ACM SIGGRAPH / Eurographics Symposium on Computer Animation 2024
M. Skouras and H. Wang
(Guest Editors)

COMPUTER GRAPHICS *forum*
Volume 43 (2024), Number 8

Long-term Motion In-betweening via Keyframe Prediction

Seokhyeon Hong[1] Haemin Kim[1] Kyungmin Cho[2] Junyong Noh[1]

[1]KAIST, Visual Media Lab
[2]Anigma Technologies

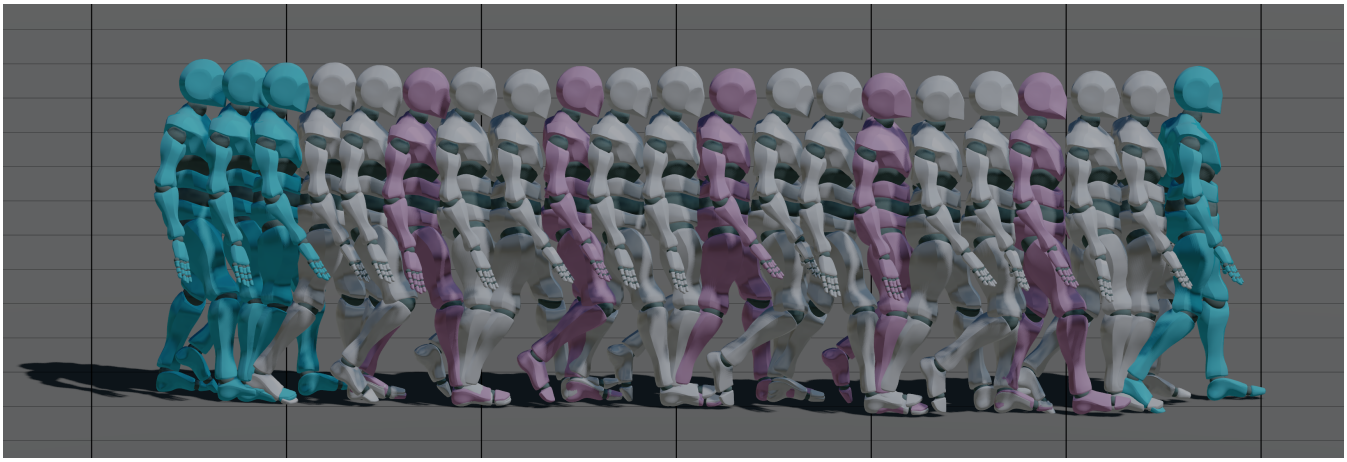


Figure 1: *Our method synthesizes long-term motion in-betweening in a two-stage manner. Given constrained frames (blue), keyframes (purple) are first predicted, and then the remaining transition frames (gray) are subsequently synthesized.*

Abstract

Motion in-betweening has emerged as a promising approach to enhance the efficiency of motion creation due to its flexibility and time performance. However, previous in-betweening methods are limited to generating short transitions due to growing pose ambiguity when the number of missing frames increases. This length-related constraint makes the optimization hard and it further causes another constraint on the target pose, limiting the degrees of freedom for artists to use. In this paper, we introduce a keyframe-driven approach that effectively solves the pose ambiguity problem, allowing robust in-betweening performance on various lengths of missing frames. To incorporate keyframe-driven motion synthesis, we introduce a keyframe score that measures the likelihood of a frame being used as a keyframe as well as an adaptive keyframe selection method that maintains appropriate temporal distances between resulting keyframes. Additionally, we employ phase manifolds to further resolve the pose ambiguity and incorporate trajectory conditions to guide the approximate movement of the character. Comprehensive evaluations, encompassing both quantitative and qualitative analyses, were conducted to compare our method with state-of-the-art in-betweening approaches across various transition lengths. The code for the paper is available at https://github.com/seokhyeonhong/long-mib

CCS Concepts
• ***Computing methodologies*** → ***Animation;***

1. Introduction

Creating lifelike and expressive 3D character animations is a challenging task due to the complexity and stochastic nature of motion data. While keyframing and motion capture have traditionally been popular methods for animation creation, they require laborious manual interventions by experts. To address this problem, motion in-betweening, which generates intermediate frames be-

tween given context frames at the beginning and a target frame at the end of a motion sequence, has emerged as an alternative approach [HP18, HYNP20, KBS*22, KAS*20, TWH*22, QZZ22]. By combining the advantages of keyframing and motion capture, motion in-betweening offers both flexibility and agility in the motion creation process. Specifically, like a keyframing method, it provides flexibility in motion creation and editing by allowing animators to freely set the context and target frames. In addition, compared to the traditional keyframing method, it dramatically reduces the creation time by generating motion frames from a much sparser set of keyframes leveraging a large motion capture dataset.

While previous in-betweening methods can generate natural motions, they are still difficult to utilize for practical use. As transition lengths extend, they often produce unnatural motions with inconsistent transition. This is because the stochasticity of motion data becomes bigger for longer transition frames, making the optimization hard. Furthermore, this length-related constraint causes another restriction on the target pose. Because the possible transition length range is limited, the target pose should be carefully selected to be structurally similar to the context frames to satisfy temporal coherence in such a short transition.

To address these issues, we propose a novel method for long-term motion in-betweening. Our method is based on two-stage hierarchical approach that predicts intermediate keyframes first given constrained frames, and subsequently synthesizes the entire motion sequence using these predicted keyframes as anchor points. As keyframes encapsulate salient moments in a motion sequence, they can serve as effective anchor points that alleviate the increasing complexity of transitions on longer frames. For the keyframe-driven motion synthesis, we introduce novel concepts called a *keyframe score* and an *adaptive keyframe selection*. The keyframe score is an indicator that measures the likelihood of a frame being used as a keyframe, and it is used to select the most probable frames that can represent the entire motion sequence during the motion synthesis process. For the even distribution of keyframes over time, adaptive keyframe selection method ensures proper temporal distances between adjacent keyframes while taking the importance encoded in the keyframe scores.

Additionally, we leverage phase manifolds learned by a periodic autoencoder (PAE) [SMK22] and trajectory features. As phase manifolds encode the relationship between time and pose, employing phase to motion synthesis disambiguates the pose given timing, and it leads to the improved naturalness of generated motions. Trajectory features describe an approximate path of the character motion and therefore they can be used to enhance user control over the generated motions.

In summary, our contributions can be summarized as follows:

- Long-term motion in-betweening in which the range of transition lengths can be elongated to alleviate the constraints on the transition length and target poses.
- Keyframe scores and adaptive keyframe selection to involve keyframe-driven synthesis for motion in-betweening.
- Utilization of auxiliary features that enable time-pose alignment and user editability.

2. Related Work

2.1. Motion In-betweening

Motion in-betweening is a conditional motion synthesis task that generates smooth and natural transitions conditioned on temporally sparse pose constraints. Early work on transition generation employed optimization techniques with spacetime constraints [RGBC96, WK88] and radial basis functions [RCB98, RISC01]. More sophisticated statistical models were also used in transition synthesis, such as maximum a posteriori [CH07, MCC09], non-linear Markov models [LGN14], geostatistical models [MK05], and Gaussian process [WFH07].

Along with the development of deep learning models, neural networks have become a promising approach for motion in-betweening. Pioneering work in neural motion in-betweening [HP18] introduced the use of recurrent neural networks (RNN) with space-time constraints. However, this method was constrained to generate transitions of fixed lengths. Subsequent strategies were introduced to address this limitation by allowing for transitions of varying lengths [HYNP20]. Tang et al. [TWH*22] further extended this work by utilizing a conditional variational autoencoder with a mixture of experts (MoE) decoder. Recent work [SSKS23, TWW*23] additionally used phase priors learned by a PAE [SMK22] as an auxiliary feature to disambiguate a pose given timing. Despite these advancements, the RNN-based architectures are known to struggle with long-range predictions due to their sequential dependency, making them unsuitable for long-term transition synthesis.

Convolutional neural networks (CNN) have also been used to synthesize all the missing and partially filled frames. Inspired by image inpainting studies, CNN-based approaches have represented motion sequences as image-like feature maps and train a CNN autoencoder to infill the entire motion sequence [KAS*20]. Additional training strategies, such as adversarial training [HGMN19, ZLB*20] or the cross-attention mechanism [CWZ*21], were employed for better performance. Recent CNN-based methodologies handled motion in-betweening by interpolating latent vectors from a task-agnostic motion prior learned from a large set of motion capture data [HSZ*22, LVC*21]. Some of these methods utilize user-specified sparse keyframes [ZLB*20, LVC*21]. Unfortunately, the limited receptive field of CNNs often restricts their ability to capture the global information necessary for synthesizing long-term transitions.

Leveraging the high parallelism of transformers [VSP*17], neural motion in-betweening frameworks improve the generated quality even in a single forward pass [DLZ*22, KBS*22, OVH*22, QZZ22]. The state-of-the-art motion in-betweening framework [QZZ22] employed two transformers with additional training strategies such as learned embeddings and relative attention. In this work, we propose a novel divide-and-conquer strategy, moving beyond the previous approach of using two networks. Our method first identifies sparse and critical keyframes and then refines the motion to achieve high quality results.

2.2. Keyframe-driven Motion Synthesis

As keyframes capture representative poses in a motion sequence, animators can create a character animation without editing every frame but with just a few keyframes. Spacetime constraints [WK88] create physically valid motion that satisfies keyframes that a character should pass through, allowing interactive motion editing [Gle97], rapid prototyping [LP02], and abstraction of character animation [GRGC15]. Motion warping [WP95] uses keyframes as constraints on a smooth deformation to be applied to the motion curves. Tangent-space optimization [CÖS19] extends the spacetime optimization problem in the tangent space of the animation curves. In this work, we propose a novel concept of keyframe scores that brings the role of keyframes into the motion in-betweening framework, which can enhance the quality of generated results by identifying critical moments.

Extracting keyframes from a motion sequence is an important problem because it is easier to edit a motion sequence with a few keyframes than to edit dense frames one by one. To sample sparse keyframes from a dense set of motion capture frames, the Salient Poses algorithm [RASS18] formulated the optimal keyframe selection problem as a shortest path problem, which can be solved with dynamic programming. Saliency Diagrams [NRLN19] further extended the Salient Poses to allow users to interactively select the number of keyframes to use. Inspired by video keyframe extraction methods using deep reinforcement learning (DRL) [ZQX18, ZXC18], graph-based DRL for efficient unsupervised keyframe selection was also proposed [MHM*21]. In this work, we employ keyframe-driven motion synthesis via keyframe scores whose ground truth data is generated by the Salient Poses algorithm [RASS18].

2.3. Trajectory-conditioned Motion Synthesis

Trajectory information has been widely utilized in motion control tasks as it specifies the desired positions and directions of a character. Early attempts on data-driven motion synthesis involved constructing a motion graph where the edges represent plausible transitions between motion segments in the database [KGP02, LCR*02]. This graph-based approach has been employed to address the trajectory traversal problem of the root joint by finding an optimal path within the graph [KGP02, LCR*02, SH07, MC12, SKY10]. More sophisticated motion controllers employed the root trajectory in a fixed-length window that covers the past and future of the current timestep [BC15, HKS17, ZSKS18, SZKS19]. Similar to previous methods, we employ the trajectory feature as a conditional input that a character must follow, which can also be arbitrarily edited by users.

3. Method

The objective of our method is to generate seamless transition frames between multiple context frames placed in the beginning and a target frame placed at the end, even when the transition is lengthy. To this end, we employ a hierarchical two-stage approach that identifies keyframes first to represent meaningful moments of a motion sequence, then generates a natural motion based on these keyframes as shown in Figure 2. Specifically, KeyframeNet

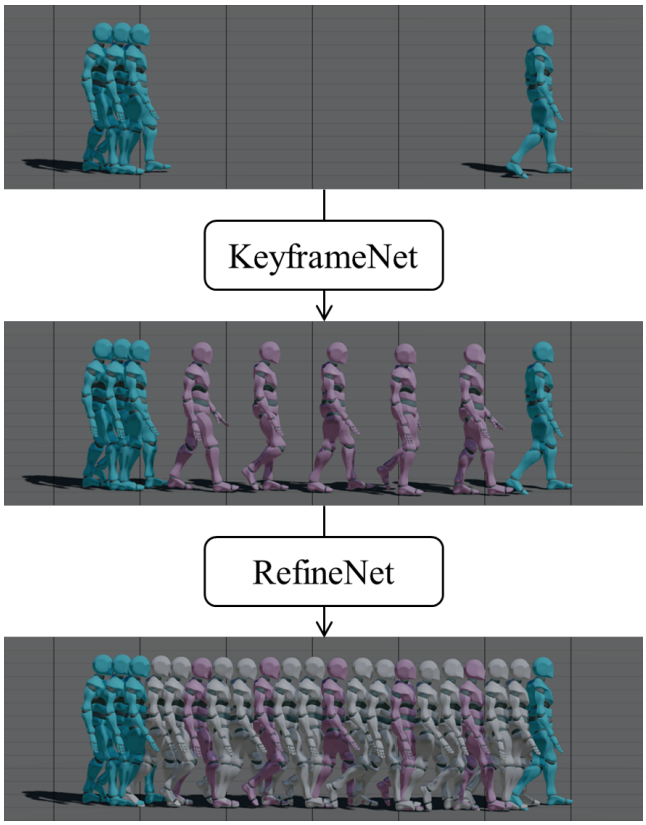


Figure 2: *System overview of our method. Given constrained frames (blue), KeyframeNet followed by an adaptive keyframe selection method determines sparse keyframes (purple), and then RefineNet synthesizes seamless transition frames (gray).*

predicts keyframes based on the context frames and the target frame (Section 3.2). Then, using an adaptive keyframe selection method, representative keyframes are selected, which are then used as anchor points of interpolation to approximate the motion sequence (Section 3.3). Finally, RefineNet adds high-frequency details given the approximated motion sequence, resulting in the final results (Section 3.4).

3.1. Data Formatting

Each motion sequence with T frames is represented as a series of pose vectors. Here, $T = t_{\text{ctx}} + t_{\text{trans}} + 1$, where t_{ctx} is the number of context frames, t_{trans} is the number of transition frames, and the remaining 1 frame is the target frame provided to the network. Each motion sequence $\mathbf{s}$ is defined as follows:

$$\mathbf{s} = [\mathbf{s}_1, \mathbf{s}_2, \dots, \mathbf{s}_T]^\top \in \mathbb{R}^{T \times D},$$

where D is the degrees of freedom of the pose vector at each frame. $\mathbf{s}_i = [\mathbf{s}_i^r, \mathbf{s}_i^p]$ is a skeletal pose vector with J joints at frame i, where $\mathbf{s}_i^r \in \mathbb{R}^{6J}$ is the local joint rotations of all joints as we employ the orthonormal 6D representation [ZBL*19], and $\mathbf{s}_i^p \in \mathbb{R}^3$ is the global root position. Therefore, the degrees of freedom of each pose vector is computed as $D = 6J + 3$.

As conditional inputs of our method, we employ a trajectory vec-

tor, a learned phase latent vector, and a mask vector. The trajectory vector is denoted as follows:

$$\mathbf{t} = [\mathbf{t}_1, \mathbf{t}_2, \ldots, \mathbf{t}_T]^\top \in \mathbb{R}^{T\times 4}.$$

We denote the trajectory feature at each frame i as $\mathbf{t}_i = [\mathbf{t}_i^p, \mathbf{t}_i^d]$ where $\mathbf{t}_i^p \in \mathbb{R}^2$ and $\mathbf{t}_i^d \in \mathbb{R}^2$ are root position and forward direction of the character on the 2D horizontal plane. Meanwhile, the phase latent vector is denoted as follows:

$$\mathbf{p} = [\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_T]^\top \in \mathbb{R}^{T\times 2N},$$

where N is the number of phase channels. Specifically, $\mathbf{p}$ is computed by a signed phase shift $\mathbf{S}$ and an amplitude $\mathbf{A}$:

$$\mathbf{p}_i^{2n-1} = \mathbf{A}_i^n \cdot \sin(2\pi \cdot \mathbf{S}_i^n), \quad \mathbf{p}_i^{2n} = \mathbf{A}_i^n \cdot \cos(2\pi \cdot \mathbf{S}_i^n),$$

where n is the phase channel index and i is the frame index. We obtain $\mathbf{S}$ and $\mathbf{A}$ via a pre-trained PAE [SMK22]. Finally, the binary-valued mask vector $\mathbf{m} \in \mathbb{R}^{T\times 1}$ indicates which frames are kept or masked out when given to the network. Specifically, $\mathbf{m}_i$ is set to one if $\mathbf{s}_i$ is given, and zero if $\mathbf{s}_i$ is masked out.

3.1.1. Keyframe Scores

To incorporate keyframe-driven neural motion synthesis, we introduce a *keyframe score* that encodes the importance of each frame within a motion sequence. Specifically, frames with high keyframe scores have the representative features of the motion. Therefore, they can serve as salient frames for effective motion synthesis.

To compute the ground truth keyframe scores from a motion sequence, we employ the Salient Poses algorithm [RASS18] that identifies a set of optimal keyframes from a motion sequence. The algorithm begins by defining a cost function that quantifies the dissimilarity between the original motion and an interpolated motion approximated by a set of keyframes. In our case, the cost function is defined as the average L2 norm of the global joint position differences between approximated and ground truth motion. For the approximation between any two frames, we used linear interpolation (lerp) for the root position and spherical linear interpolation (slerp) for the joint rotations. Subsequently, given a motion sequence with K frames, the algorithm computes sets of optimal keyframes $\{\mathcal{S}_1, \ldots, \mathcal{S}_{K-2}\}$ that minimize the cost function, in which each $\mathcal{S}_k$ comprises k frame indices selected as keyframes. Note that the maximum number of keyframes is $K-2$ because the first and last frame of the motion are used as given endpoints.

To compute the importance level of each frame across the entire sequence, we take a similar approach to Saliency Diagrams [NRLN19], which takes all possible keyframe selections into account. First, we take the last context frame and the target frame as base endpoints. In this process, our intention is to compute keyframe scores only for the transition frames (i.e. $K = t_{\text{trans}} + 2$) instead of considering the entire sequence. After computing all $\mathcal{S}_k$ for $k \in \{1, \ldots, t_{\text{trans}}\}$ via Salient Poses, we count the number of times that each frame was chosen as a keyframe. These counted values are then scaled within the range of $[0, 1]$ by dividing them by t_{trans} to ensure the most frequently selected frame has a value of one, and these normalized values represent the keyframe scores for the transition frames. To ensure consistency in dimensionality during network training, a value of one is padded to cover both the context frames and the target frame, which are not considered in the computation of keyframe scores. As a result, the ground truth keyframe scores are represented as $\mathbf{k} \in \mathbb{R}^{T\times 1}$ where $\mathbf{k}_{t\notin\{t\,|\,t_{\text{ctx}}<t<T\}} = 1$.

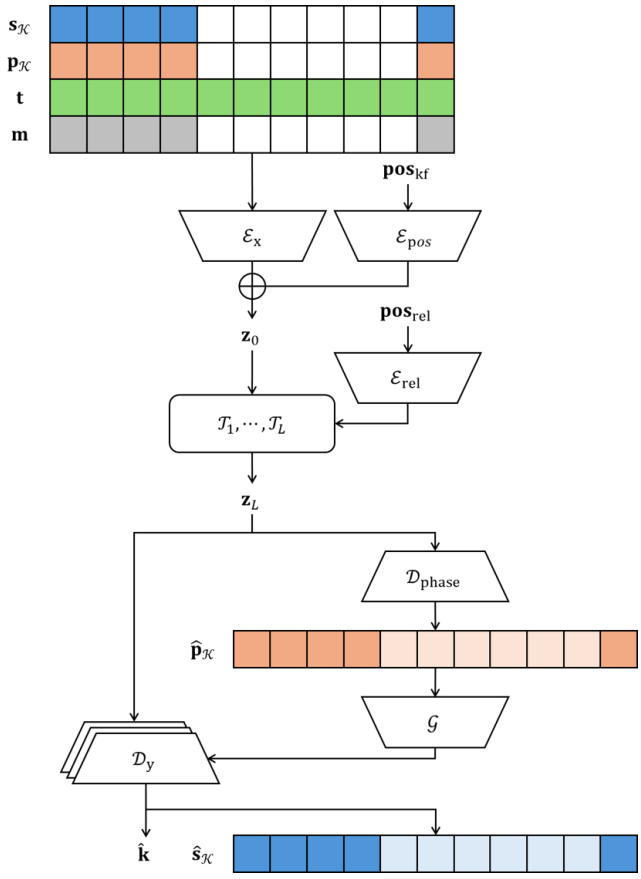


Figure 3: *KeyframeNet architecture. $\bigoplus$ represents element-wise addition. The motion decoder is based on a mixture-of-experts (MoE) architecture with multiple MLP layers.*

3.2. KeyframeNet

As shown in Figure 3, the goal of KeyframeNet is to predict the motion, phase manifolds, and keyframe scores of the transition frames given the masked motion, masked phase, trajectory condition, and mask. Therefore, the input of KeyframeNet is formulated as follows:

$$\mathbf{x}_\mathcal{K} = [\mathbf{s}_\mathcal{K}, \mathbf{p}_\mathcal{K}, \mathbf{t}, \mathbf{m}].$$

Here, $\mathbf{s}_\mathcal{K} = \mathbf{s} \odot \mathbf{m}$ and $\mathbf{p}_\mathcal{K} = \mathbf{p} \odot \mathbf{m}$ where $\odot$ denotes an element-wise multiplication operator with broadcasting. Note that the pose and phase values on transition frames are masked out when training KeyframeNet, while the trajectory vector remains intact to provide it as a condition encompassing all the frames.

The network architecture of KeyframeNet is inspired by two-stage transformers (TS-Trans) [QZZ22] that employs learned relative positional embeddings. Specifically, $\mathbf{x}_\mathcal{K}$ is projected onto the latent space via the multi-layer perceptron (MLP) encoder $\mathcal{E}_{\text{x}}$, then

the learned keyframe positional encoding is added to inject the sequence order information, resulting in:

$$\mathbf{z}_0 = \mathcal{E}_{\mathrm{x}}(\mathbf{x}_{\mathcal{K}}) + \mathcal{E}_{\mathrm{pos}}(\mathbf{pos}_{\mathrm{kf}}), \tag{1}$$

where $\mathbf{pos}_{\mathrm{kf}} \in \mathbb{R}^{T\times 2}$ is the position of each frame relative to both the last context frame and the target frame and $\mathcal{E}_{\mathrm{pos}}$ is an MLP encoder. The latent vector $\mathbf{z}_0$ is then given as the input to the transformer block.

The transformer block consists of L layers where $\mathcal{T}_l$ represents the l-th transformer layer. Each transformer layer consists of a multi-head self-attention (MHSA) layer and a position-wise feed-forward network (PFFN), along with pre-layer normalization (LayerNorm) and residual connection:

$$\tilde{\mathbf{z}}_l = \mathbf{z}_{l-1} + \mathrm{MHSA}(\mathrm{LayerNorm}(\mathbf{z}_{l-1})), \tag{2}$$

$$\mathbf{z}_l = \tilde{\mathbf{z}}_l + \mathrm{PFFN}(\mathrm{LayerNorm}(\tilde{\mathbf{z}}_l)), \tag{3}$$

where $\mathbf{z}_l$ is the output of $\mathcal{T}_l$. The learned relative positional encoding, which is denoted as $\mathcal{E}_{\mathrm{rel}}(\mathbf{pos}_{\mathrm{rel}})$, is also used to incorporate the pairwise distance between any two frames. $\mathbf{pos}_{\mathrm{rel}} \in \mathbb{R}^{(2T-1)\times 1}$ is the relative distances between two arbitrary frames and $\mathcal{E}_{\mathrm{rel}}$ is an MLP module. To be more specific, we defined the distance between frame i and frame j as $i-j$, with both i and j being within the inclusive range of $[1,T]$. Given this definition, the range of possible values for $\mathbf{pos}_{\mathrm{rel}}$ spans to $[-T+1, T-1]$, including both ends, and thus the dimension is $2T-1$. The learned relative positional encoding is injected to the attention layers via a skewing mechanism [HVU*18].

For the decoder that synthesizes a motion sequence, we employ a weight-blended MoE architecture [ZSKS18] which consists of multiple identically structured expert networks whose weights are blended by a gating network functioned by phase features. To this end, we predict phase manifolds $\hat{\mathbf{p}}_{\mathcal{K}}$ from $\mathbf{z}_L$ via the phase decoder $\mathcal{D}_{\mathrm{phase}}$. Subsequently, $\hat{\mathbf{p}}_{\mathcal{K}}$ is used as the input of the gating network $\mathcal{G}$ to blend the network parameters of each expert of $\mathcal{D}_{\mathrm{y}}$:

$$\mathbf{y}_{\mathcal{K}} = \mathcal{D}_{\mathrm{y}}(\mathbf{z}_L, \mathcal{G}(\hat{\mathbf{p}}_{\mathcal{K}})). \tag{4}$$

The output of KeyframeNet is formulated as follows:

$$\mathbf{y}_{\mathcal{K}} = [\hat{\mathbf{s}}_{\mathcal{K}}, \hat{\mathbf{p}}_{\mathcal{K}}, \hat{\mathbf{k}}],$$

where $\hat{\mathbf{s}}_{\mathcal{K}}$, $\hat{\mathbf{p}}_{\mathcal{K}}$, and $\hat{\mathbf{k}}$ are predicted pose vectors, phase features, and keyframe scores, respectively. We apply the sigmoid function to clamp the values of $\hat{\mathbf{k}}$ within the range of $(0,1)$. Because the phase manifolds, motion, and keyframe scores are predicted simultaneously, KeyframeNet can learn the joint relationship of the motion and timing with its importance of each frame during training. Keyframes with higher scores capture more probable poses at that timing, and the accurate reconstruction of keyframes contributes to the generation of high quality results. This comprehensive training procedure that identifies high-score keyframes facilitates avoiding the "dying out" problem, which is often observed in long-term motion generation.

3.3. Adaptive Keyframe Selection

In this section, we introduce an adaptive keyframe selection algorithm that ensures the uniform distribution of the selected keyframes along the temporal axis. While the keyframe score encoded for each frame is crucial when choosing representative frames, keeping proper temporal distances between keyframes is also an important factor to summarize the motion. For example, keyframes clustered at a similar timing may not correctly represent the entire sequence. Furthermore, if two adjacent keyframes are distant, interpolation might underapproximate the motions with dynamic actions, such as 360 degree turn. To alleviate this problem, we introduce a simple and effective retrieval algorithm that maintains appropriate temporal distances between keyframes while leveraging the importance encoded in keyframe scores.

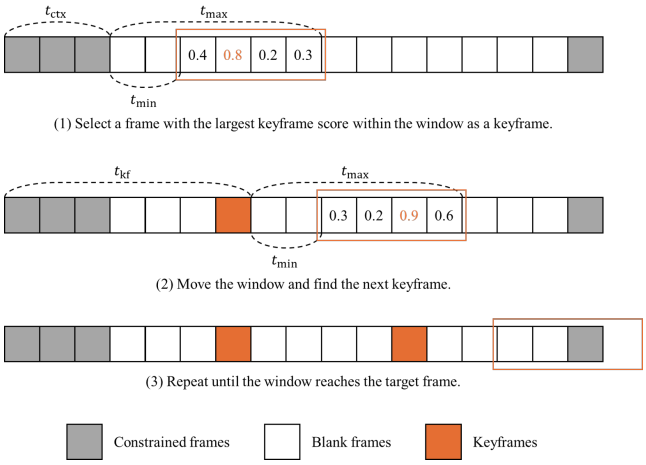


Figure 4: *Visual example on the process of adaptive keyframe selection. The number in each frame represents its keyframe score, where the orange numbers are the highest score in the window.*

As shown in Figure 4, adaptive keyframe selection is a sliding window-based algorithm. We first define a window that covers frames $[t_{\mathrm{ctx}}+t_{\mathrm{min}}, t_{\mathrm{ctx}}+t_{\mathrm{max}}]$. Then, the frame with the largest keyframe score within the window, denoted as t_{kf}, is selected as a keyframe. The window is then moved to cover frames $[t_{\mathrm{kf}}+t_{\mathrm{min}}, t_{\mathrm{kf}}+t_{\mathrm{max}}]$. This process is repeated until the target frame is included in the window. Given this set of keyframes, we approximate the poses of the non-keyframes by interpolating keyframe poses using lerp for the root position and slerp for the joint rotations. This interpolated sequence is used as an input $\mathbf{s}_{\mathcal{R}}$ to RefineNet. This approach not only prevents possible artifacts resulting from unevenly distributed keyframes but also ensures that the selected keyframes effectively represent the key moments of the motion.

3.4. RefineNet

As shown in Figure 5, RefineNet generates a natural motion sequence $\hat{\mathbf{s}}_{\mathcal{R}}$ given the input motion $\mathbf{s}_{\mathcal{R}}$. The architectural design of RefineNet closely resembles that of KeyframeNet, with the additional residual learning scheme [HZRS16] that can improve overall learning efficiency in the motion in-betweening task [OVH*22]. Therefore, we formulate the objective of RefineNet as the prediction of the difference of both motion and phase to be added to the predicted values by KeyframeNet to generate natural and coherent results.

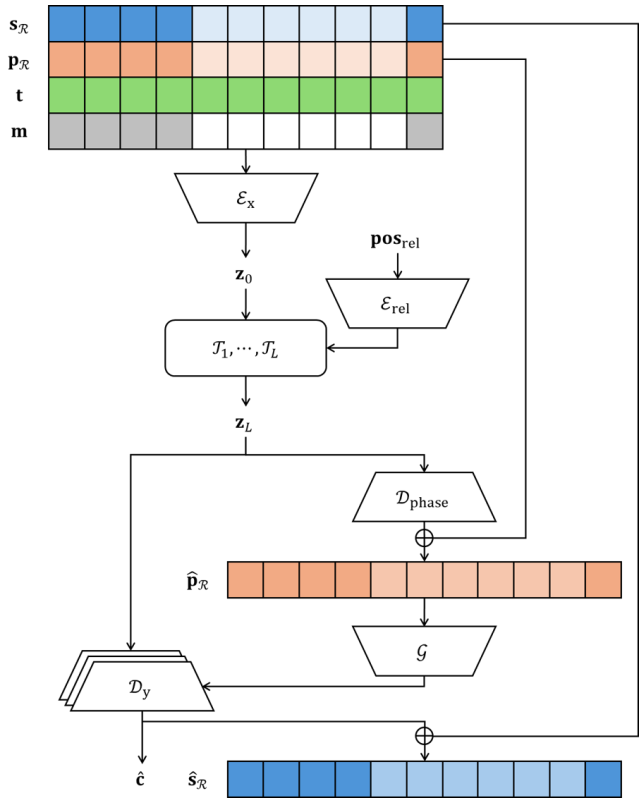


Figure 5: *RefineNet architecture.* $\bigoplus$ *represents element-wise addition for the residual connection.*

The input of RefineNet is formulated as follows:

$$\mathbf{x}_\mathcal{R} = [\mathbf{s}_\mathcal{R}, \hat{\mathbf{p}}_\mathcal{K}, \mathbf{t}, \mathbf{m}],$$

where $\mathbf{s}_\mathcal{R}$ is computed by interpolating the key poses in $\hat{\mathbf{s}}_\mathcal{K}$. $\mathbf{t}$ is the same trajectory vector as that used in KeyframeNet, and $\hat{\mathbf{p}}_\mathcal{K}$ is the output phase generated by KeyframeNet. While $\mathbf{s}_\mathcal{R}$ does not contain missing frames, the transition frames still needs to be further refined, and thus we keep the mask vector $\mathbf{m}$ same as used in KeyframeNet to indicate which frames need to be newly generated or kept intact. Additionally, because $\mathbf{s}_\mathcal{R}$ is the motion approximated by interpolating the keyframes derived from KeyframeNet and $\hat{\mathbf{p}}_\mathcal{K}$ is the phase manifolds predicted by KeyframeNet, we finish training KeyframeNet first so that its outputs can be available to be used as inputs when training RefineNet.

For the transformer layers of RefineNet, we incorporate learned relative positional encoding to consider frame-wise distances between any two frames in the attention layers, while discarding the use of keyframe positional encoding following TS-Trans [QZZ22]. Note that during the synthesis of transition frames, keyframe positional encoding was employed to account for the relative distances of each frame to the context and target frames. This was essential for the initial synthesis because the features within the context and target frames were known, while those within the transition frames are masked out. For the input of RefineNet, however, comprehensive information is available for each frame, and therefore considering distances to the context and target frames is no longer necessary. This decision aligns with the fundamental distinction between the purpose of RefineNet, which is to refine an approximated motion, and that of KeyframeNet, which involves creating a new motion.

The output of RefineNet is formulated as follows:

$$\mathbf{y}_\mathcal{R} = [\hat{\mathbf{s}}_\mathcal{R}, \hat{\mathbf{c}}],$$

where $\hat{\mathbf{s}}_\mathcal{R}$ and $\hat{\mathbf{c}}$ are the predicted motion and contact labels, respectively. These contact labels of both feet and toe joints are used to apply IK post-processing to improve the quality of generated motion when necessary.

3.5. Training

KeyframeNet and RefineNet share the identical network architecture with the difference in the absence of $\mathcal{E}_\text{pos}$ in RefineNet. The motion encoder $\mathcal{E}_\text{x}$ consists of 2 linear layers and each layer is followed by the parametric rectified linear unit (PReLU) activation function [HZRS15]. The positional encoders $\mathcal{E}_\text{pos}$ and $\mathcal{E}_\text{rel}$ and the phase decoder $\mathcal{D}_\text{phase}$ consist of 2 linear layers with the PReLU activation. The gating network $\mathcal{G}$ consists of 3 linear layers with the PReLU activation, with the last layer being followed by the softmax function to modulate the expert weights of $\mathcal{D}_\text{y}$. The motion decoder $\mathcal{D}_\text{y}$ consists of 8 experts where each expert consists of 2 linear layers with the PReLU activation. For the transformer layers, we use 6 layers (i.e. $L = 6$) and each of them uses 8 attention heads and the rectified linear unit (ReLU) activation for the PFFN modules. We use 512 hidden units for all the network components.

During training, we randomly sample the number of transition frames every iteration from a uniform distribution $\mathcal{U}[t_\text{min\_trans}, t_\text{max\_trans}]$ where $t_\text{min\_trans} = 5$ and $t_\text{max\_trans} = 90$. We also use randomly constrained poses in the transition frames as a regularization to avoid overfitting.

The losses for KeyframeNet and RefineNet, denoted as $L_\mathcal{K}$ and $L_\mathcal{R}$ respectively, are as follows:

$$\begin{aligned} L_\mathcal{K} =& \alpha_\text{rot} L_\text{rot} + \alpha_\text{pos} L_\text{pos} \\ &+ \alpha_\text{traj} L_\text{traj} + \alpha_\text{phase} L_\text{phase} + \alpha_{score} L_\text{score} \end{aligned} \tag{5}$$

$$\begin{aligned} L_\mathcal{R} =& \beta_\text{rot} L_\text{rot} + \beta_\text{pos} L_\text{pos} \\ &+ \beta_\text{traj} L_\text{traj} + \beta_\text{phase} L_\text{phase} + \beta_\text{contact} L_\text{contact}, \end{aligned} \tag{6}$$

where each term is defined as follows. First, both networks use L1 reconstruction loss of local joint rotations, global joint positions, trajectory features, and phase manifolds:

$$L_\text{rot} = \frac{1}{t_\text{trans}} \sum_{t=t_\text{ctx}+1}^{T-1} \|\mathbf{s}_t^r - \hat{\mathbf{s}}_t^r\|_1, \tag{7}$$

$$L_\text{pos} = \frac{1}{t_\text{trans}} \sum_{t=t_\text{ctx}+1}^{T-1} \|\text{FK}(\mathbf{s}_t) - \text{FK}(\hat{\mathbf{s}}_t)\|_1, \tag{8}$$

$$L_\text{traj} = \frac{1}{t_\text{trans}} \sum_{t=t_\text{ctx}+1}^{T-1} \left(\|\mathbf{t}_t^p - \hat{\mathbf{t}}_t^p\|_1 + \|1 - \mathbf{t}_t^d \cdot \hat{\mathbf{t}}_t^d\|_1 \right), \tag{9}$$

$$L_\text{phase} = \frac{1}{t_\text{trans}} \sum_{t=t_\text{ctx}+1}^{T-1} \left(\|\mathbf{p}_t - \hat{\mathbf{p}}_t\|_1 \right), \tag{10}$$

where $\mathrm{FK}(\cdot)$ is a forward kinematics operation, which can resolve positional errors caused by accumulated rotational errors in a kinematic chain [PGA18]. $\mathbf{s}^r$, $\mathbf{t}^p$, and $\mathbf{t}^d$ are the ground truth of local joint rotations, trajectory position, and trajectory direction, respectively, as defined in Section 3.1. The predicted trajectory vector $\hat{\mathbf{t}}$ is not directly computed by the networks. Instead, we compute $\hat{\mathbf{t}}$ from the predicted motion $\hat{\mathbf{s}}$ with the same way in Section 3.1. We also use additional loss terms designed for the purpose of each network. For KeyframeNet, we use an additional reconstruction loss for keyframe scores:

$$L_{\text{score}} = \frac{1}{t_{\text{trans}}} \sum_{t=t_{\text{ctx}}+1}^{T-1} \|\mathbf{k}_t - \hat{\mathbf{k}}_t\|_1. \tag{11}$$

For RefineNet, we use a reconstruction loss term for foot contact labels:

$$L_{\text{contact}} = \frac{1}{t_{\text{trans}}} \sum_{t=t_{\text{ctx}}+1}^{T-1} \|\mathbf{c}_t - \hat{\mathbf{c}}_t\|_1. \tag{12}$$

Notably, all the loss terms are computed on the transition frames because constrained frames are restored by the input frames. In our case, the coefficient for each loss term is set to $\alpha_{\text{rot}} = 1.0, \alpha_{\text{pos}} = 1.0, \alpha_{\text{traj}} = 0.5, \alpha_{\text{phase}} = 1.0, \alpha_{score} = 1.0$, $\beta_{\text{rot}} = 1.0, \beta_{\text{pos}} = 2.0, \beta_{\text{traj}} = 0.5, \beta_{\text{phase}} = 1.0$, and $\beta_{\text{contact}} = 0.1$.

4. Experiments and Evaluation

4.1. Implementation Details

We used the LaFAN1 [HYNP20] and 100STYLE [MSK22] datasets for training. The LaFAN1 dataset contains various action types such as walking, dancing, and crawling, while the 100STYLE dataset contains 100 different styles of locomotions, such as aeroplane, kick, old, and zombie. We retargeted all the motion sequences to a common character with 22 joints, and we sampled both datasets at 30Hz frame rate, resulting in 496,672 and 2,390,073 motion frames in total for each LaFAN1 and 100STYLE. We used Subject 5 of LaFAN1 as the test subject and other subjects as the training subjects. For the 100STYLE dataset, we randomly sampled 20% of the motion sequences as the test data while the remainder was used as the training data. We then extracted training and test sets by sliding a window of 101 frames through the motion sequences, shifting by 20 and 50 frames for the LaFAN1 and 100STYLE datasets, respectively. All windows were pre-processed so that the root joint of the last context frame faces the Z^+ axis and is located at the origin of the XZ plane. For all experiments, we set $t_{\text{ctx}} = 10$.

Our method is implemented in PyTorch and runs on an Intel i9-11900F 2.5GHz CPU with 64GB memory, and an NVIDIA GeForce RTX 3090 GPU. We used the Adam optimizer [KB14] with a learning rate scheduled by a Noam scheduler [VSP*17] with the warm-up step set to 8,000 iterations, and 64 motion clips were taken as a batch. KeyframeNet and RefineNet were trained for 500 epochs and 1,000 epochs, respectively, which required around 2.5 hours and 6.5 hours each for the LaFAN1 dataset and 5 hours and 13 hours each for the 100STYLE dataset. For the evaluation, we disabled the IK post-processing unless it is explicitly specified.

4.2. Quantitative Evaluation

To compare the performance of our method to those of competing methods, we used the transition benchmarks L2P and L2Q introduced by Harvey et al. [HYNP20]. L2P measures the average L2 distance of the normalized global joint positions where the mean and standard deviation for the normalization are calculated from the training dataset. Like L2P, L2Q measures the average L2 distance of the global joint rotations in quaternion but without normalization. We also used the Normalized Power Spectrum Similarity (NPSS) [GMK*19] that evaluates the angular differences of joint rotations in the frequency domain. This captures the perceptual similarity of motions that may not be discernible when using metrics evaluated solely on raw motion features, such as L2P and L2Q. Finally, we measured the foot skating (FS) to evaluate the contact consistency of the generated motions, assuming the velocity of toes and feet should be close to zero when the contact with the ground occurs. We define FS as follows:

$$\text{FS} = \frac{1}{t_{\text{trans}}} \sum_{t=t_{\text{ctx}}+1}^{T-1} \|\hat{\mathbf{c}}_t \cdot \hat{\mathbf{v}}_t^f\|_1, \tag{13}$$

where $\hat{\mathbf{v}}_t^f \in \mathbb{R}^4$ and $\hat{\mathbf{c}}_t \in \mathbb{R}^4$ are the velocity and predicted contact labels, respectively, for the feet and toe joints at frame t. This metric indicates if the network was learned to capture the coherence between the predicted contact information and actual foot velocities. For the interpolated motions, we used the ground contact labels $\mathbf{c}$ because the predicted contact labels are not available for interpolated motions.

In Table 1, we compare the values evaluated by different metrics for the following methods: naive interpolation, RTN [HYNP20], TS-Trans [QZZ22], and ours method. For a fair comparison, we trained RTN, TS-Trans, and ours on up to 90 transition frames. The table shows that our method achieved better performance compared to the others. Specifically, the errors occurred by the interpolation significantly grow as t_{trans} increases, while those occurred by the other learning-based methods grow less rapidly in general. Both TS-trans and ours outperformed RTN for all the metrics throughout all experiments with different transition frames. This proves that transformers are better suited than the LSTM architectures for stable motion synthesis. Both TS-Trans and ours achieved equivalent results for short transitions of $t_{\text{trans}} = 15$. However, our model consistently outperformed TS-Trans in all metrics for long transitions of $t_{\text{trans}} = 30, 60$, and 90 with notable performance gaps as t_{trans} increased. This proves that our method that involves trajectory features, phase manifolds, and keyframe scores is more robust than other motion in-betweening methods for long-term transition synthesis while not sacrificing the performance on short transitions. Especially, considering diverse actions contained in LaFAN1 and diverse styles in 100STYLE, we can conclude that our method is generalizable to handle various motion sequences. This is verified by the fact that our method achieved the best results for most cases and the second best results for a few cases.

4.3. Qualitative Evaluation

We qualitatively compared our method with interpolation, RTN and TS-Trans, and the results are shown in Figure 6. We set $t_{\text{trans}} = 90$ to

Dataset	LaFAN1															
Metric	L2P ↓				L2Q ↓				10× NPSS ↓				10× FS ↓			
t_{trans}	15	30	60	90	15	30	60	90	15	30	60	90	15	30	60	90
Interp	0.78	1.21	2.07	3.01	0.38	0.54	0.70	0.82	0.14	0.59	1.85	3.54	1.95	2.35	1.85	1.40
RTN	0.62	0.95	1.53	2.24	0.33	0.46	0.62	0.73	0.12	0.47	1.55	3.04	0.92	0.83	0.88	0.81
TS-Trans	0.40	0.69	1.28	2.00	**0.26**	0.40	0.58	0.72	**0.10**	0.42	1.41	2.90	**0.29**	0.37	0.61	0.85
Ours	**0.37**	**0.54**	**0.78**	**1.01**	0.28	**0.38**	**0.50**	**0.56**	**0.10**	**0.38**	**1.16**	**2.06**	0.32	**0.36**	**0.42**	**0.57**

Dataset	100STYLE															
Metric	L2P ↓				L2Q ↓				10× NPSS ↓				10× FS ↓			
t_{trans}	15	30	60	90	15	30	60	90	15	30	60	90	15	30	60	90
Interp	2.08	2.68	3.39	4.15	0.29	0.39	0.50	0.62	0.09	0.37	1.06	1.94	0.91	1.06	0.85	0.61
RTN	1.17	1.60	2.10	2.59	0.22	0.30	0.37	0.44	0.06	0.23	0.68	1.31	0.41	0.44	0.52	0.56
TS-Trans	0.56	0.85	1.34	1.96	**0.13**	0.18	0.25	0.34	**0.03**	**0.13**	0.41	0.96	**0.07**	**0.08**	0.15	0.30
Ours	**0.53**	**0.73**	**0.88**	**1.00**	**0.13**	**0.17**	**0.21**	**0.24**	0.04	**0.13**	**0.34**	**0.58**	0.08	0.09	**0.10**	**0.11**

Table 1: *Quantitative comparison of benchmark results with interpolation, RTN, TS-Trans, and our method with the LaFAN1 dataset. The best result of each column is in bold.*

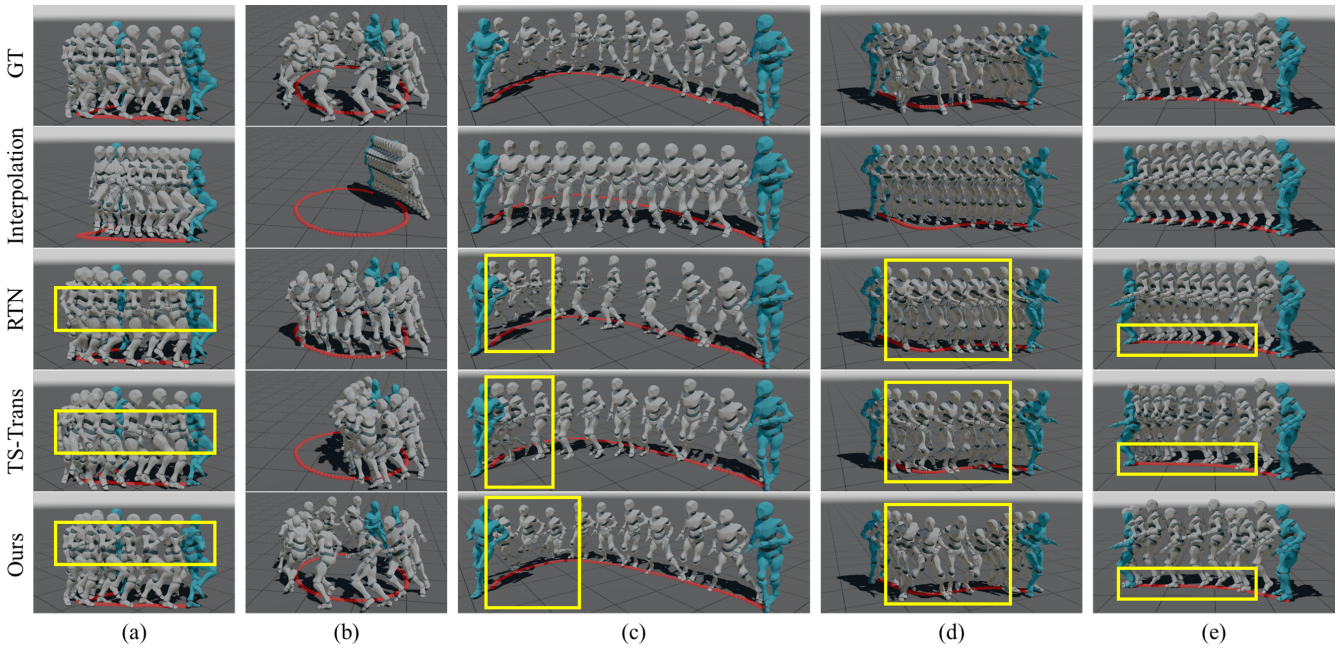


Figure 6: *Qualitative comparison of our method with the competing methods. Red dots represent the ground truth trajectory. Poses are rendered every 10 frames and constrained frames are rendered in blue while transition frames are rendered in gray.*

compare the performance of different methods on generating long transitions. Overall, interpolation showed overly smooth and implausible results compared to the learning-based methods. Compared to other learning-based methods, ours produced more distinct and dynamic poses with high-frequency details. Specifically, as shown in Figure 6-(a) and (d), RTN and TS-Trans produced inactive movements for hands and upper body, while ours produced more active movements that resemble the ground truth motions. Additionally, as shown in Figure 6-(b) and (c), our method generated better results than other methods in both motion quality and trajectory alignment. While RTN produced plausible results on trajectory following, it generated static poses across the entire frames, which resulted in floating artifacts. Although TS-Trans generated motions with better quality than those generated by RTN, those motions eventually converged to static poses at the end of the transitions, which also resulted in floating artifacts. The floating artifacts caused by static poses are more noticeable from the results of jumping motions as shown in Figure 6-(e). Our method produced plausible jumping movements where both feet leave the ground, whereas those produced by RTN and TS-Trans show the feet staying on the ground while the entire body moves forward. As a result, we can conclude that our method yields motions that are more dy-

namic and conform better to the given trajectory compared to other methods. For more animated visualizations, including longer transition frames than those presented in the paper, please refer to the supplementary video.

4.4. Ablation Study

We conducted an ablation study on our design choices to demonstrate the effectiveness of each component, including trajectory conditions, phase manifolds, and keyframe-based synthesis. Table 2 shows the quantitative results achieved by removing each component, with the complete model yielding the best results. Removing trajectory conditions had the largest effect on the results compared to other components. Notably, the performance drop becomes more rapid on longer transitions of 60 and 90 frames. This shows that providing the approximate movement of the character via trajectory conditions have a meaningful effect on disambiguating the movements of the character in the long transitions. We also observed that removing phase features had a negative effect. Especially, the performance drop of FS was particularly noticeable. This implies that incorporating phase features effectively improves the quality of generated motions with less artifacts. Finally, excluding the keyframe-driven synthesis had a negative effect on L2P while it did not lead to substantial differences in other metrics. Considering that positional errors are more visually noticeable than rotational errors, we can conclude that the keyframe-driven synthesis process contributes to refining additional details while maintaining the overall quality. We provide more comprehensive evaluations with animated results in the supplementary video.

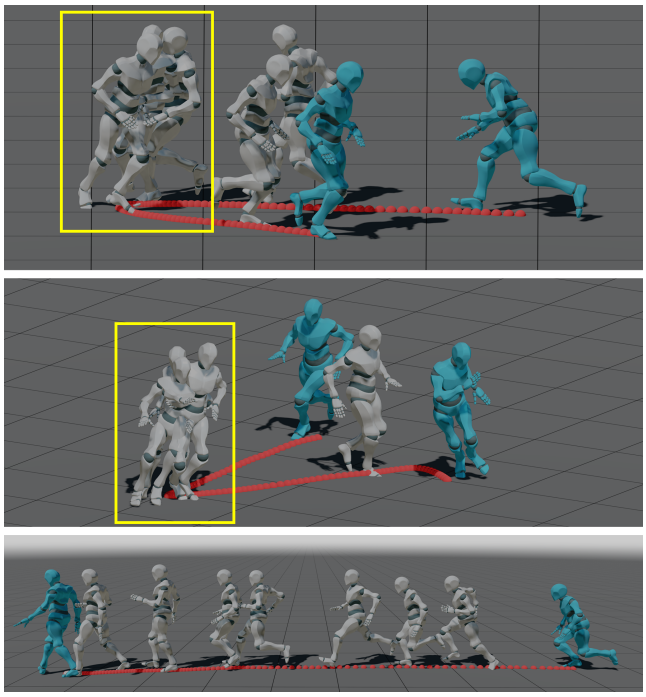


Figure 7: *Visual examples of selected keyframes given constrained frames and the trajectory. Keyframes are concentrated on the edge points along the curvy trajectories as highlighted in the yellow boxes, while they are evenly distributed along the straight trajectory.*

4.5. Effectiveness of Adaptive Keyframe Selection

To prove the effectiveness of the adaptive keyframe selection method in generating temporally coherent results, we present results of different keyframe selection methods. In Table 3, we present the benchmark results produced using five different keyframe selection methods: (i) keyframes with scores higher than a threshold, (ii) a fixed number of top-scored keyframes, (iii) all the predicted intermediate frames as keyframes, (iv) uniform selection of predicted transition frames with a fixed interval, and (v) our adaptive keyframe selection. The cases of (iii) and (iv) do not take keyframe scores into consideration unlike other cases. Because keyframe-driven synthesis was designed for long-term motion synthesis, we evaluated the metrics on 60 and 90 transition frames. For a fair comparison, we set the average number of selected keyframes for each method to be similar except the case of (iii) where the number of keyframes does not change. Specifically, for 60 transition frames, 3.00 frames were selected from threshold, top-k selection, and uniform selection, while 2.65 frames were selected from ours. Additionally, for 90 transition frames, 4.59, 4.00, 4.00, and 4.40 frames were selected from threshold, top-k, uniform selection, and ours, respectively. Consequently, adaptive keyframe selection produced comparable results on L2Q and NPSS and outperformed other selection methods for L2P and FS, especially when the transition became longer. Notably, using all the transition frames as keyframes produced the worst results, showing that selecting meaningful frames through keyframe scores has a positive effect on improving the quality of generated results. Furthermore, taking frames at a uniform distance as keyframes produced comparable results to ours, demonstrating that maintaining proper temporal distances between keyframes is also an important factor. As a result, our adaptive keyframe selection that combines both keyframe scores and proper temporal distances between keyframes produced the best results among those produced by various selection methods.

Figure 7 presents the keyframes selected using our adaptive keyframe selection method. Notably, when the trajectory includes drastic curves, which requires rapid changes of movements, keyframes are highly concentrated on the knee points of the trajectory. In contrast, when the trajectory is straight and the transition involves relatively simple locomotions, keyframes are evenly distributed. This verifies that our keyframe scores and adaptive keyframe selection method are useful in capturing salient moments of the transitions and can lead to improved quality of the in-betweening results.

4.6. Application with Trajectory Editing

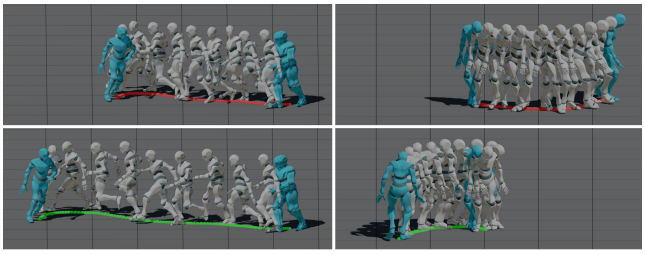
To demonstrate how our method can be applied to the trajectory editing, we evaluated our method on various trajectories created by two approaches. In one scenario, we scaled the translational velocity of each trajectory point as shown in Figure 8-(a). In the other scenario, we randomly sampled a trajectory vector from another

Metric	L2P ↓				L2Q ↓				10× NPSS ↓				10× FS ↓			
t_{trans}	15	30	60	90	15	30	60	90	15	30	60	90	15	30	60	90
(-) Traj	0.40	0.62	1.05	1.78	0.29	0.41	0.56	0.73	0.11	0.42	1.43	3.17	0.37	0.51	0.80	1.53
(-) Phase	0.38	0.55	0.86	1.18	0.29	**0.38**	**0.50**	0.58	**0.10**	**0.38**	**1.14**	2.13	0.65	0.77	1.14	1.66
(-) Keyframe	0.43	0.65	0.91	1.12	0.29	0.40	0.51	0.57	0.11	0.41	1.18	**2.06**	**0.30**	**0.36**	0.44	**0.55**
Ours	**0.37**	**0.54**	**0.78**	**1.01**	**0.28**	**0.38**	**0.50**	**0.56**	**0.10**	**0.38**	1.16	**2.06**	0.32	**0.36**	**0.42**	0.57

Table 2: *Ablation results of our method with the LaFAN1 dataset. (-) represents the exclusion of the component from the complete model. Ours in the last row presents the complete method. The best result for each column is in bold.*

Metric	L2P ↓		L2Q ↓		10× NPSS ↓		10× FS ↓	
t_{trans}	60	90	60	90	60	90	60	90
Threshold	0.81	1.21	**0.48**	0.57	**1.10**	2.11	0.57	0.93
Top-k	0.80	1.19	**0.48**	0.57	1.11	2.10	0.56	0.91
All	0.86	1.13	0.53	0.60	1.24	2.18	0.76	0.89
Uniform	0.79	1.03	0.51	0.57	1.18	2.09	0.43	0.59
Ours	**0.78**	**1.01**	0.50	**0.56**	1.16	**2.06**	**0.42**	**0.57**

Table 3: *Benchmark results of different keyframe selection methods with the LaFAN1 dataset. Ours with the proposed adaptive keyframe selection produces the best results. The best result for each column is in bold.*



(a) Velocity scaling (b) Random trajectory

Figure 8: *Generated transition frames (gray) given constrained frames (blue) using our method on different trajectories. Red dots represent the original trajectories and the green dots represent the edited trajectories.*

data sample as shown in Figure 8-(b). For both cases, we maintained the poses and trajectory features in the context frames same as the original ground truth motion. Then we transformed the target pose so that its root position and direction align with the transformation of the trajectory of the target frame. Overall, our method produced motions that adapt to different trajectories while not losing the naturalness. he given trajectories are unseen inputs that are not included in the dataset. Natural-looking generated results from this experiment suggest that our method allows users to edit the trajectory to produce desired results in practical applications. For the animated results, please refer to the supplementary video.

To quantitatively evaluate the edited results, we measured the positional error between the generated trajectory and the input trajectory instead of using benchmark metrics because there are no ground truth motions on edited trajectories. As shown in Table 4, our results closely followed the given trajectory with only less than 0.3cm of error on average in the worst case. To demonstrate that this level of error does not significantly affect visual quality, we conducted an additional user study. We sampled 20 motions, and each of them was evaluated on a 5-point Likert scale, 5 being the best, for two questions: accuracy of trajectory following and naturalness of the generated motion. We recruited 19 participants (10 males and 9 females in the age distribution from 24 to 32) for this study. As shown in Figure 9, we obtained the scores of 4.69 and 4.36 in trajectory accuracy and naturalness, respectively, which implies that the generated outputs were favored by most of the participants. This shows that our method is capable of generating plausible motion from arbitrary trajectories, which is important for real-world applications.

Dataset	LaFAN1			
t_{trans}	15	30	60	90
Random	0.108	0.217	0.206	0.259
Velocity scaling	0.099	0.205	0.186	0.247

Dataset	100STYLE			
t_{trans}	15	30	60	90
Random	0.051	0.082	0.075	0.087
Velocity scaling	0.048	0.083	0.074	0.090

Table 4: *Positional errors between the input and generated trajectories in centimeters.*

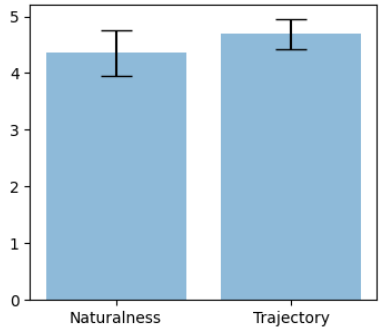


Figure 9: *User study results with a 95% of confidence level.*

5. Discussion

Although our method can effectively deal with long-term transitions, it still has some limitations to overcome. Like other learning-based methods, our method is limited by the training dataset. For instance, when the target pose or given trajectory significantly deviates from the training dataset, our method may yield suboptimal

outcomes with floating artifacts. This happens when the trajectory and the root joint features are represented with respect to the world coordinate system, making the neural networks struggle to deal with such out-of-distribution data. Similarly, our method is not applicable to infinitely long transitions because the range of the target pose or trajectory curves can be out of the training distribution. To overcome these limitations, developing coordinate-invariant or length-invariant in-betweening methods inspired by motion control algorithms could be a promising direction for future work.

Additionally, our method relies on the performance of keyframe extraction, which currently utilizes handcrafted features. To overcome this limitation, a more elegant solution would be to create an automated keyframe extraction module, which eliminates the need for manual pre-processing and potentially enhances the motion in-betweening performance. This would include a learning-based component which encodes salient information into motion representation, thereby enabling automatic keyframe extraction from raw motion sequences.

Lastly, our method struggles to generate highly diverse results. To address this limitation, future directions could involve the integration of generative models like variational autoencoders [KW13] or diffusion models [HJA20, SDWMG15]. These improvements have the potential to enhance realism by generating subtle and high-frequency details in the results, such as diverse hand movements and different but plausible footstep patterns while following the same trajectory and the target frame.

6. Conclusion

In this work, we present a method for long-term motion in-betweening via keyframe prediction. By employing a keyframe-driven process for motion in-betweening, we can effectively resolve the pose ambiguity problem, resulting in high performance on long-term transition synthesis. We show that phase manifolds are effective in capturing the relationship of time and pose, enhancing the quality of generated motions. Both quantitative and qualitative experiment results prove that our method is more stable on long-term prediction while producing comparable results on short transitions when compared with the state-of-the-art method. The ablation studies on our design components, including trajectory conditions, phase manifolds, and keyframe-driven synthesis, verify that each component has a crucial impact on improving the quality of the generated results. We also show that our method robustly handles arbitrary trajectories. Our future work includes developing a more generalizable model that can handle arbitrarily long transitions as well as reflecting the stochastic nature of motions using generative models.

Acknowledgements

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (RS-2024-00333478).

References

[BC15] BÜTTNER M., CLAVET S.: Motion matching-the road to next gen animation. *Proc. of Nucl. ai 1*, 2015 (2015), 2. 3

[CH07] CHAI J., HODGINS J. K.: Constraint-based motion optimization using a statistical dynamic model. In *ACM SIGGRAPH 2007 papers*. 2007, pp. 8–es. 2

[CÖS19] CICCONE L., ÖZTIRELI C., SUMNER R. W.: Tangent-space optimization for interactive animation control. *ACM Transactions on Graphics (TOG) 38*, 4 (2019), 1–10. 3

[CWZ*21] CAI Y., WANG Y., ZHU Y., CHAM T.-J., CAI J., YUAN J., LIU J., ZHENG C., YAN S., DING H., ET AL.: A unified 3d human motion synthesis model via conditional variational auto-encoder. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2021), pp. 11645–11655. 2

[DLZ*22] DUAN Y., LIN Y., ZOU Z., YUAN Y., QIAN Z., ZHANG B.: A unified framework for real time motion completion. In *Proceedings of the AAAI Conference on Artificial Intelligence* (2022), vol. 36, pp. 4459–4467. 2

[Gle97] GLEICHER M.: Motion editing with spacetime constraints. In *Proceedings of the 1997 symposium on Interactive 3D graphics* (1997), pp. 139–ff. 3

[GMK*19] GOPALAKRISHNAN A., MALI A., KIFER D., GILES L., ORORBIA A. G.: A neural temporal model for human motion prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2019), pp. 12116–12125. 7

[GRGC15] GUAY M., RONFARD R., GLEICHER M., CANI M.-P.: Space-time sketching of character animation. *ACM Transactions on Graphics (ToG) 34*, 4 (2015), 1–10. 3

[HGMN19] HERNANDEZ A., GALL J., MORENO-NOGUER F.: Human motion prediction via spatio-temporal inpainting. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2019), pp. 7134–7143. 2

[HJA20] HO J., JAIN A., ABBEEL P.: Denoising diffusion probabilistic models. *Advances in neural information processing systems 33* (2020), 6840–6851. 11

[HKS17] HOLDEN D., KOMURA T., SAITO J.: Phase-functioned neural networks for character control. *ACM Transactions on Graphics (TOG) 36*, 4 (2017), 1–13. 3

[HP18] HARVEY F. G., PAL C.: Recurrent transition networks for character locomotion. In *SIGGRAPH Asia 2018 Technical Briefs*. 2018, pp. 1–4. 2

[HSZ*22] HE C., SAITO J., ZACHARY J., RUSHMEIER H., ZHOU Y.: Nemf: Neural motion fields for kinematic animation. *Advances in Neural Information Processing Systems 35* (2022), 4244–4256. 2

[HVU*18] HUANG C.-Z. A., VASWANI A., USZKOREIT J., SHAZEER N., SIMON I., HAWTHORNE C., DAI A. M., HOFFMAN M. D., DINCULESCU M., ECK D.: Music transformer. *arXiv preprint arXiv:1809.04281* (2018). 5

[HYNP20] HARVEY F. G., YURICK M., NOWROUZEZAHRAI D., PAL C.: Robust motion in-betweening. *ACM Transactions on Graphics (TOG) 39*, 4 (2020), 60–1. 2, 7

[HZRS15] HE K., ZHANG X., REN S., SUN J.: Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision* (2015), pp. 1026–1034. 6

[HZRS16] HE K., ZHANG X., REN S., SUN J.: Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2016), pp. 770–778. 5

[KAS*20] KAUFMANN M., ASKAN E., SONG J., PECE F., ZIEGLER R., HILLIGES O.: Convolutional autoencoders for human motion infilling. In *2020 International Conference on 3D Vision (3DV)* (2020), IEEE, pp. 918–927. 2

[KB14] KINGMA D. P., BA J.: Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014). 7

[KBS*22] KIM J., BYUN T., SHIN S., WON J., CHOI S.: Conditional motion in-betweening. *Pattern Recognition 132* (2022), 108894. 2

[KGP02] KOVAR L., GLEICHER M., PIGHIN F.: Motion graphs. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques* (2002), pp. 473–482. 3

[KW13] KINGMA D. P., WELLING M.: Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114* (2013). 11

[LCR*02] LEE J., CHAI J., REITSMA P. S., HODGINS J. K., POLLARD N. S.: Interactive control of avatars animated with human motion data. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques* (2002), pp. 491–500. 3

[LGN14] LEHRMANN A. M., GEHLER P. V., NOWOZIN S.: Efficient nonlinear markov models for human motion. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2014), pp. 1314–1321. 2

[LP02] LIU C. K., POPOVIĆ Z.: Synthesis of complex dynamic character motion from simple animations. *ACM Transactions on Graphics (TOG) 21*, 3 (2002), 408–416. 3

[LVC*21] LI J., VILLEGAS R., CEYLAN D., YANG J., KUANG Z., LI H., ZHAO Y.: Task-generic hierarchical human motion prior using vaes. In *2021 International Conference on 3D Vision (3DV)* (2021), IEEE, pp. 771–781. 2

[MC12] MIN J., CHAI J.: Motion graphs++ a compact generative model for semantic motion analysis and synthesis. *ACM Transactions on Graphics (TOG) 31*, 6 (2012), 1–12. 3

[MCC09] MIN J., CHEN Y.-L., CHAI J.: Interactive generation of human animation with deformable motion models. *ACM Transactions on Graphics (TOG) 29*, 1 (2009), 1–12. 2

[MHM*21] MO C., HU K., MEI S., CHEN Z., WANG Z.: Keyframe extraction from motion capture sequences with graph based deep reinforcement learning. In *Proceedings of the 29th ACM International Conference on Multimedia* (2021), pp. 5194–5202. 3

[MK05] MUKAI T., KURIYAMA S.: Geostatistical motion interpolation. In *ACM SIGGRAPH 2005 Papers*. 2005, pp. 1062–1070. 2

[MSK22] MASON I., STARKE S., KOMURA T.: Real-time style modelling of human locomotion via feature-wise transformations and local motion phases. *Proceedings of the ACM on Computer Graphics and Interactive Techniques 5*, 1 (2022), 1–18. 7

[NRLN19] NGHIEM N., ROBERTS R., LEWIS J. P., NOH J.: Saliency diagrams. In *SIGGRAPH Asia 2019 Technical Briefs*. 2019, pp. 49–52. 3, 4

[OVH*22] ORESHKIN B. N., VALKANAS A., HARVEY F. G., MÉNARD L.-S., BOCQUELET F., COATES M. J.: Motion inbetweening via deep δ-interpolator. *arXiv preprint arXiv:2201.06701* (2022). 2, 5

[PGA18] PAVLLO D., GRANGIER D., AULI M.: Quaternet: A quaternion-based recurrent model for human motion. *arXiv preprint arXiv:1805.06485* (2018). 7

[QZZ22] QIN J., ZHENG Y., ZHOU K.: Motion in-betweening via two-stage transformers. *ACM Transactions on Graphics (TOG) 41*, 6 (2022), 1–16. 2, 4, 6, 7

[RASS18] ROBERTS R., ANJYO K., SEO J., SEOL Y.: Optimal and interactive keyframe selection for motion capture. In *SIGGRAPH Asia 2018 Technical Briefs*. 2018, pp. 1–4. 3, 4

[RCB98] ROSE C., COHEN M. F., BODENHEIMER B.: Verbs and adverbs: Multidimensional motion interpolation. *IEEE Computer Graphics and Applications 18*, 5 (1998), 32–40. 2

[RGBC96] ROSE C., GUENTER B., BODENHEIMER B., COHEN M. F.: Efficient generation of motion transitions using spacetime constraints. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques* (1996), pp. 147–154. 2

[RISC01] ROSE III C. F., SLOAN P.-P. J., COHEN M. F.: Artist-directed inverse-kinematics using radial basis function interpolation. In *Computer graphics forum* (2001), vol. 20, Wiley Online Library, pp. 239–250. 2

[SDWMG15] SOHL-DICKSTEIN J., WEISS E., MAHESWARANATHAN N., GANGULI S.: Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning* (2015), PMLR, pp. 2256–2265. 11

[SH07] SAFONOVA A., HODGINS J. K.: Construction and optimal search of interpolated motion graphs. In *ACM SIGGRAPH 2007 papers*. 2007, pp. 106–es. 3

[SKY10] SHUM H. P., KOMURA T., YAMAZAKI S.: Simulating multiple character interactions with collaborative and adversarial goals. *IEEE Transactions on Visualization and Computer Graphics 18*, 5 (2010), 741–752. 3

[SMK22] STARKE S., MASON I., KOMURA T.: Deepphase: Periodic autoencoders for learning motion phase manifolds. *ACM Transactions on Graphics (TOG) 41*, 4 (2022), 1–13. 2, 4

[SSKS23] STARKE P., STARKE S., KOMURA T., STEINICKE F.: Motion in-betweening with phase manifolds. *Proceedings of the ACM on Computer Graphics and Interactive Techniques 6*, 2 (2023), 1–17. 2

[SZKS19] STARKE S., ZHANG H., KOMURA T., SAITO J.: Neural state machine for character-scene interactions. *ACM Trans. Graph. 38*, 6 (2019), 209–1. 3

[TWH*22] TANG X., WANG H., HU B., GONG X., YI R., KOU Q., JIN X.: Real-time controllable motion transition for characters. *ACM Transactions on Graphics (TOG) 41*, 4 (2022), 1–10. 2

[TWW*23] TANG X., WU L., WANG H., HU B., GONG X., LIAO Y., LI S., KOU Q., JIN X.: Rsmt: Real-time stylized motion transition for characters. *arXiv preprint arXiv:2306.11970* (2023). 2

[VSP*17] VASWANI A., SHAZEER N., PARMAR N., USZKOREIT J., JONES L., GOMEZ A. N., KAISER Ł., POLOSUKHIN I.: Attention is all you need. *Advances in neural information processing systems 30* (2017). 2, 7

[WFH07] WANG J. M., FLEET D. J., HERTZMANN A.: Gaussian process dynamical models for human motion. *IEEE transactions on pattern analysis and machine intelligence 30*, 2 (2007), 283–298. 2

[WK88] WITKIN A., KASS M.: Spacetime constraints. *ACM Siggraph Computer Graphics 22*, 4 (1988), 159–168. 2, 3

[WP95] WITKIN A., POPOVIC Z.: Motion warping. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques* (1995), pp. 105–108. 3

[ZBL*19] ZHOU Y., BARNES C., LU J., YANG J., LI H.: On the continuity of rotation representations in neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2019), pp. 5745–5753. 3

[ZLB*20] ZHOU Y., LU J., BARNES C., YANG J., XIANG S., ET AL.: Generative tweening: Long-term inbetweening of 3d human motions. *arXiv preprint arXiv:2005.08891* (2020). 2

[ZQX18] ZHOU K., QIAO Y., XIANG T.: Deep reinforcement learning for unsupervised video summarization with diversity-representativeness reward. In *Proceedings of the AAAI Conference on Artificial Intelligence* (2018), vol. 32. 3

[ZSKS18] ZHANG H., STARKE S., KOMURA T., SAITO J.: Mode-adaptive neural networks for quadruped motion control. *ACM Transactions on Graphics (TOG) 37*, 4 (2018), 1–11. 3, 5

[ZXC18] ZHOU K., XIANG T., CAVALLARO A.: Video summarisation by classification with deep reinforcement learning. *arXiv preprint arXiv:1807.03089* (2018). 3