3.45 Δ 105%	0.22 Δ 120%	0.62 Δ 121%	0.98 Δ 81%	1.25 Δ 44%	0.0023 Δ 109%	0.0391 Δ 108%	0.2013 Δ 79%	0.4493 Δ 40%
FCN	0.25 Δ 150%	0.69 Δ 77%	1.63 Δ 83%	4.69 Δ 179%	0.20 Δ 100%	0.44 Δ 57%	0.77 Δ 43%	1.39 Δ 60%	0.0030 Δ 173%	0.0280 Δ 49%	0.1524 Δ 36%	0.8571 Δ 166%
MC-Trans ^a	0.23 Δ 130%	0.74 Δ 90%	1.37 Δ 54%	—	0.17 Δ 70%	0.44 Δ 57%	0.71 Δ 31%	—	0.0019 Δ 73%	0.0291 Δ 55%	0.1430 Δ 27%	—
ERD-QV	0.23 Δ 130%	0.65 Δ 67%	1.28 Δ 44%	2.24 Δ 33%	0.17 Δ 70%	0.42 Δ 50%	0.69 Δ 28%	0.94 Δ 8%	0.0020 Δ 82%	0.0258 Δ 37%	0.1328 Δ 18%	0.3311 Δ 3%
Δ -Interp	0.13 Δ 30%	0.47 Δ 21%	1.00 Δ 12%	3.23 Δ 92%	0.11 Δ 10%	0.32 Δ 14%	0.57 Δ 6%	1.15 Δ 32%	0.0014 Δ 27%	0.0217 Δ 15%	0.1217 Δ 8%	0.4539 Δ 41%
Ours (\mathcal{T}_{ctx} only) ^b	0.17 Δ 70%	0.49 Δ 26%	1.07 Δ 20%	2.00 Δ 19%	0.13 Δ 30%	0.33 Δ 18%	0.60 Δ 11%	0.92 Δ 6%	0.0015 Δ 36%	0.0212 Δ 13%	0.1238 Δ 10%	0.3369 Δ 5%
Ours (\mathcal{T}_{ctx} only) ^c	0.13 Δ 30%	0.46 Δ 18%	1.04 Δ 17%	1.94 Δ 15%	0.11 Δ 10%	0.32 Δ 14%	0.59 Δ 9%	0.91 Δ 5%	0.0015 Δ 36%	0.0211 Δ 12%	0.1210 Δ 8%	0.3349 Δ 4%
Ours	0.13 Δ 30%	0.45 Δ 15%	1.01 Δ 13%	1.88 Δ 12%	0.12 Δ 20%	0.32 Δ 14%	0.59 Δ 9%	0.90 Δ 3%	0.0015 Δ 36%	0.0209 Δ 11%	0.1209 Δ 8%	0.3365 Δ 5%
Ours	0.10	0.39	0.89	1.68	0.10	0.28	0.54	0.87	0.0011	0.0188	0.1124	0.3217

^a We use MC-Trans’s local mode for comparison, as it shares the same input and output data format (represented in joint’s local space) as our method.

^b These are results after applying the post-processing mentioned in Section 3.6.2.

^c The \mathcal{T}_{ctx} here is trained with all loss terms in Section 3.5 except for the L_{smooth} . The results are evaluated after applying the post-processing.

joint’s position is linearly interpolated, and all joints’ quaternion rotations are spherically interpolated (Slerp). The second baseline is implemented through Fully-Convolutional Network (FCN). As FCN is widely used in image inpainting tasks, we would like to explore its ability of motion infilling and compare it with our Transformer-based method. The implementation details of the FCN baseline can be found in Appendix A. These methods are then compared with the simplified (\mathcal{T}_{ctx} only) and full versions of our method. The experiment results can be found in Table 1. The Context Transformer and Detail Transformer that produces these statistics were trained for 181000 and 1430000 iterations, respectively.

Compared with the baselines and previous methods (MC-Trans and ERD-QV), our method demonstrate improved performance by a large margin on all benchmark indicators. To demonstrate the effectiveness of our two-stage design, we also provide the result of a Context Transformer (the second last row in Table 1) trained with all loss terms shown in Section 3.5 except for the L_{smooth} . The results show that a single stage with all necessary loss terms cannot achieve results comparable to a two-stage design. Compared with the concurrent work Δ -Interp, our method also shows better results on all metrics. Especially, it performs significantly better on longer transitions with 45 frames (exceeding the maximum transition length during training). Our method is more robust in generating longer transitions than Δ -Interp (see the bottom-right graph in Figure 10). Note that our method uses much fewer training parameters than Δ -Interp. The Δ -Interp shown in Table 1 has 40.4 million trainable parameters, while our Context Transformer and Detail Transformer have 10.4 million and 10.1 million trainable parameters, respectively. A per-clip comparison between ERD-QV, Δ -Interp and our method on LAFAN1’s transition benchmark can be found in Appendix B.

4.3 Qualitative Results

Figure 6 shows the qualitative results of FCN baseline, ERD-QV, our method, and their corresponding ground truth. To be consistent with our method, our implementation of ERD-QV uses 6D rotation representation instead of quaternions as the input.

FCN baseline can produce moderate results on shorter transitions. However, FCN’s result is sensitive to the length of transition. When exceeding the maximum transition length seen in the training, the generated motion starts to lose details (see the first row of the last two columns in Figure 6). Compared with ERD-QV, the transitions generated by our method have a better distribution of joint velocity along the time dimension. We observe some sudden changes in joint position and rotation (especially the root joint) in the transition generated by ERD-QV. Empirically, when an output frame is far from the target frame, the ERD-QV generates motion based on past context, similar to motion prediction (motion prediction stage). Its output is less related to the target frame. When approaching the target frame, ERD-QV starts to transition to the target frame (motion transition stage). Since ERD-QV generates motion in an autoregressive manner, it outputs one frame at each time step, generating motion at a local level. Lack of global planning of motion can result in abrupt changes between the two stages, thus causing the artifacts we mentioned. In contrast, our method has the following characteristics: 1) We adopt a two-stage approach to generate motion transition by our Context and Detail Transformer. 2) Our method outputs the whole transition at once instead of one frame at a time. 3) The self-attention mechanism can gather information about all frames regardless of their distance. These three features enhance our method’s ability to plan the transition motion at a global level. They can prevent the character’s pose from changing abruptly or reaching the target pose too early during the transition.

We also qualitatively compare our method with the Δ -Interp [Oreshkin et al. 2022] in Figure 7. Δ -Interp predicts a delta motion on top of the Slerp between keyframes and sums them up as the final output. The authors of Δ -Interp claim that predicting a delta motion simplifies the motion in-betweening task as Slerp can produce a good initial prediction. However, we argue that using Slerp can lead to artifacts that cannot be fixed by the delta motion predicted by Δ -Interp, while our method’s direct output of transitions can achieve better results. The issue with Slerp is that it only takes two keyframes into account. The interpolated motion can be inconsistent with the rest of the context frames. Take the root rotation of a character as an



Fig. 6. Motion in-betweenings generated by FCN baseline, ERD-QV, our method and the corresponding ground truth. The character is visualized from left to right for every 5 frames. All generated transitions are based on 10 context frames and 1 target frame (visualized by semi-transparent poses). The transition length is 30 frames for first two columns and 45 frames for the last two columns. We highlight the part of transition (enclosed by orange dashed lines) where our method perform better than other methods. For more results, please refer to the supplementary video.

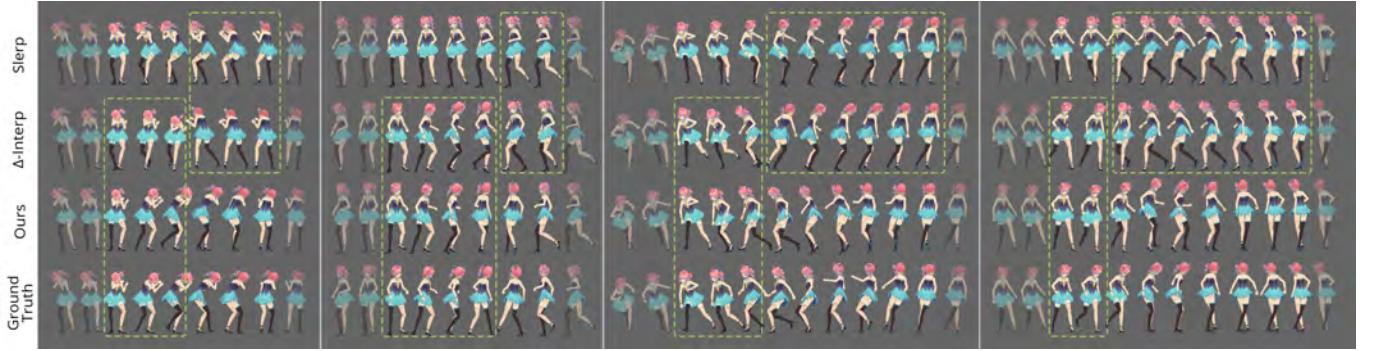


Fig. 7. A qualitative comparison between our method and Δ -Interp. The transition length is 30 frames for first two columns and 45 frames for the last two columns. Similar poses are enclosed by green dashed lines. When the results of Slerp is significantly different from the ground truth, we can observe a sudden change of character pose in the result of Δ -Interp.

example. The context frames may suggest that the character rotates to the left, while the Slerp of keyframes may lead to a rotation to the right. This inconsistency can result in a sudden change of character pose in the generated transition when the delta motion predicted by Δ -Interp is applied. In Figure 7, we highlight the similar poses generated by different methods by green dashed lines. We can see that the initial parts of the motions generated by Δ -Interp are similar to the ground truth. However, as the result of Slerp deviates from the ground truth, the Δ -Interp finally reaches a point where it can no longer produce delta motion that "fixes" the result of Slerp, leaving the last part of the transition similar to the Slerp's result. On the contrary, since our method directly predicts the output motion, it is unlikely to run into such issues. Besides the problem introduced by Slerp, the performance of Δ -Interp quickly deteriorates as the transition length exceeds the maximum length in the training stage. We can see from the last columns of Figure 7 that the transitions produced by Δ -Interp have far fewer details compared with the ground truth and our results. Δ -Interp adopts a learned positional encoding by feeding a one-hot encoding into fully-connected layers. In contrast, our Keyframe Positional Encoding and Learned Relative Positional Encoding demonstrate better robustness in generating

longer transitions. For more qualitative results, please refer to the accompanying video.

4.4 Effectiveness of Proposed Positional Encodings

To prove the effectiveness of our proposed Keyframe Positional Encoding and Learned Relative Positional Encoding, we conduct an ablation study by removing or replacing the positional encodings of the Context Transformer network. Table 2 quantitatively compares different variants of the Context Transformer. pe_{sin_abs} represents the Context Transformer implemented using the vanilla Transformer Encoder [Vaswani et al. 2017] with absolute sinusoidal positional encoding. While the model can produce a moderate result within 30 frames (the maximum transition length during training), it fails to generalize to transitions longer than 30 frames (see its results on 45 frames). To alleviate this issue, we use the relative sinusoidal positional encoding pe_{sin_rel} instead, following the efficient implementation introduced in [Huang et al. 2018]. It demonstrated a boosted performance on transition with 45 frames. However, results reported by pe_{sin_abs} and pe_{sin_rel} are inferior to RNN-based ERD-QV (see the second last row in Table 2). We further add our Keyframe Positional Encoding ($pe_{sin_rel} + pe_k$) and replace relative

Table 2. The performance of Context Transformer \mathcal{T}_{ctx} on LAFAN1 with different choices of positional encoding.

Lengths (frames)	L2P		L2Q		NPSS	
	30	45	30	45	30	45
pe_{sin_abs}	1.08	3.39	0.62	1.41	0.1283	0.7045
pe_{sin_rel}	1.07	2.66	0.61	1.09	0.1248	0.3947
$pe_{sin_rel} + pe_{kf}$	1.05	2.08	0.60	0.96	0.1256	0.3527
pe_{rel}	1.07	2.00	0.60	0.93	0.1235	0.3674
$pe_{rel} + pe_{kf}$ (full \mathcal{T}_{ctx})	1.04	1.94	0.59	0.91	0.1210	0.3349
ERD-QV	1.28	2.24	0.69	0.94	0.1328	0.3311
ERD-QV w/ pe_{kf}	1.23	2.15	0.66	0.97	0.1333	0.3755

sinusoidal positional encoding with our Learned Relative Positional Encoding (pe_{rel}). They both exhibit improved performance on 45 frames. Finally, we show the results with both proposed positional encoding ($pe_{rel} + pe_{kf}$). In summary, our proposed positional encodings can enhance the model’s generalization to long transitions lengths. They are essential for our Transformer-based architecture outperform RNN-based methods. Moreover, we replace the time-to-arrival embedding (zta) of ERD-QV with our Keyframe Positional Encoding to see how pe_{kf} performs with RNN. It reports similar results as ERD-QV (see the last row of Table 2). We think the reason is mainly due to the autoregressive nature of RNN-based method. As Transformer generates a whole transition at once, the relative distance to the context frames is important. But for RNN, its hidden state carries sufficient past context for it to generate the next frame, making the distance to context frames less vital. While distance to the target frame is crucial for both Transformer and RNN method, zta already carries this information, making pe_{kf} non-essential for the RNN-based method.

4.5 Ablation Study

4.5.1 Loss Terms. We perform an ablation study on loss terms used in the training of Context Transformer and Detail Transformer. Table 3 shows the quantitative results trained with different sets of loss terms. L_{rot} and L_{pos} are trivial reconstruction loss terms improving L2Q and L2P metrics respectively. Our early experiments discovered that the transitions generated by the Context Transformer are likely to be discontinuous at both ends. Adding L_{smooth} can penalize for such discontinuities. It is reflected by the improvement of NPSS metrics after adding L_{smooth} , since eliminating discontinuities can make the generated transition more similar to the ground truth on the frequency domain. The L_c is another reconstruction loss term for the Detail Transformer to predict foot contact information, which does not noticeably affect the benchmark metrics when applied alone. However, when both L_{foot} and L_c are applied, foot-sliding artifacts are alleviated and reflected by metric improvements shown in the last row of Table 3.

4.5.2 Masks. We investigate the effectiveness of the masks used in our framework by removing the input mask m_{in} and attention mask m_{atten} from the Context Transformer. Results in Table 4 (1st vs. 2nd

Table 3. An ablation study on the loss terms used in the training of Context Transformer \mathcal{T}_{ctx} and Detail Transformer \mathcal{T}_{det} . Metrics are evaluated on LAFAN1.

Lengths (frames)	L2P		L2Q		NPSS	
	30	45	30	45	30	45
$\mathcal{T}_{ctx} + L_{rot}$	3.86	5.92	0.63	0.97	0.1284	0.3553
$+L_{pos}$	1.05	1.95	0.60	0.92	0.1267	0.3505
$+L_{smooth}$ (full \mathcal{T}_{ctx})	1.04	1.94	0.59	0.91	0.1210	0.3349
$\mathcal{T}_{det} + L_{rot}$	7.32	9.60	0.59	0.91	0.1215	0.3371
$+L_{pos}$	0.93	1.76	0.55	0.88	0.1153	0.3236
$+L_c$	0.94	1.77	0.55	0.88	0.1153	0.3257
$+L_{foot}$ (full \mathcal{T}_{det})	0.89	1.68	0.54	0.87	0.1124	0.3217

row, 3rd vs. 4th row) show that removing m_{in} would slightly deteriorate the model’s performance, mainly caused by the inability to distinguish meaningful values from placeholder values after the removal. Similarly, after removing m_{atten} , multi-headed self-attention would be corrupted by placeholder values on every Transformer layer, leading to a significant decrease in model performance in Table 4 (1st vs. 3rd row, 2nd vs. 4th row).

Table 4. An ablation study on masks used in the Context Transformer \mathcal{T}_{ctx} . m_{in} and m_{atten} represents the input mask and attention mask of \mathcal{T}_{ctx} . Metrics are evaluated on LAFAN1.

Lengths (frames)	L2P		L2Q		NPSS	
	30	45	30	45	30	45
\mathcal{T}_{ctx}	1.04	1.94	0.59	0.91	0.1210	0.3349
(w/o) m_{in}	1.04	1.95	0.60	0.93	0.1221	0.3566
(w/o) m_{atten}	1.41	2.91	0.64	1.01	0.1396	0.3846
(w/o) m_{in} & m_{atten}	1.48	2.92	0.66	1.03	0.1423	0.3928

4.6 Generating longer In-betweening on Quadrupeds

We further evaluate our method on the quadruped motion dataset introduced in [Zhang et al. 2018]. The dataset has 147541 frames in total and we take roughly a third of it (33376 frames) as test set. We train both the Context and Detail Transformer with 60 frames (2 seconds) of maximum transition length to see their performance on predicting longer transitions. We slide a window of 72 frames and offset it by 1 frame to obtain 111680 clips as the training set. For the test set, the window length and offset are set to 135 and 20 frames respectively, resulting in 1569 clips. Similar to Section 4.2, we use the L2P, L2Q, and NPSS metrics and compare our method with the interpolation baseline. The results are presented in Table 5. For qualitative results, please refer to the accompanying video.

4.7 Motion In-betweening Tool in Autodesk Maya

We implemented a tool leveraging our system inside Autodesk Maya, a widely used digital content creation software for animation authoring. Different from the tool in [Harvey et al. 2020] where users inject noises to generate in-betweening variations, our tool is more

Table 5. Qualitative results on quadruped dataset [Zhang et al. 2018]. The models are trained with maximum transition length of 60 frames.

	L2P			L2Q			NPSS		
Lengths	30	60	90	30	60	90	30	60	90
Interp.	0.70	1.26	1.96	0.56	0.72	0.84	0.0921	0.3139	0.5852
\mathcal{T}_{ctx}	0.45	0.74	1.50	0.41	0.55	0.74	0.0518	0.1784	0.4145
\mathcal{T}_{det}	0.35	0.63	1.27	0.31	0.46	0.66	0.0450	0.1680	0.3889

artist-friendly by giving animators fine control over generated results through full or partial pose constraints between keyframes. Figure 8 demonstrates our tool with transitions generated under different constrained modes. The tool provides three buttons: generating transition for user-selected time range (green button), adding constraints to the selected joints at the current frame (cyan button), and removing constraints from the selected joints (red button). When in fully constrained mode (Figure 8b), arbitrary poses can be specified between context and target poses. The generated transition will satisfy the given spatial-temporal constraint. Although we can achieve the same goal by predicting each sub-sequence between constraints separately, feeding constraints into the model are better since it outputs the whole transition through a single model inference and generates more coherent transitions near constrained poses. When in partially constrained mode (Figure 8c), only some joints are constrained. Animators can focus on parts of the character’s body where a particular pose or contact is required and let the tool figure out the natural transition for the rest of the body.

We conducted a small-scale user study to evaluate the effectiveness of our tool. Three professional animators were invited to complete three animation transitions (details shown in Appendix E) with and without our tool (six tasks in total). The tasks were presented in a random order and the time spent on each task was recorded. The results in Table 6 indicate that our tool can provide about 100% speed up. We also collected positive feedback on the pose constraint functionality. Besides providing fine control over the generated result, it can act as a way to manage key poses (an analogy to traditional keyframes) so that animators can safely regenerate any transition without worrying about their handcraft key poses being overwritten.

We further evaluate the speed of our system comparing with [Harvey et al. 2020] and [Oreshkin et al. 2022] in Table 7. Our method provides a significant speedup. Moreover, the time taken by our method is less sensitive to the transition length as frames are processed in parallel, while [Harvey et al. 2020] [Oreshkin et al. 2022] are autoregressive during inference.

5 DISCUSSIONS AND LIMITATIONS

Typically, during the training of the Transformer, dropout is applied to the multi-headed self-attention of each sub-layer. However, we find through experiments that adding dropout does not lead to better model performance. All models in our experiments are trained without dropout. We examined the generated animation when dropout is applied. It has many discontinuities as if some high-frequency noise is added to the result. Adding dropout might

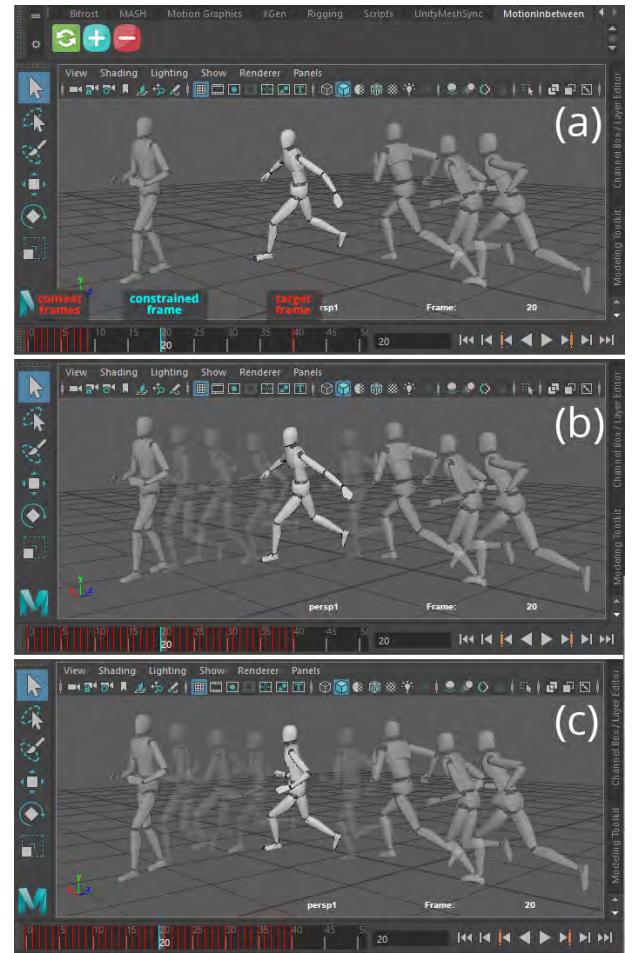


Fig. 8. Transition generation tool inside Autodesk Maya. (a) A screenshot taken before running the tool. From left to right: target frame pose (semi-transparent), constrained pose, and context frame poses (semi-transparent). On the timeline, key frames and pose constraints are visualized by red and cyan bars. (b) Transition generated in fully constrained mode. (c) Transition result in partially constrained mode. In this case, only root joint’s position and rotation are constrained.

Table 6. The average time taken by three professional animators to complete three animation transitions in our user study.

Clip	Frames (fps=30)	Time Spent w/o Tool	Time Spent w/ Tool	Speed Improvement in %
A	30	3.25h	1.75h	86%
B	60	4.92h	2.42h	103%
C	60	4.58h	2.17h	112%

be helpful if Transformer is applied to motion denoising tasks, as input motions are expected to have glitches. However, for motion in-betweening tasks, it is not beneficial as context frames are not noisy during inference. We also tried other strategies for tweaking

Table 7. A speed performance comparison between our method and ERD-QV and Δ -Interp. Batch size indicates the number of input/output clips in a single model inference. All values are in milliseconds. Data are collected on Intel i7-7700K @ 4.20 GHZ CPU and Nvidia GTX 1080Ti GPU.

Batch Size	1			100		
Transition Length	30	60	90	30	60	90
ERD-QV	33.1	57.2	85.2	36.7	64.0	91.6
Δ -Interp	32.5	33.9	36.0	352.5	428.5	478.7
ours (T_{ctx} only)	6.3	6.2	7.0	12.9	19.5	26.1
ours	12.1	12.7	11.9	19.2	26.4	32.7

the dropout rate during training, such as gradually reducing dropout to zero. But the resulting model does not significantly differ from models trained with zero dropout. There might be concerns that zero dropout might lead to overfitting. However, adopting random transition length during training is enough to overcome the issue. We also found sampling random constrained frames useful during the training of Detail Transformer. It helps to fix overfitting and improve benchmark metrics (see Figure 11 in Appendix C).

Similar to many data-driven methods, our system might generate undesirable results when the input context is ambiguous or too distinct from the motion covered in the training set. Figure 9 shows an undesired transition generated from an ambiguous input and can be further fixed by adding pose constraints. Moreover, even with the help of Detail Transformer, foot-sliding artifacts might still exist. Taking advantage of the foot contact information in the output and applying inverse kinematics could alleviate the problem. Although our method can generalize to longer transitions, when the transition is too long (more than 1.5 times the maximum transition length during training), the quality of generated transition rapidly decreases. Another limitation of our method is that the pose constraints are applied to the joints' local position and rotations. However, in practical use scenarios, animators may prefer to constrain end effectors of a joint chain (hands, feet, etc.). It would require the model to handle inverse kinematics while generating in-betweening. We will further investigate it in future work.

6 CONCLUSION

This paper presents a novel framework that generates motion in-betweening through two stages by leveraging Transformer Encoder network architecture. It reports state-of-the-art motion in-betweening benchmark results on LAFAN1 with less trainable parameters, significantly outperforming existing RNN, CNN and Transformer-based methods. With the help of our proposed Keyframe Positional Encoding and Learned Relative Positional Encoding, our method is robust in generating plausible transitions with variable lengths. It can generalize to longer transitions exceeding the maximum transition length seen during training with good quality. Our network trains and evaluates at a faster speed by taking advantage of the parallel processing power of GPU. We demonstrate how this system can be applied in animation authoring software. The proposed tool boost users' productivity with real-time feedback and fine control over generated results through full and partial pose constraints.

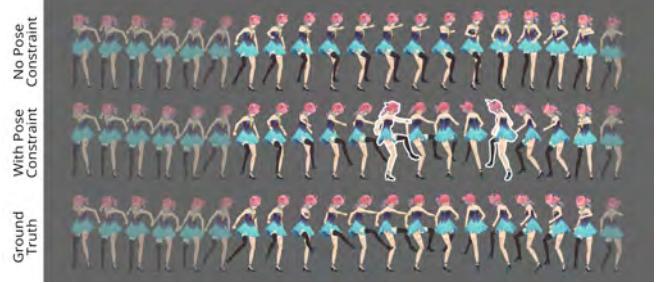


Fig. 9. An example of undesired transition generated based on ambiguous input (visualized every two frames). We expect to create a transition of the character spinning around herself (refer to the ground truth in the third row). Although the context frames (semi-transparent poses on the left) exhibit the tendency of character spinning, the gap between the last context frame and the target frame (the last semi-transparent pose on the right) is small. The character can either rotate 360° then transition to the target pose (desired result) or follow its inertia for a few frames then rotate in the opposite direction to the target pose (undesired result in the first row). The ambiguity of input can be resolved by adding pose constraints (poses with white outlines) to get the desired transition shown in the second row.

ACKNOWLEDGMENTS

This work was done with the help of Netease Fuxi AI Lab and Thunderfire Studio. We appreciate their kindness for providing the art assets shown in this paper. We would like to thank Changjie Fan, Jie Hou and Yang Liu for their valuable feedback. We also want to thank Wei Zhang, Zhongqiao Xu, Qing Liu, Lie Ma and Hanming Zhang from Cangjie Animation Center for arranging and participating in our user study. This work was supported in part by the NSF China (No. 62172363). Kun Zhou was partially supported by the Xplore Prize.

REFERENCES

- Emre Aksan, Manuel Kaufmann, Peng Cao, and Otmar Hilliges. 2021. A spatio-temporal transformer for 3D human motion prediction. In *2021 International Conference on 3D Vision (3DV)*. IEEE, 565–574.
- Okan Arikan and David A Forsyth. 2002. Interactive motion generation from examples. *ACM Transactions on Graphics (TOG)* 21, 3 (2002), 483–490.
- Autodesk. 2022. Maya. <https://www.autodesk.com/products/maya/overview>
- Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450* (2016).
- Emad Barsoum, John Kender, and Zicheng Liu. 2018. HP-GAN: Probabilistic 3D Human Motion Prediction via GAN. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 1418–1427.
- Philippe Beaudoin, Stelian Coros, Michiel van de Panne, and Pierre Poulin. 2008. Motion-motif graphs. In *Proceedings of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. 117–126.
- Blender Foundation. 2022. Blender. <http://www.blender.org>
- Michael Büttner and Simon Clavet. 2015. Motion Matching. https://www.youtube.com/watch?v=z_wpgHFSWss&t=658s.
- Yujun Cai, Lin Huang, Yiwei Wang, Tat-Jen Cham, Jianfei Cai, Junsong Yuan, Jun Liu, Xu Yang, Yiheng Zhu, Xiaohui Shen, et al. 2020. Learning progressive joint propagation for human motion prediction. In *Proceedings of the European Conference on Computer Vision (ECCV)*. Springer, 226–242.
- Yujun Cai, Yiwei Wang, Yiheng Zhu, Tat-Jen Cham, Jianfei Cai, Junsong Yuan, Jun Liu, Chuanxian Zheng, Sijie Yan, Henghui Ding, et al. 2021. A Unified 3D Human Motion Synthesis Model via Conditional Variational Auto-Encoder. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 11645–11655.
- Jinxiang Chai and Jessica K Hodgins. 2005. Performance animation from low-dimensional control signals. *ACM Transactions on Graphics (TOG)* 24, 3 (2005), 686–696.

- Jinxiang Chai and Jessica K Hodgins. 2007. Constraint-based motion optimization using a statistical dynamic model. *ACM Transactions on Graphics (TOG)* 26, 3 (2007), 8–es.
- Mia Xu Chen, Orhan Firat, Ankur Bapna, Melvin Johnson, Wolfgang Macherey, George Foster, Llion Jones, Mike Schuster, Noam Shazeer, Niki Parmar, et al. 2018. The Best of Both Worlds: Combining Recent Advances in Neural Machine Translation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 76–86.
- Hsu-kuang Chiu, Ehsan Adeli, Borui Wang, De-An Huang, and Juan Carlos Niebles. 2019. Action-agnostic human pose forecasting. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, 1423–1432.
- Loïc Ciccone, Cengiz Özturel, and Robert W Sumner. 2019. Tangent-space optimization for interactive animation control. *ACM Transactions on Graphics (TOG)* 38, 4 (2019), 1–10.
- Zihang Dai, Zhilin Yang, Yiming Yang, Jaime G Carbonell, Quoc Le, and Ruslan Salakhutdinov. 2019. Transformer-XL: Attentive Language Models beyond a Fixed-Length Context. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. 2978–2988.
- Lingwei Dang, Yongwei Nie, Chengjiang Long, Qing Zhang, and Guiqing Li. 2021. MSR-GCN: Multi-Scale Residual Graph Convolution Networks for Human Motion Prediction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 11467–11476.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).
- Yinglin Duan, Tianyang Shi, Zhengxia Zou, Yenan Lin, Zhehui Qian, Bohan Zhang, and Yi Yuan. 2021. Single-Shot Motion Completion with Transformer. *arXiv preprint arXiv:2103.00776* (2021).
- Petros Faloutsos, Michiel Van de Panne, and Demetri Terzopoulos. 2001. Composable controllers for physics-based character animation. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*. 251–260.
- Anthony C Fang and Nancy S Pollard. 2003. Efficient synthesis of physically valid human motion. *ACM Transactions on Graphics (TOG)* 22, 3 (2003), 417–426.
- Katerina Fragkiadaki, Sergey Levine, Panna Felsen, and Jitendra Malik. 2015. Recurrent network models for human dynamics. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 4346–4354.
- Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. 2017. Convolutional sequence to sequence learning. In *International Conference on Machine Learning*. PMLR, 1243–1252.
- Romi Geleijn, Adrian Radziszewski, Julia Beryl van Straaten, and Henrique Galvan Debara. 2021. Lightweight Quaternion Transition Generation with Neural Networks. In *2021 IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops (VRW)*. IEEE, 579–580.
- 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.
- Keith Grochow, Steven L Martin, Aaron Hertzmann, and Zoran Popović. 2004. Style-based inverse kinematics. *ACM Transactions on Graphics (TOG)* 23, 3 (2004), 522–531.
- Liang-Yan Gui, Yu-Xiong Wang, Xiaodan Liang, and José MF Moura. 2018. Adversarial geometry-aware human motion prediction. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 786–803.
- Félix G Harvey and Christopher Pal. 2018. Recurrent transition networks for character locomotion. In *SIGGRAPH Asia 2018 Technical Briefs*. 1–4.
- 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.
- Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 1026–1034.
- Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 770–778.
- Alejandro Hernandez, Jurgen Gall, and Francesc Moreno-Noguer. 2019. Human motion prediction via spatio-temporal inpainting. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 7134–7143.
- Jessica K Hodgins, Wayne L Wooten, David C Brogan, and James F O’Brien. 1995. Animating human athletics. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*. 71–78.
- Daniel Holden, Oussama Kanoun, Maksym Perepichka, and Tiberiu Popa. 2020. Learned motion matching. *ACM Transactions on Graphics (TOG)* 39, 4 (2020), 53–1.
- Daniel Holden, Taku Komura, and Jun Saito. 2017. Phase-functioned neural networks for character control. *ACM Transactions on Graphics (TOG)* 36, 4 (2017), 1–13.
- Daniel Holden, Jun Saito, and Taku Komura. 2016. A deep learning framework for character motion synthesis and editing. *ACM Transactions on Graphics (TOG)* 35, 4 (2016), 1–11.
- Daniel Holden, Jun Saito, Taku Komura, and Thomas Joyce. 2015. Learning motion manifolds with convolutional autoencoders. In *SIGGRAPH Asia 2015 technical briefs*. 1–4.
- Cheng-Zhi Anna Huang, Ashish Vaswani, Jakob Uszkoreit, Noam Shazeer, Ian Simon, Curtis Hawthorne, Andrew M Dai, Matthew D Hoffman, Monica Dinculescu, and Douglas Eck. 2018. Music transformer. *arXiv preprint arXiv:1809.04281* (2018).
- Ashesh Jain, Amir R Zamir, Silvio Savarese, and Ashutosh Saxena. 2016. Structural-rnn: Deep learning on spatio-temporal graphs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 5308–5317.
- Manuel Kauffmann, Emre Aksan, Jie Song, Fabrizio Pece, Remo Ziegler, and Otmar Hilliges. 2020. Convolutional autoencoders for human motion infilling. In *2020 International Conference on 3D Vision (3DV)*. IEEE, 918–927.
- Jihoon Kim, Taehyun Byun, Seungyoun Shin, Jungdam Won, and Sungjoon Choi. 2022. Conditional Motion In-betweening. *arXiv preprint arXiv:2202.04307* (2022).
- Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).
- Lucas Kovar and Michael Gleicher. 2004. Automated extraction and parameterization of motions in large data sets. *ACM Transactions on Graphics (TOG)* 23, 3 (2004), 559–568.
- Lucas Kovar, Michael Gleicher, and Frédéric Pighin. 2002. Motion Graphs. *ACM Transactions on Graphics (TOG)* 21, 3 (July 2002), 473–482.
- Jehee Lee, Jinxiang Chai, Paul SA Reitsma, Jessica K Hodgins, and Nancy S Pollard. 2002. Interactive control of avatars animated with human motion data. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*. 491–500.
- Yongjoo Lee, Kevin Wampler, Gilbert Bernstein, Jovan Popović, and Zoran Popović. 2010. Motion fields for interactive character locomotion. In *ACM SIGGRAPH Asia 2010 papers*. 1–8.
- Andreas M Lehrmann, Peter V Gehler, and Sebastian Nowozin. 2014. Efficient nonlinear markov models for human motion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 1314–1321.
- Jiaman Li, Ruben Villegas, Duygu Ceylan, Jimei Yang, Zhengfei Kuang, Hao Li, and Yajie Zhao. 2021. Task-generic hierarchical human motion prior using vaes. In *2021 International Conference on 3D Vision (3DV)*. IEEE, 771–781.
- Hung Yu Ling, Fabio Zinno, George Cheng, and Michiel Van De Panne. 2020. Character controllers using motion vaes. *ACM Transactions on Graphics (TOG)* 39, 4 (2020), 40–1.
- Xiaoli Liu, Jianqin Yin, Jin Liu, Pengxiang Ding, Jun Liu, and Huaping Liu. 2020. Trajectorycnn: a new spatio-temporal feature learning network for human motion prediction. *IEEE Transactions on Circuits and Systems for Video Technology* 31, 6 (2020), 2133–2146.
- Juliette Martinez, Michael J Black, and Javier Romero. 2017. On human motion prediction using recurrent neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2891–2900.
- Jianyuan Min and Jinxiang Chai. 2012. Motion graphs++ a compact generative model for semantic motion analysis and synthesis. *ACM Transactions on Graphics (TOG)* 31, 6 (2012), 1–12.
- Jianyuan Min, Yen-Lin Chen, and Jinxiang Chai. 2009. Interactive generation of human animation with deformable motion models. *ACM Transactions on Graphics (TOG)* 29, 1 (2009), 1–12.
- Tomohiko Mukai and Shigeru Kuriyama. 2005. Geostatistical motion interpolation. *ACM Transactions on Graphics (TOG)* 24, 3 (2005), 1062–1070.
- J Thomas Ngo and Joe Marks. 1993. Spacetime constraints revisited. In *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*. 343–350.
- Boris N Oreshkin, Antonios Valkanas, Félix G Harvey, Louis-Simon Ménard, Florent Bocquelet, and Mark J Coates. 2022. Motion Inbetweening via Deep Δ -Interpolator. *arXiv preprint arXiv:2201.06701* (2022).
- Sang Il Park, Hyun Joon Shin, and Sung Yong Shin. 2002. On-line locomotion generation based on motion blending. In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics Symposium on Computer animation*. 105–111.
- Dario Pavlo, Christoph Feichtenhofer, Michael Auli, and David Grangier. 2020. Modeling human motion with quaternion-based neural networks. *International Journal of Computer Vision* 128, 4 (2020), 855–872.
- Dario Pavlo, David Grangier, and Michael Auli. 2018. Quaternet: A quaternion-based recurrent model for human motion. *arXiv preprint arXiv:1805.06485* (2018).
- Mathis Petrovich, Michael J Black, and Güldem Varol. 2021. Action-conditioned 3d human motion synthesis with transformer vae. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 10985–10995.
- Charles Rose, Michael F Cohen, and Bobby Bodenheimer. 1998. Verbs and adverbs: Multidimensional motion interpolation. *IEEE Computer Graphics and Applications* 18, 5 (1998), 32–40.
- Charles Rose, Brian Guenter, Bobby Bodenheimer, and Michael F Cohen. 1996. Efficient generation of motion transitions using spacetime constraints. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*. 147–154.
- Alla Safanova and Jessica K. Hodgins. 2007. Construction and Optimal Search of Interpolated Motion Graphs. *ACM Transactions on Graphics (TOG)* 26, 3 (July 2007), 106–es.
- Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. 2018. Self-Attention with Relative Position Representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language*

- Technologies, Volume 2 (Short Papers)*. 464–468.
- Sebastian Starke, He Zhang, Taku Komura, and Jun Saito. 2019. Neural state machine for character-scene interactions. *ACM Transactions on Graphics (TOG)* 38, 6 (2019), 209–1.
- Sebastian Starke, Yiwei Zhao, Taku Komura, and Kazi Zaman. 2020. Local motion phases for learning multi-contact character movements. *ACM Transactions on Graphics (TOG)* 39, 4 (2020), 54–1.
- Sebastian Starke, Yiwei Zhao, Fabio Zinno, and Taku Komura. 2021. Neural animation layering for synthesizing martial arts movements. *ACM Transactions on Graphics (TOG)* 40, 4 (2021), 1–16.
- Xiangjun Tang, He Wang, Bo Hu, Xu Gong, Ruifan Yi, Qilong Kou, and Xiaogang Jin. 2022. Real-time Controllable Motion Transition for Characters. *arXiv preprint arXiv:2205.02540* (2022).
- Jochen Tautges, Arno Zinke, Björn Krüger, Jan Baumann, Andreas Weber, Thomas Helten, Meinard Müller, Hans-Peter Seidel, and Bernd Eberhardt. 2011. Motion reconstruction using sparse accelerometer data. *ACM Transactions on Graphics (TOG)* 30, 3 (2011), 1–12.
- Graham W Taylor, Geoffrey E Hinton, and Sam Roweis. 2006. Modeling human motion using binary latent variables. *Advances in neural information processing systems* 19 (2006).
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*. 5998–6008.
- Ruben Villegas, Jimei Yang, Duygu Ceylan, and Honglak Lee. 2018. Neural kinematic networks for unsupervised motion retargetting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 8639–8648.
- Jack M Wang, David J Fleet, and Aaron Hertzmann. 2007. Gaussian process dynamical models for human motion. *IEEE transactions on pattern analysis and machine intelligence* 30, 2 (2007), 283–298.
- Qiang Wang, Bei Li, Tong Xiao, Jingbo Zhu, Changliang Li, Derek F Wong, and Lidia S Chao. 2019. Learning deep transformer models for machine translation. *arXiv preprint arXiv:1906.01787* (2019).
- Tingwu Wang, Yunrong Guo, Maria Shugrina, and Sanja Fidler. 2020. Unicon: Universal neural controller for physics-based character motion. *arXiv preprint arXiv:2011.15119* (2020).
- Andrew Witkin and Michael Kass. 1988. Spacetime constraints. In *Proceedings of the 15th annual conference on Computer graphics and interactive techniques*. 159–168.
- He Zhang, Sebastian Starke, Taku Komura, and Jun Saito. 2018. Mode-adaptive neural networks for quadruped motion control. *ACM Transactions on Graphics (TOG)* 37, 4 (2018), 1–11.
- Xinyi Zhang and Michiel van de Panne. 2018. Data-driven autocompletion for keyframe animation. In *Proceedings of the 11th Annual International Conference on Motion, Interaction, and Games*. 1–11.
- 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.
- Yi Zhou, Jingwan Lu, Connelly Barnes, Jimei Yang, Sitao Xiang, et al. 2020. Generative tweening: Long-term inbetweening of 3d human motions. *arXiv preprint arXiv:2005.08891* (2020).

A IMPLEMENTATION DETAILS OF THE FCN BASELINE

Table 8 shows the implementation details of the FCN baseline. Note that we deliberately limit the number of downsamplings and upsamplings in the network to two. As the minimum transition length is five frames in LAFAN1’s transition benchmark, the input size on the time dimension is not long enough to afford more downsamplings and upsamplings without adding extra padding to the input. The input of FCN is exactly the same as the Context Transformer.

Table 8. The architecture of FCN baseline. For each convolutional layer, except for the last layer, is followed by a 1D batch normalization layer and a ReLU activation.

Type	Kernel	Stride	Padding	Dilation	Output
Conv1d	3×3	1	1	1	256
Conv1d	3×3	2	1	1	512
Conv1d	3×3	1	1	1	512
Conv1d	3×3	2	1	1	1024
Conv1d	3×3	1	1	1	1024
Conv1d	3×3	1	2	2	1024
Conv1d	3×3	1	1	1	1024
TransConv1d	3×3	2	1	1	512
Conv1d	3×3	1	1	1	512
TransConv1d	3×3	2	1	1	256
Conv1d	3×3	1	1	1	256
Conv1d	3×3	1	1	1	135

B PER-CLIP COMPARISON ON LAFAN1 BENCHMARK

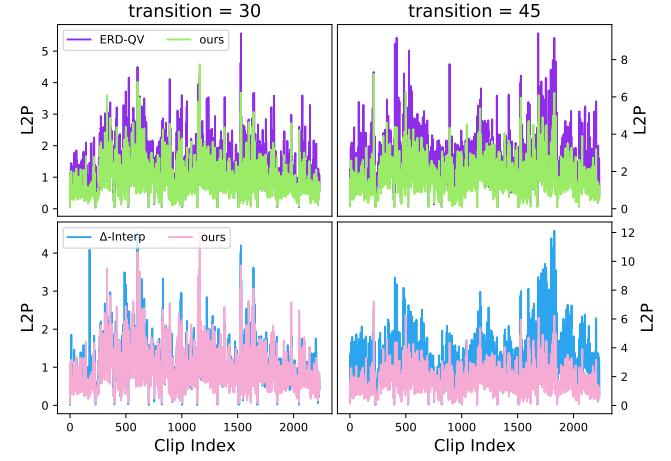


Fig. 10. Per-clip L2P metric comparisons with ERD-QV [Harvey et al. 2020] and Δ -Interp [Oreshkin et al. 2022] on LAFAN1’s transition benchmark. Transition lengths are 30 and 45 frames. All models are trained with maximum transition length equals to 30 frames. Our method report better L2P metrics on majority of benchmark clips.

C RANDOM POSE CONSTRAINT SAMPLING

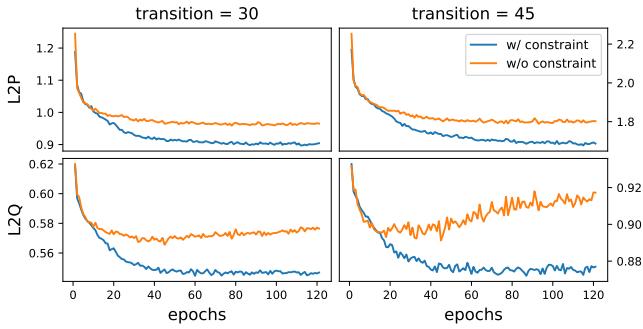


Fig. 11. The validation loss of two Detail Transformers, one of which adopts random pose constraint sampling during training, while the other does not. The result shows that random sampling of constrained frames can improve performance and alleviate overfitting.

D PERFORMANCE OF THE CONTEXT TRANSFORMER WITH DIFFERENT HYPERPARAMETERS

We compare multiple Context Transformers trained with different layer dimension d , layer count L and head count H . We report their L2P metric on LAFAN1’s transition benchmark with transition lengths equal to 5, 15, 30, and 45 frames.

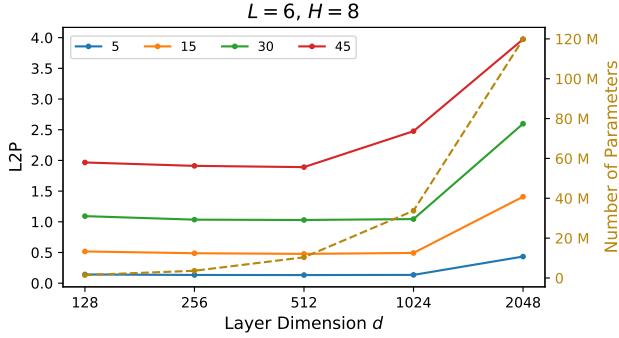


Fig. 12. We increase d from 128 to 2048 while fixing $L = 6$ and $H = 8$. Increasing d improves performance initially and reduces performance after reaching 512 or 1024. Since the performance of $d = 512$ and $d = 1024$ are similar on transition lengths within 30 frames (maximum length during training), but $d = 1024$ considerably deteriorates performance on 45 frames, we use $d = 512$ in our final model.

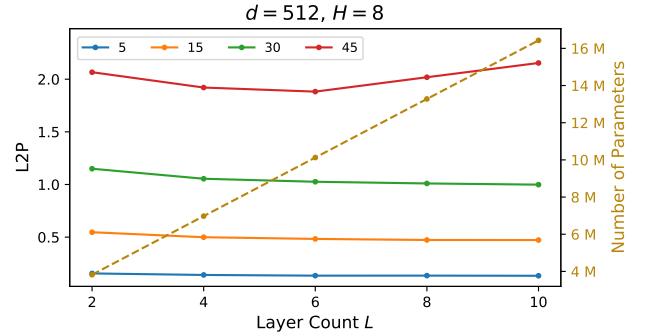


Fig. 13. Fixing $d = 512$ and $H = 8$, we evaluate Context Transformer’s performance with different layer count L . Increasing L improves performance on transition lengths less than 30 frames but hurts performance on 45 frames when L exceeds 6. We choose $L = 6$ in our final model.

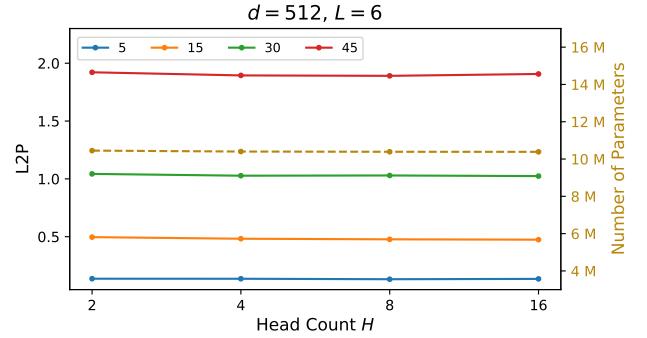
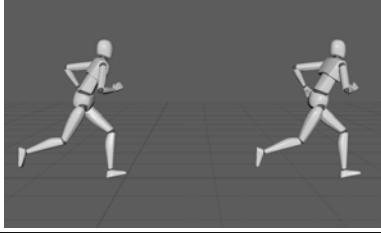
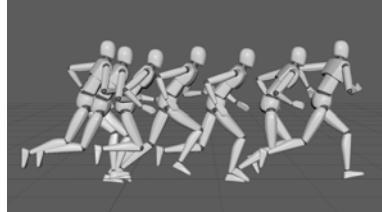
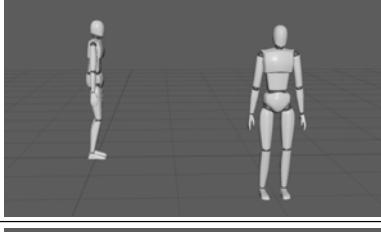
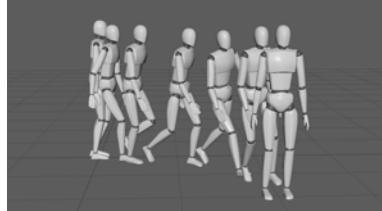
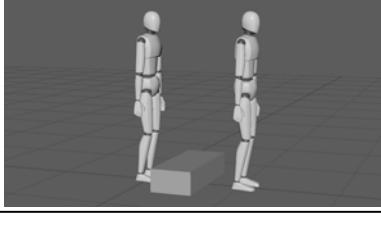
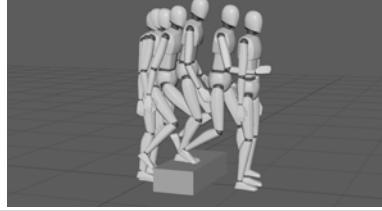


Fig. 14. When keeping $d = 512$ and $L = 6$, we experiment with various combinations of head count H and head dimension d_K , where $d = H \times d_K$. The results show that their performance differences are negligible.

E ANIMATION TASKS IN USER STUDY

Table 9. The animation tasks in our user study. Animators were given the initial poses and asked to create animations similar to the reference with and without our motion in-betweening tool.

Clip	Length ^a	Description	Initial Poses	Reference Animation
A	30	A running cycle.		
B	60	Walk forward and turn right.		
C	60	Walk over a box.		

^aLength is in frames. Frame rate is 30fps.