Continuous Intermediate Token Learning with Implicit Motion Manifold for Keyframe Based Motion Interpolation

Clinton A. Mo[1], Kun Hu[1,*], Chengjiang Long[2], Zhiyong Wang[1]
[1]School of Computer Science, The University of Sydney, NSW 2006, Australia
[2]Meta Reality Labs, Burlingame, CA, USA
clmo6615@uni.sydney.edu.au, {kun.hu, zhiyong.wang}@sydney.edu.au, clong1@meta.com

Abstract

Deriving sophisticated 3D motions from sparse keyframes is a particularly challenging problem, due to continuity and exceptionally skeletal precision. The action features are often derivable accurately from the full series of keyframes, and thus, leveraging the global context with transformers has been a promising data-driven embedding approach. However, existing methods are often with inputs of interpolated intermediate frame for continuity using basic interpolation methods with keyframes, which result in a trivial local minimum during training. In this paper, we propose a novel framework to formulate latent motion manifolds with keyframe-based constraints, from which the continuous nature of intermediate token representations is considered. Particularly, our proposed framework consists of two stages for identifying a latent motion subspace, i.e., a keyframe encoding stage and an intermediate token generation stage, and a subsequent motion synthesis stage to extrapolate and compose motion data from manifolds. Through our extensive experiments conducted on both the LaFAN1 and CMU Mocap datasets, our proposed method demonstrates both superior interpolation accuracy and high visual similarity to ground truth motions.

1. Introduction

Pose-to-pose keyframing is a fundamental principle of character animation, and animation processes often rely on key pose definitions to efficiently construct motions [6, 11, 30]. In computer animation, keyframes are temporally connected via interpolation algorithms, which derive intermediate pose attributes to produce smooth transitions between key poses. However, human motion is often complex and difficult to be effectively represented by sparse keyframe sequences alone. While this can be addressed

---
*Corresponding author.
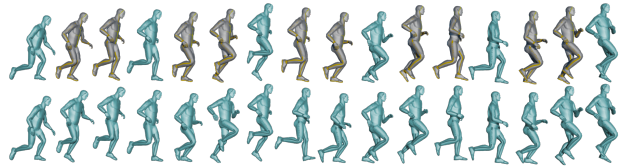 Code available at: https://github.com/MiniEval/CITL

Figure 1. An example of motion interpolation by our method (first row), given the keyframes of a hopping motion (in blue), compared with the ground truth (second row).

by producing denser sequences of key poses, this approach is laborious for animators, thereby increasing the cost of keyframed animation processes. Even with Motion Capture (MoCap) workflows, artists must often resort to keyframing in order to clean artifacts, impose motion constraints, or introduce motion features irreplicable by motion capture performers.

Learning-based motion interpolation methods have recently been proposed as an acceleration of the keyframed animation process, by automatically deriving details within keyframe transitions as shown in Figure 1. Various machine learning methods have been explored to enable more realistic interpolation solutions from high quality MoCap databases, e.g. by using recurrent networks [14, 15, 40] or transformer-based approaches [10, 26, 31]. Guiding data-driven interpolation with real motions is particularly attractive for keyframe animation workflows, as realistic motions often require the greatest amount of keyframing, by virtue of their subtle motion details and physical constraints.

Naturally, as a sequence in-painting problem, motion interpolation can be formulated as a masked sequence-to-sequence task, which the recent popular transformer approach is expected to learn effectively [4, 38, 42, 43]. However, sequential learning of masked continuous attributes with transformers is largely impaired by the conventional masked tokens for intermediate data. A token is defined as an individual data element on the extendable axis of a sequence, namely the temporal axis for motions. In cur-

rent sequence modelling formulations, a token is usually represented by a one-hot vocabulary vector to specify individual words or masked elements, which poses a limitation on continuous attributes. Since continuous attributes can be assigned any real value, there exists no value by which a masking token can be defined without corresponding to an otherwise valid input. Previous approaches have employed transformer decoder-level mask tokens and linear interpolation (LERP)-based tokens have been explored to work around this issue [10, 16, 31]. However, these approaches have innate incompatibilities with the transformer architecture. Singular mask token representations, regardless of their point of introduction, result in discontinuous hidden representations, which are antithetical to the evaluation of continuous motion data. On the other hand, the use of LERP as a pre- or post-processing step necessarily introduces an accurate starting estimate to the solution, which transformer models are prone to becoming over-reliant on [24, 45]. To fully address these limitations, we propose a novel transformer-based framework that learns to model keyframe sequences into latent motion manifold representations for intermediate tokens, which reflects the smooth and continuous nature of human motion.

As illustrated in Figure 2, our proposed framework incorporates three stages with transformers to convert a keyframe sequence into a complete motion sequence: Stage-I is a *keyframe encoding stage* to formulate the overall motion patterns from the keyframe sequence into keyframe context tokens as a guidance for further modelling; Stage-II is an *intermediate token generation stage*, where temporal indices are mapped into intermediate token representations with the keyframe context tokens, which serve as an implicit latent motion manifold constraint; and Stage-III, a *motion synthesis stage*, takes the obtained intermediate tokens by injecting them within the keyframe token sequence, and interpolating them to derive a refined motion sequence estimation.

With this framework, our transformer-based approach exhibits two key advantages over existing approaches that enable its high-quality motion interpolation: a) Manifold learning allows our framework to establish temporal continuity in its latent representation space, and b) The latent motion manifold constrains our transformer model to concentrate its attention exclusively towards motion keyframes, as opposed to intermediate tokens derived from non-keyframe poses, such as those derived from LERP, thereby forcing a necessary alignment between the known and unknown tokens adaptively.

In addition, we identify an adverse link between continuous features and normalisation methods with per-token re-centering. Specifically, layer normalisation (LayerNorm) [1], which is commonly used in transformer architectures, constrains the biases of token features based on their individual distributions. Though this is well-known to be effective with linguistic models [25, 42], continuous data inherently contain biases that should be leveraged at sequence level. Therefore, we introduce a sequence-level re-centering (Seq-RC) technique, where positional pose attributes of keyframes are recentred based on their distribution throughout a motion sequence, and root-mean-square normalisation (RMSNorm) [47] layers are then employed to perform magnitude-only normalisation. Though RMSNorm was initially proposed as only a speedup to LayerNorm, our observations demonstrate that Seq-RC leads to superior performance in terms of accuracy and visual similarity to MoCap sequences.

In summary, our paper's key contributions are threefold:

1. We propose a novel transformer-based architecture consisting of three cooperative stages. It constrains the evaluation of unknown intermediate representations of continuous attributes to the guidance of keyframe context tokens in a learned latent manifold.

2. We devise sequence-level re-centering (Seq-RC) normalisation to effectively operate with real scalar attributes with minimal accuracy loss.

3. Extensive comparisons and ablation results obtained on LaFAN1 and CMU Mocap strongly demonstrate the superiority of our method over the state-of-the-art.

2. Related work

In this section, we explore existing machine learned methods by which motion keyframes, or key tokens of other mediums, can be used to generate full sequences. We also review known methods for intermediate data prediction.

2.1. Motion synthesis and completion

The necessities of motion interpolation techniques have existed since the early days of computer animation. A key advantage of computer animation over traditionally drawn animation is its ability to automatically evaluate a smooth motion from a keyframe representation. The widely accepted method for applying interpolation to motion keyframes is through the use of function curves (F-Curves), by which various mathematical functions can be defined between intervals. The common functions used today for keyframed motion representation are often based on Bézier spline curves [18, 37], though any function can technically be employed for this purpose. In addition, inverse kinematics-based constraints [34] have been used jointly with Markov chain methods [23] and decision graphs [21] to generate constrained keyframe or motion sequence transitions in interactive applications such as 3D video games.

More recently, deep learning methods have enabled effective motion completion from sparse keyframe representations. By formulating motion interpolation as a motion synthesis problem with keyframe constraints, a neural

network-assisted keyframed animation approach is emerging as a more effective solution to the current approaches. Recurrent neural networks have been able to derive realistic motion details from motion keyframe compositions [15, 22, 48] and real-time control schemes [40]. In addition, sequence masking approaches such as BERT-based and autoencoder-based methods [7, 10, 19, 26, 29, 31] have enabled full keyframe sequence analysis for completing motions with a more comprehensive context. However, these methods are severely affected by the tendency of transformer-based networks that toward a trivial local minimum, given an initial LERP starting point. This limitation in transformers has been thoroughly observed and documented as the result of early gradient instabilities in attention weights [2, 24, 42, 45].

Learned pose manifolds have become a prominent approach to synthesis plausible human poses, which restrain pose attributes to a specified space [32, 41]. These methods mainly focus on dense poses from complete motion sequences. Our work extends this concept for an incomplete and sparse scenario, by using the keyframes as constraints to derive an implicit motion manifold.

2.2. Transformer-based temporal in-painting

Inter-frame video interpolation is a similar task to motion interpolation, due to its common goal of predicting transitions between frames on the temporal axis. Like motion in-painting methods, temporal transformers for video interpolation use blended inputs as masks for continuous attributes [28]. Alternatively, when interpolating with an individual interval, the mask tokens can be copied from the previous keyframe, with positional encoding being the sole difference between input tokens [27]. While the feature extraction process of video data can take full advantage of the global context afforded by using transformers, other mechanisms such as convolutions can be integrated with a transformer for video in-painting as well [36]. For skeleton-based motion data, existing studies suggest that graph convolutional networks can improve analytic and synthetic performance [8, 13, 26, 30]; however, the application of pure convolutional approaches for data synthesis is only valid when the interval length between keyframes is constant.

2.3. Masked data modelling

Various mask-based machine learning techniques have been proposed to estimate missing values of incomplete sequences based on their known values. The use of masked token is well known to be highly effective for producing pre-trained linguistic transformer models [9, 25, 38, 42]. However, due to aforementioned limitations on token representations for continuous attributes, adapting transformer-based masked data modelling for computer vision tasks has largely focused on masking schemes. Discretised tokens

for visual data is a proposed workaround [3, 44]; however, the information loss of tokenisation renders this technique unfeasible for precise mediums like motion data. Masked auto-encoders assign mask tokens to an encoded latent space [16, 44], and adopt them to masked sequences at the decoder level. This allows the transformer encoder to learn solely from known tokens; however, the monolithic definition of masked tokens results in a discontinuous sequence representation in latent space.

Additionally, masked convolutional networks have been a consistently effective approach for image-based data. The popular U-Net structure of convolutional models [33] involves a data bottleneck, which elicits behaviours for ignoring masked image regions [17, 39, 46]. However, the pooling technique of convolutional models restrains the learned features to a fixed scale, and struggles to extrapolate global features with increased data scales, e.g. higher image resolutions or longer videos. Conversely, transformer-based methods are particularly effective with long-range feature learning, and as such, we believe them to be the more suitable for keyframe-guided sequential motion learning.

3. Methodology

We formulate the motion interpolation task as a data imputation problem from a sparse representation. The primary objective of our solution is to define motion sequence representations as a function of motion keyframes, which we learn using a transformer-based model. As shown in Figure 2, our solution comprises of three cooperative stages to learn a latent motion manifold using transformers to convert a sequence $K = \{x_0, x_{t_1}, x_{t_2}, ..., x_{N-1}\}$ at frames $T_K = \{0, t_1, t_2, ..., N-1\}$ into a complete motion sequence $Y = \{y_0, y_1, y_2, ..., y_{N-1}\}$ with temporal indices $T = \{0, 1, 2, ..., N-1\}$ (i.e. $T_K \subset T$):

- Stage-I, a **keyframe encoding stage** formulates the keyframe sequence K into keyframe context tokens $\Phi^{\text{key}}(K) = \{\phi_0, \phi_{t_1}, \phi_{t_2}, ..., \phi_{N-1}\}$, which serves as the motion encoding for further modelling.

- Stage-II, an **intermediate token generation stage** maps the temporal indices $t \in T \backslash T_K$ into intermediate tokens $\Phi^{\text{imd}}(t|\Phi^{\text{key}}(K))$ with the guidance of keyframe context, which constrains the latent space to obtain an implicit motion manifold.

- Stage-III, a **motion synthesis stage** injects the intermediate tokens between the keyframe tokens $\Phi^{\text{key}}(K)$, and decodes the resulting token sequence with a transformer Φ^{syn} to derive a motion sequence $\hat{Y} = \{\hat{y}_0, \hat{y}_1, ..., \hat{y}_{N-1}\}$ as an estimation of Y.
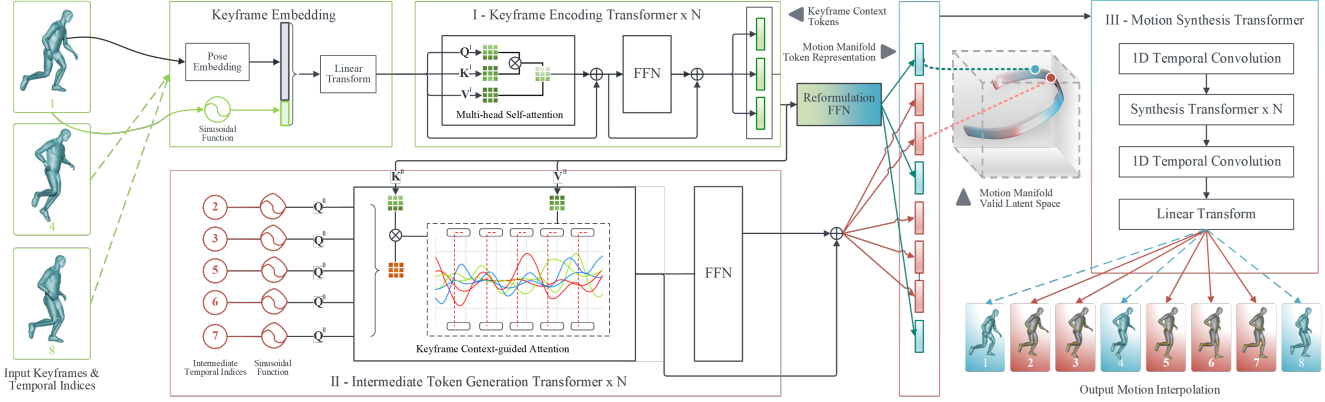
Figure 2. Overview of our transformer architecture and its three main components: (I) Encoding transformer Φ^{key}, (II) Intermediate token generation transformer Φ^{imd}, and (III) Motion synthesis transformer Φ^{syn}.

3.1. Keyframe poses to motion sequence context

Our pose representations are comprised of seven elements per joint: $P_t \in \mathbb{R}^{J \times 3}$ values for global 3D positions produced through forward kinematics (FK), and $q_t \in \mathbb{R}^{J \times 4}$ values for unit quaternion representations of local rotations, that is, $x_t = [P_t, q_t] \in \mathbb{R}^{J \times 7}$, where J represents the number of joints. In motion capture, local 3D positions for each joint are generally constant, determined by its positional offsets from its parent joint, and do not require explicit representation for the input. The sole exception to this is the root joint, where the global position is the local position.

In Stage-I, we encode the keyframe sequence K into a learned keyframe context token representation $\Phi^{\text{key}}(K)$. It is used as a feature map for both the intermediate token generation and motion synthesis stages. First, we project the pose data x_t, $t \in T_K$, of each keyframe into a pose embedding vector $x'_t \in \mathbb{R}^d$ using a linear layer, where d is the token embedding dimension. Next, we adopt sinusoidal positional encoding (PE) PE_{pos} [42] of n-dimension to x'_t. Unlike natural language processing (NLP) approaches, we do not add PE to our token representations, but instead concatenate a fixed-length PE for two key reasons:

- PE acts as a sliding binary vector, and thus can represent 2^n positions using n elements. For our task, we set n to 16, allowing our PE vector to produce 2^{16} positions, which is sufficient for our purposes.

- Additive PE introduces minor disruptions in token representations. While the discrete data of NLP can benefit from slight variations in token representations, it acts as a hindrance in continuous data due to the necessity of precision.

3.2. Sequence-level re-centering normalisation

Our Stage-I transformer Φ^{key} includes a number of layers based on multi-head self-attentions, each followed by GELU-activated feed-forward networks (FFNs). We replace all instances of LayerNorm the transformer encoders with RMSNorm [47], which does not involve a re-centering step. This avoids the token-level re-centering present in LayerNorm, which we observed to be detrimental for our regression tasks. We believe that this is due to feature biases that are present in and crucial to continuous attributes, e.g. a pose with a high root position values will result in all joints having similarly high position values. Therefore, we introduce a sequence level re-centering scheme for normalisation at the transformer input-level only, re-centreing the root positions of the input based on the mean root positions of all keyframes in the input sequence.

3.3. Motion manifold with context-guided attention

In Stage-II, our intermediate token generation transformer Φ^{imd} aims to learn motion manifolds navigable by temporal indices $t \in \mathbb{N}$ by the guidance of the keyframing context from $\Phi^{\text{key}}(K)$. In detail, we accomplish this through a keyframe context-guided attention mechanism, where the attention derives key and value mappings exclusively from linear transformations of $\Phi^{\text{key}}(K)$. For each intermediate token, the query is simply a sinusoidal embedding of the token's temporal position.

We constrain our manifold implicitly using two mechanisms. Firstly, the intermediate tokens are entirely sourced as a product of $\Phi^{\text{key}}(K)$ value transformations, which inherently limits the range of latent representations in the manifold. Secondly, the 1D convolutional layers in Stage-III entail feature dependencies between temporally adjacent tokens, which enables the disparately obtained Φ^{key} and Φ^{imd} tokens to converge towards coordination.

3.4. Interpolated motion synthesis

In Stage-III, a transformer Φ^{syn} is introduced to take the intermediate token representations m_t and estimate the

complete motion sequence $\hat{Y}$. Since the primary role of keyframe context tokens $\Phi^{\text{key}}(K)$ is to derive the intermediate token embeddings, an additional projection is performed using a single FFN $\text{FFN}(\Phi^{\text{key}}(K))$ to reformulate the keyframing context tokens into keyframe tokens that adhere to the motion manifold. The intermediate tokens obtained from Φ^{imd} are ready to be used directly. To this end, we can construct the resulting motion manifold $\hat{M}$ with a token sequence $\{\hat{m}_0, \hat{m}_1, ..., \hat{m}_{N-1}\}$ as follows:

$$\hat{m}_t = \begin{cases} \text{FFN}(\Phi^{\text{key}}(K)_t), & \text{if } t \in K \\ m_t, & \text{otherwise} \end{cases} \quad (1)$$

where $\Phi^{\text{key}}(K)_t$ is the context token representation for the keyframe of index t in the sequence.

Before applying the transformer Φ^{syn}, we feed the tokens of $\hat{M}$ through a 1D convolutional layer of kernel size 3. For each layer of Φ^{syn}, a self-attention function is followed by a FFN. The output token sequence of Φ^{syn} is fed through another 1D convolutional layer before the final linear projection. The final output $\hat{Y}_t$ consists of a root position estimation $\hat{p}_{t,0} \in \mathbb{R}^3$ of $p_{t,0}$ and local quaternions $\hat{q}_t \in \mathbb{R}^{J \times 4}$. Note that $p_{t,0}$ and q_t jointly can be used to compute the global position P_t and rotation Q_t information by FK.

3.5. Loss functions

The stages in our method are trained jointly in an end-to-end manner using a set of loss functions. To determine the loss for a motion sequence of length N, we use ℓ_1 distance for the following features obtained from $\hat{Y}$ and Y:

- **Local/root position loss**: With predicted and real coordinate values of the root joint at the t-th frame as $\hat{p}_{t,0} \in \mathbb{R}^3$ and $p_{t,0} \in \mathbb{R}^3$ respectively,

$$L_{root} = \frac{1}{N} \sum_{t=0}^{N-1} ||\hat{p}_{t,0} - p_{t,0}||_1. \quad (2)$$

- **Local rotation loss**: With predicted (pre-normalised) and expected (unit-normalised) quaternion values of all joints at frame t as $\hat{q}_{t,j} \in \mathbb{R}^4$ and $q_{t,j} \in \mathbb{R}^4$ respectively,

$$L_{quat} = \frac{1}{NJ} \sum_{t=0}^{N-1} \sum_{j=0}^{J-1} ||\hat{q}_{t,j} - q_{t,j}||_1. \quad (3)$$

- **Global position loss**: With predicted and real FK-derived coordinate values of all joints at frame t as $\hat{P}_t \in \mathbb{R}^{J \times 3}$ and $P_t \in \mathbb{R}^{J \times 3}$ respectively,

$$L_{FK_p} = \frac{1}{NJ} \sum_{t=0}^{N-1} ||\hat{P}_t - P_t||_1. \quad (4)$$

- **Global rotation loss**: With predicted and real FK-derived quaternion values of all joints at frame t as $\hat{Q}_t \in \mathbb{R}^{J \times 4}$ and $Q_t \in \mathbb{R}^{J \times 4}$ respectively,

$$L_{FK_q} = \frac{1}{NJ} \sum_{t=0}^{N-1} ||\hat{Q}_t - Q_t||_1. \quad (5)$$

Although quaternions are unit-normalised in practice, we found that calculating L_{quat} with non-normalised predictions resulted in improved gradient stability during the training process and, in turn, greater training convergence.

In summary, our training loss function L is as follows:

$$L = \alpha_l(L_{root} + L_{quat}) + \alpha_g(L_{FK_p} + L_{FK_q}), \quad (6)$$

where α_l and α_g are local and global feature loss scaling parameters, respectively. The accuracy of local attributes is best prioritised over that of global attributes, since normalised quaternions remain in use for deriving global features, which lead to gradient instability [10, 15].

4. Experiments and results

4.1. Datasets and metrics

We benchmark our method in the motion interpolation task against both the state-of-the-art RNN-based network [15] and BERT-based network [10] in motion transition generation. To evaluate the effectiveness of each model, they are to complete the following motion interpolation task:

- **Input**: The model is provided with the keyframes K of an N-frame ground truth motion. While keyframes can be defined for any combination of frames, we place each keyframe evenly every 5, 15, or 30 frames for consistency, starting with the first frame, e.g., $K = \{x_0, x_5, x_{10}, ..., x_{|K|}\}$ for 5-frame intervals.

- **Expected output**: Each model is to output an N-frame motion $\hat{Y} = \{\hat{y}_0, \hat{y}_1, ..., \hat{y}_{N-1}\}$ given the keyframes K. This output is compared with the ground truth with the L2P and L2Q metrics used in state-of-the-art comparisons [10, 15]. These metrics measure the average ℓ_2 errors of all positional and rotational attributes respectively for each pose. In addition, we apply the Normalised Power Spectrum Similarity (NPSS) [12] metric to measure visual similarities between the estimated and actual motion outputs.

We source our motions for both training and evaluation from the Ubisoft La Forge Animation (LaFAN1) dataset [15] and the Carnegie-Mellon University Motion Capture (CMU Mocap) dataset [2]. The CMU Mocap motions are resampled from their original 120 frames per second (FPS) down to 30 FPS considering the computational cost, and to match the frame rate of the LaFAN1 dataset. While LaFAN1 focuses largely on motions with visibly dynamic details such as locomotion and sports actions, CMU Mocap provides a larger variety of motions, many of which exhibit more minute details and ambiguous motion trajectories. From a data-level perspective, this suggests that root

---

[2]Dataset available at: http://mocap.cs.cmu.edu/

Table 1 — CMU:acrobatics, CMU:basketball, CMU:golf

Category	CMU:acrobatics									CMU:basketball									CMU:golf								
	L2P			L2Q			NPSS			L2P			L2Q			NPSS			L2P			L2Q			NPSS		
Interval	5	15	30	5	15	30	5	15	30	5	15	30	5	15	30	5	15	30	5	15	30	5	15	30	5	15	30
LERP	0.234	0.899	1.718	0.308	1.052	1.598	0.183	0.854	1.856	0.122	0.544	1.051	0.182	0.695	1.044	0.078	0.352	1.078	**0.029**	0.181	0.509	**0.044**	0.196	0.473	**0.060**	0.396	1.296
BERT	0.353	0.947	1.744	0.516	1.211	1.712	0.249	0.886	1.865	0.165	0.547	1.043	0.221	0.734	1.075	0.070	0.296	0.972	0.066	0.181	0.511	0.071	0.197	0.475	0.078	0.280	1.187
Δ-interpolator	**0.189**	0.727	1.395	**0.270**	0.814	1.342	**0.152**	0.642	1.560	0.140	0.594	1.212	0.206	0.717	1.089	0.089	0.344	1.023	0.056	0.193	0.511	0.085	0.216	0.487	0.078	0.390	1.276
TG$_{complete}$	0.400	0.982	1.624	0.499	1.177	1.772	0.297	1.004	2.054	0.181	0.482	0.921	0.255	0.718	1.116	0.106	0.291	0.769	0.113	0.222	0.461	0.121	0.244	0.458	0.110	0.208	0.635
MAE	0.295	0.697	1.156	0.730	1.219	1.511	0.444	0.878	1.444	0.163	0.405	0.739	0.252	0.620	0.915	0.095	0.255	0.855	0.080	0.139	0.214	0.110	0.185	0.267	0.114	0.177	0.316
Ours	0.217	**0.471**	**0.791**	0.328	**0.643**	**0.931**	0.167	**0.410**	**0.769**	**0.101**	**0.274**	**0.456**	**0.165**	**0.473**	**0.668**	**0.054**	**0.160**	**0.329**	0.048	**0.083**	**0.116**	0.083	**0.138**	**0.182**	0.081	**0.104**	**0.155**

Table 1 — CMU:salsa, CMU:swim, CMU:walk/run

Category	CMU:salsa									CMU:swim									CMU:walk/run								
	L2P			L2Q			NPSS			L2P			L2Q			NPSS			L2P			L2Q			NPSS		
Interval	5	15	30	5	15	30	5	15	30	5	15	30	5	15	30	5	15	30	5	15	30	5	15	30	5	15	30
LERP	0.272	1.091	2.087	**0.362**	1.265	2.193	0.212	0.788	2.114	**0.114**	0.475	0.925	**0.169**	0.630	1.167	0.205	0.873	1.242	**0.070**	0.366	0.720	**0.095**	0.327	0.512	**0.028**	0.133	0.506
BERT	0.337	1.131	2.164	0.509	1.395	2.315	0.233	0.800	2.111	0.184	0.506	0.958	0.256	0.697	1.214	**0.175**	0.771	1.213	0.129	0.346	0.516	0.124	0.355	0.565	0.040	0.115	0.458
Δ-interpolator	0.287	1.124	2.176	0.391	1.297	2.183	0.217	0.780	2.093	0.125	0.498	0.938	0.193	0.650	1.145	0.242	0.650	1.369	0.083	0.375	0.757	0.124	0.355	0.565	0.049	0.152	0.616
TG$_{complete}$	0.348	1.117	2.136	0.485	1.373	2.270	0.193	0.737	2.120	0.234	0.533	0.924	0.321	0.732	1.231	0.268	0.781	1.253	0.138	0.370	0.700	0.167	0.368	0.568	0.066	0.135	0.369
MAE	0.279	0.864	1.930	0.595	1.188	2.058	0.368	0.904	2.623	0.281	0.508	0.820	0.662	0.903	1.222	0.612	0.893	1.536	0.118	0.309	0.554	0.154	0.315	0.459	0.068	0.138	0.336
Ours	**0.205**	**0.662**	**1.519**	0.393	**0.823**	**1.417**	**0.186**	**0.588**	**1.707**	0.134	**0.307**	**0.583**	0.186	**0.431**	**0.945**				0.076	**0.188**	**0.400**	0.103	**0.222**	0.362	0.044	**0.091**	**0.247**

Table 1 — LaFAN1:crawl, LaFAN1:dance, LaFAN1:get up

Category	LaFAN1:crawl									LaFAN1:dance									LaFAN1:get up								
	L2P			L2Q			NPSS			L2P			L2Q			NPSS			L2P			L2Q			NPSS		
Interval	5	15	30	5	15	30	5	15	30	5	15	30	5	15	30	5	15	30	5	15	30	5	15	30	5	15	30
LERP	**0.088**	0.481	1.045	**0.093**	0.365	0.671	**0.082**	0.512	1.541	0.141	0.683	1.287	**0.150**	0.583	0.984	0.111	0.516	1.232	**0.095**	0.505	1.118	**0.104**	0.402	0.704	**0.072**	0.458	1.385
BERT	0.181	0.522	1.075	0.166	0.403	0.699	0.143	0.422	1.432	0.209	0.717	1.311	0.212	0.622	1.012	0.100	0.467	1.158	0.190	0.545	1.147	0.182	0.446	0.734	0.099	0.394	1.290
Δ-interpolator	0.097	0.437	0.970	0.106	0.351	0.660	0.120	0.518	1.497	0.164	0.703	1.308	0.178	0.607	1.014	0.139	0.507	1.255	0.126	0.584	1.279	0.136	0.459	0.797	0.098	0.442	1.345
TG$_{complete}$	0.203	0.500	0.996	0.180	0.413	0.719	0.207	0.506	1.309	0.244	0.729	1.332	0.232	0.630	1.024	0.130	0.455	1.077	0.200	0.493	0.960	0.186	0.434	0.726	0.153	0.439	1.051
MAE	0.244	0.497	0.909	0.243	0.408	0.633	0.321	0.501	1.141	0.225	0.640	1.069	0.251	0.569	0.871	0.189	0.468	0.961	0.263	0.513	0.913	0.269	0.446	0.668	0.276	0.443	0.891
Ours	0.115	**0.343**	**0.681**	0.136	**0.312**	**0.530**	0.127	**0.324**	**0.858**	**0.128**	**0.506**	**0.917**	0.168	**0.494**	**0.777**	**0.087**	**0.357**	**0.829**	0.135	**0.340**	**0.645**	0.150	**0.329**	**0.543**	0.105	**0.265**	**0.664**

Table 1 — LaFAN1:jump/hop, LaFAN1:obstacles, LaFAN1:walk

Category	LaFAN1:jump/hop									LaFAN1:obstacles									LaFAN1:walk								
	L2P			L2Q			NPSS			L2P			L2Q			NPSS			L2P			L2Q			NPSS		
Interval	5	15	30	5	15	30	5	15	30	5	15	30	5	15	30	5	15	30	5	15	30	5	15	30	5	15	30
LERP	0.178	0.837	1.327	**0.146**	0.544	0.779	0.073	0.304	0.642	0.153	0.796	1.616	**0.126**	0.497	0.818	0.049	0.256	0.757	0.124	0.669	1.451	**0.117**	0.467	0.793	**0.052**	0.265	0.950
BERT	0.277	0.886	1.331	0.222	0.584	0.803	0.092	0.280	0.602	0.227	0.830	1.629	0.183	0.529	0.834	0.062	0.230	0.696	0.182	0.682	1.442	0.169	0.492	0.801	0.059	0.228	0.882
Δ-interpolator	0.173	0.777	1.331	0.153	0.520	0.811	0.094	0.301	0.727	0.150	0.732	1.484	0.134	0.486	0.805	0.070	0.271	0.879	0.134	0.647	1.389	0.135	0.471	0.799	0.067	0.242	0.872
TG$_{complete}$	0.299	0.854	1.391	0.244	0.608	0.923	0.136	0.401	0.628	0.247	0.640	1.246	0.205	0.475	0.770	0.108	0.290	0.643	0.217	0.570	1.148	0.188	0.446	0.742	0.092	0.248	0.566
MAE	0.275	0.737	1.123	0.262	0.536	0.757	0.111	0.299	0.585	0.263	0.574	1.077	0.250	0.426	0.643	0.118	0.228	0.466	0.219	0.525	0.912	0.224	0.414	0.597	0.135	0.261	0.578
Ours	**0.151**	**0.557**	**0.940**	0.163	**0.455**	**0.677**	**0.052**	**0.216**	**0.450**	**0.130**	**0.366**	**0.789**	0.138	**0.307**	**0.510**	**0.048**	**0.129**	**0.335**	**0.111**	**0.322**	**0.639**	0.128	**0.284**	**0.456**	0.054	**0.139**	**0.344**

Table 1. Comparison of average L2P, L2Q, and NPSS performance on motion samples from various categories, each with 121 frames in length. Lower values indicate more accurate predictions. The top performer of each test is highlighted in **bold**.

position accuracy is more important in the LaFAN1 dataset compared to the CMU Mocap dataset. We employ both datasets to demonstrate our method's ability to adapt with different levels of motion dynamics.

For our model, the positional data of each motion sample is recentred around the XYZ means of the keyframed root positions. Given that unit quaternion values are restricted to a range of $[-1, 1]$, while positional values can be of any scalar value, we rescale positional data such that $L_{root} \approx L_{quat}$ with initial model weights. During training, we randomly select between $[\lfloor \frac{|Y|}{24} \rfloor, \lfloor \frac{|Y|}{4} \rfloor]$ keyframes for each sampled motion Y, with the first and last frames being stipulated as keyframes. Each motion sample batch has a random length of $|Y| \in [72, 144]$.

4.2. Implementation details

We implement our method with 8 layers for each transformer, with a token embedding size of $d = 512$, and an FFN size of $4 \times d$. Each multi-head attention layer is split between 8 attention heads. Since our method relies on coordination between Stage-I and Stage-II, the training stability of deeper models benefits greatly from larger batch sizes. For our 8-layer setting, we found that a batch size of 64 motions per epoch is sufficient for convergence.

We train our model using the Adam optimiser [20] for 50,000 epochs with a scheduled learning rate. Specifically, we employ both a warm-up and decay strategy for our learn-

	Φ^{syn} Key & Intermediate Input	Layers	Norm	PE mode	L2P	L2Q	NPSS
(a)	N/A - No Φ^{syn}	8	Seq-RC	concat	0.665	0.699	0.640
(b)	Φ^{key} & Φ^{imd}	8	LayerNorm	concat	0.539	0.620	0.581
(c)	Φ^{key} & Φ^{imd}	8	Seq-RC	additive	0.587	0.661	0.565
(d)	keyframe embeddings & Φ^{imd}	2	Seq-RC	concat	0.684	0.711	0.675
		4			0.610	0.642	0.532
		8			*Divergence*		
		12			*Divergence*		
	Φ^{key} & Φ^{imd}	2	Seq-RC	concat	0.652	0.691	0.609
		4			0.534	0.608	0.501
		8			0.438	0.530	0.384
		12			**0.416**	**0.515**	**0.380**

Table 2. Ablation study of our architectural features: (a) Stage-III manifold refinement (b) Seq-RC, (c) concatenated PE, (d) bridging Φ^{key} into Φ^{syn}, convergability & depth comparison.

ing rate using the following strategy [42]:

$$lr(e) = 4\text{e-}4 \times \min(e^{-\frac{1}{2}}, e \times 1000^{-\frac{3}{2}}), \qquad (7)$$

where e represents the number of current training epoch. In addition, we linearly scale $\alpha_g \in [0, 1]$ for 1,000 epochs after warm-up, in order to avoid conflicting gradients caused by ambiguous quaternion polarity when backpropagating through FK functions, i.e. $FK(Q) = FK(-Q)$ for any set of joint rotation quaternions Q. We set $\alpha_l = 1$ throughout the training process.

4.3. Comparison to the state-of-the-art

We benchmark the performance of our method against state-of-the-art models by evaluating their L2P, L2Q, and NPSS metrics with testing datasets. Table 1 compares
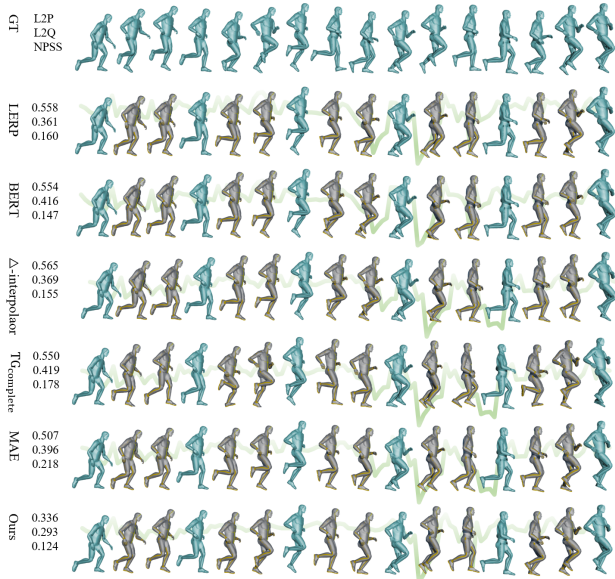
Figure 3. An example of motion interpolation by each of the tested methods, compared with the ground truth motion (first row). The green curves indicate the offsets regarding the ℓ_1 distance between the interpolation and ground truth manifolds. Less turbulent offsets indicate more visually similar motion predictions.

$|Y|$	# Params	31	61	121
LERP	-	0.0330	0.0360	0.0370
TG$_{complete}$	15.6M	0.5001	1.0272	1.9774
BERT	29.3M	0.0541	0.0570	0.0596
MAE	54.9M	0.0755	0.0793	0.0820
Ours	83.2M	0.0793	0.0830	0.0850

Table 3. Inference time in seconds and parameter count for different motion lengths $|Y|$. Keyframes of each evaluation were evenly placed every 15 frames, starting from the first frame.
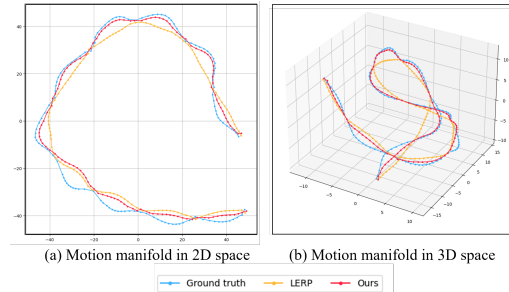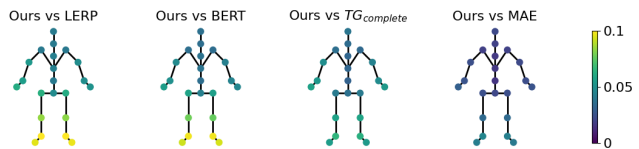

(a) Motion manifold in 2D space (b) Motion manifold in 3D space

— Ground truth   — LERP   — Ours

Figure 4. Sample motion manifolds obtained by t-SNE.

Ours vs LERP   Ours vs BERT   Ours vs $TG_{complete}$ Ours vs MAE

Figure 5. Performance improvement of our architecture by joints in L2P + L2Q, compared to existing methods.

the performance of our architecture against the BERT-based motion in-painting transformer [10], the encoder-decoder-based Δ-interpolator [31], the RNN-based approach TG$_{complete}$ [15], and the masked auto-encoder (MAE) architecture [16].

The quantitative performance of our model is greatly improved over all existing methods, as well as LERP, in a large majority of keyframing scenarios. The performance improvement of our method compared to LERP increases with the length of keyframe intervals, as using learning based methods provides the opportunity to reconstruct non-linear motion details. Note that for a short keyframe interval, a linear estimation ($f(x + \Delta x) = f(x) + \hat{f}'(x)\Delta x + o(\Delta x)$) of a continuous motion (function) can be relatively accurate, which explains why similar performance is found between LERP and our method for the 5-frame interval setting. However, other existing methods are significantly worse than LERP. Our model notably outperforms both TG$_{complete}$ and MAE with its single token mask in every scenario. Thus, a clear motion interpolation improvement can be observed from our decoupling strategy with motion manifold technique, compared to the RNN model.

The BERT-based method [10] exhibits a clear disadvantage in its performance due to its reliance on LERP-based input mask tokens. By deriving the mask token embeddings from a sub-optimal estimation, self-attention mechanisms tend to converge towards reproducing the input token rather than composing more realistic poses, as it is close to a trivial local minimum to learn. Consequently, such models never learn to fully consider the keyframe tokens as their main source of information. Figure 3 highlights the near-identical latent manifolds of LERP output and BERT-based evaluation. We observe a similar behaviour with the Δ-interpolator model, where LERP-based transformations are applied as a post-processing step [31]. While its Δ-mode strategy allows the model to perform marginal improvements over LERP more frequently, it is still heavily reliant on the performance of LERP, which does not bode well with longer keyframe intervals. To this end, we can deduce that the realistic interpolation can be difficult with LERP-reliant solutions. Conversely, our manifold learning approach fully considers the continuous joint positions and rotations of the input keyframes, and is able to converge upon a significantly more optimal solution.

Table 2 documents an ablation study for each our architecture's contributions. Major improvements to the architecture's L2P, L2Q, and NPSS performance can be observed for the inclusion of manifold self-attention in Stage III, the replacement of LayerNorm with our sequence-level re-centering normalisation scheme, and concatenation over addition of PE. It should be noted that the Stages I and

	L2P			L2Q			NPSS		
Interval	5	15	30	5	15	30	5	15	30
LERP	0.35	1.28	2.46	0.22	0.66	1.17	0.0021	0.0430	0.2663
TG$_{complete}$	0.22	0.64	1.25	0.17	0.45	0.68	0.0019	0.0247	0.1298
BERT	0.22	0.60	1.14	0.15	0.38	0.60	0.0016	0.0251	0.1270
Δ-interpolator	0.16	0.53	1.05	0.12	0.33	0.59	0.0015	0.0238	0.1272
Ours	0.30	0.71	1.26	0.21	0.40	0.63	0.0019	0.0284	0.1393

Table 4. Motion completion performance of our method, based on the Harvey et al. (2020) [15] setup.

II only variant of our model (i.e., (a) in Table 2) is structurally identical to the Δ-interpolator model with Δ-mode disabled [31]. In addition, we demonstrate the use of Φ_{key} token representations over separately trained keyframe embeddings in Stage-III, which leads to improved convergibility of deeper architecture settings. We further demonstrate the importance of such deeper settings, which provide a significant boost to our model's evaluation accuracy.

Figure 4 visualises the latent motion manifolds in 2D and 3D spaces for our method, LERP and ground truth using t-SNE analysis. The manifold of our method is obtained from the inputs of Stage-III, and LERP and ground truth are from the pose data. It can be observed that the lower-dimensional curves (i.e., manifolds) represent the motion of higher-dimensional in a smooth manner, and ours is very close to the one associated with the ground truth, compared with LERP. This indicates the superiority of our method to derive a high-quality latent motion space. Figure 3 illustrates an example of hopping motion interpolated by different methods with quantitative metrics. The ground truth data is reduced to a motion manifold with t-SNE. The offset of the manifold from each interpolation method compared to the ground truth one is obtained by ℓ_1 distance for visualization. Particularly, the offset values are enlarged for observation purpose. Our method gains the best motion manifold with the least offset from the ground truth manifold.

Figure 5 dissects the L2P and L2Q improvements of our method into individual joints. We can clearly observe that the main improvements of our method over LERP exist within the global positions and rotations of foot joints, whereas improvements are more widely spread compared to the MAE and RNN-based methods. On average, our approach brings a L2P and L2Q benefit to all joints.

4.4. Inference latency

The inference time of different motion interpolation methods was evaluated in our experiments, as the visual latency of keyframe adjustments is important for efficient animation work, as well as real-time applications. We implemented these methods with PyTorch on an AMD Ryzen 9 3950X processor and an NVIDIA GeForce RTX 3090 GPU. Table 3 shows that the parallelism provided by the transformer is highly beneficial to our method when interpolating complete motion sequences. Our method shows a similar order of time complexity to the LERP method, being consistently around $2.5\times$ LERP inference time, and significantly faster than the RNN-based approach. In addition, our method is stable in terms of the inference time for different sequence lengths, whilst the RNN-based approach shows a significant increasing latency from 31-frame sequence to 121-frame sequence.

4.5. Extension to motion completion

Though our model is designed for sparse keyframe interpolation, we can additionally perform motion completion as it can be defined as a specifically constructed keyframe set. Table 4 compares the performance of our model against linear interpolation and the state-of-the-art models for motion completion. With the benefit of the motion context, our model outperforms LERP, but not to the efficacy of the RNN-based [15] and transformer-based [10,31] models.

4.6. Limitations and future work

One limitation of our approach is that its maximum motion length is limited by the length of its training samples. Unlike most transformer-based solutions that can trivially employ relative positional encodings [5,35], our method relies on continuous positional vectors, such as sinusoidal encodings, for its manifold representations, and thus cannot employ the same model to accept inputs of arbitrary length. Further research for a compatible relative position representation would allow our approach to be seamlessly applied in keyframed animation workflows for longer sequences.

5. Conclusion

This paper presents a three-stage transformer-based motion interpolation method. We begin by producing learned motion keyframe context tokens in Stage-I. With context-guided attention, Stage-II generates embeddings for intermediate tokens by inferencing an implicitly constrained latent motion manifold with the guidance of the keyframe context tokens. Stage-III takes both the keyframe tokens and intermediate tokens to compose the interpolated motion sequence. In addition, we introduce a novel sequence-level re-centreing technique to address the feature biases that are more prevalent in sequences of continuous attributes. We demonstrate that the superior interpolation accuracy of our approach compared with existing RNN and masked transformer methods. As our architecture is designed for any masked sequence-to-sequence task with continuous attributes, we believe that our architecture's applications extend beyond motion interpolation.

Acknowledgment

References

[1] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016. 2

[2] Yutong Bai, Jieru Mei, Alan L Yuille, and Cihang Xie. Are transformers more robust than CNNs? *Advances in Neural Information Processing Systems*, 34:26831–26843, 2021. 3

[3] Hangbo Bao, Li Dong, and Furu Wei. Beit: Bert pre-training of image transformers. *arXiv preprint arXiv:2106.08254*, 2021. 3

[4] Didac Suris Coll-Vinent and Carl Vondrick. Representing spatial trajectories as distributions. In *Advances in Neural Information Processing Systems*. 1

[5] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860*, 2019. 8

[6] Lingwei Dang, Yongwei Nie, Chengjiang Long, Qing Zhang, and Guiqing Li. Msr-gcn: Multi-scale residual graph convolution networks for human motion prediction. In *IEEE International Conference on Computer Vision*, pages 11447–11456, 2021. 1

[7] Lingwei Dang, Yongwei Nie, Chengjiang Long, Qing Zhang, and Guiqing Li. Diverse human motion prediction via gumbel-softmax sampling from an auxiliary space. In *ACM International Conference on Multimedia*, pages 5162–5171, 2022. 3

[8] Bruno Degardin, João Neves, Vasco Lopes, João Brito, Ehsan Yaghoubi, and Hugo Proença. Generative adversarial graph convolutional networks for human action synthesis. In *IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1150–1159, 2022. 3

[9] Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. Unified language model pre-training for natural language understanding and generation. *Advances in Neural Information Processing Systems*, 32, 2019. 3

[10] Yinglin Duan, Yue Lin, Zhengxia Zou, Yi Yuan, Zhehui Qian, and Bohan Zhang. A unified framework for real time motion completion. In *AAAI Conference on Artificial Intelligence*, volume 36, pages 4459–4467, 2022. 1, 2, 3, 5, 7, 8

[11] Thomas Frank and Ollie Johnston. *Disney Animation: The Illusion of Life*. Abbeville Publishing Group, New York, 1981. 1

[12] Anand Gopalakrishnan, Ankur Mali, Dan Kifer, Lee Giles, and Alexander G Ororbia. A neural temporal model for human motion prediction. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12116–12125, 2019. 5

[13] Xiao Guo and Jongmoo Choi. Human motion prediction via learning local structure representations and temporal dependencies. In *AAAI Conference on Artificial Intelligence*, volume 33, pages 2580–2587, 2019. 3

[14] Félix G Harvey and Christopher Pal. Recurrent transition networks for character locomotion. In *ACM SIGGRAPH Asia Technical Briefs*, pages 1–4. 2018. 1

[15] Félix G Harvey, Mike Yurick, Derek Nowrouzezahrai, and Christopher Pal. Robust motion in-betweening. *ACM Transactions on Graphics*, 39(4):60–1, 2020. 1, 3, 5, 7, 8

[16] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16000–16009, 2022. 2, 3, 7

[17] Satoshi Iizuka, Edgar Simo-Serra, and Hiroshi Ishikawa. Globally and locally consistent image completion. *ACM Transactions on Graphics*, 36(4):1–14, 2017. 3

[18] Mehdi Jafari and Habib Molaei. Spherical linear interpolation and bézier curves. *General Scientific Researches*, 2(1):13–17, 2014. 2

[19] Manuel Kaufmann, Emre Aksan, Jie Song, Fabrizio Pece, Remo Ziegler, and Otmar Hilliges. Convolutional autoencoders for human motion infilling. In *International Conference on 3D Vision*, pages 918–927. IEEE, 2020. 3

[20] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 6

[21] Lucas Kovar, Michael Gleicher, and Frédéric Pighin. Motion graphs. In *ACM SIGGRAPH Classes*, pages 1–10. 2008. 2

[22] Jogendra Nath Kundu, Maharshi Gor, and R Venkatesh Babu. Bihmp-gan: Bidirectional 3d human motion prediction gan. In *AAAI Conference on Artificial Intelligence*, volume 33, pages 8553–8560, 2019. 3

[23] Andreas M Lehrmann, Peter V Gehler, and Sebastian Nowozin. Efficient nonlinear markov models for human motion. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1314–1321, 2014. 2

[24] Liyuan Liu, Xiaodong Liu, Jianfeng Gao, Weizhu Chen, and Jiawei Han. Understanding the difficulty of training transformers. In *2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020*, pages 5747–5763. Association for Computational Linguistics, 2020. 2, 3

[25] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019. 2, 3

[26] Yimeng Liu and Misha Sra. Motion improvisation: 3d human motion synthesis with a transformer. In *Annual ACM Symposium on User Interface Software and Technology*, pages 26–28, 2021. 1, 3

[27] Zhouyong Liu, Shun Luo, Wubin Li, Jingben Lu, Yufan Wu, Shilei Sun, Chunguo Li, and Luxi Yang. Convtransformer: A convolutional transformer network for video frame synthesis. *arXiv preprint arXiv:2011.10185*, 2020. 3

[28] Liying Lu, Ruizheng Wu, Huaijia Lin, Jiangbo Lu, and Jiaya Jia. Video frame interpolation with transformer. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3532–3542, 2022. 3

[29] Tiezheng Ma, Yongwei Nie, Chengjiang Long, Qing Zhang, and Guiqing Li. Progressively generating better initial guesses towards next stages for high-quality human motion prediction. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 6427–6436, 2022. 3

[30] Clinton Mo, Kun Hu, Shaohui Mei, Zebin Chen, and Zhiyong Wang. Keyframe extraction from motion capture sequences with graph based deep reinforcement learning. In *ACM International Conference on Multimedia*, pages 5194–5202, 2021. 1, 3

[31] Boris N Oreshkin, Antonios Valkanas, Félix G Harvey, Louis-Simon Ménard, Florent Bocquelet, and Mark J Coates. Motion inbetweening via deep δ-interpolator. *arXiv preprint arXiv:2201.06701*, 2022. 1, 2, 3, 7, 8

[32] Georgios Pavlakos, Vasileios Choutas, Nima Ghorbani, Timo Bolkart, Ahmed AA Osman, Dimitrios Tzionas, and Michael J Black. Expressive body capture: 3d hands, face, and body from a single image. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10975–10985, 2019. 3

[33] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015. 3

[34] Charles Rose, Brian Guenter, Bobby Bodenheimer, and Michael F Cohen. Efficient generation of motion transitions using spacetime constraints. In *Annual Conference on Computer Graphics and Interactive Techniques*, pages 147–154, 1996. 2

[35] Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. Self-attention with relative position representations. *arXiv preprint arXiv:1803.02155*, 2018. 8

[36] Zhihao Shi, Xiangyu Xu, Xiaohong Liu, Jun Chen, and Ming-Hsuan Yang. Video frame interpolation transformer. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17482–17491, 2022. 3

[37] Scott Sona Snibbe. A direct manipulation interface for 3D computer animation. In *Computer Graphics Forum*, volume 14, pages 271–283. Wiley Online Library, 1995. 2

[38] Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. Mass: Masked sequence to sequence pre-training for language generation. *arXiv preprint arXiv:1905.02450*, 2019. 1, 3

[39] Linsen Song, Jie Cao, Lingxiao Song, Yibo Hu, and Ran He. Geometry-aware face completion and editing. In *AAAI Conference on Artificial Intelligence*, volume 33, pages 2506–2513, 2019. 3

[40] Xiangjun Tang, He Wang, Bo Hu, Xu Gong, Ruifan Yi, Qilong Kou, and Xiaogang Jin. Real-time controllable motion transition for characters. *ACM Transactions on Graphics*, 41(4), 2022. 1, 3

[41] Garvita Tiwari, Dimitrije Antić, Jan Eric Lenssen, Nikolaos Sarafianos, Tony Tung, and Gerard Pons-Moll. Pose-ndf: Modeling human pose manifolds with neural distance fields. In *European Conference on Computer Vision*, pages 572–589. Springer, 2022. 3

[42] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in Neural Information Processing Systems*, 30, 2017. 1, 2, 3, 4, 6

[43] Lintao Wang, Kun Hu, Lei Bai, Yu Ding, Wanli Ouyang, and Zhiyong Wang. Multi-scale control signal-aware transformer for motion synthesis without phase. *arXiv preprint arXiv:2303.01685*, 2023. 1

[44] Rui Wang, Dongdong Chen, Zuxuan Wu, Yinpeng Chen, Xiyang Dai, Mengchen Liu, Yu-Gang Jiang, Luowei Zhou, and Lu Yuan. Bevt: Bert pretraining of video transformers. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14733–14743, 2022. 3

[45] Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tieyan Liu. On layer normalization in the transformer architecture. In *International Conference on Machine Learning*, pages 10524–10533. PMLR, 2020. 2, 3

[46] Jie Yang, Zhiquan Qi, and Yong Shi. Learning to incorporate structure knowledge for image inpainting. In *AAAI Conference on Artificial Intelligence*, volume 34, pages 12605–12612, 2020. 3

[47] Biao Zhang and Rico Sennrich. Root mean square layer normalization. *Advances in Neural Information Processing Systems*, 32, 2019. 2, 4

[48] Xinyi Zhang and Michiel van de Panne. Data-driven auto-completion for keyframe animation. In *Annual International Conference on Motion, Interaction, and Games*, pages 1–11, 2018. 3