Diverse Motion In-betweening from Sparse Keyframes with Dual Posture Stitching

Tianxiang Ren, Jubo Yu, Shihui Guo*, Ying Ma, Yutao Ouyang, Zijiao Zeng, Yazhan Zhang, Yipeng Qin

Fig. 1: Motion transitions generated by our method. The poses are rendered every ten frames. Black: key frames. Gray: generated transitions.

Abstract—In-betweening is a technique for generating transitions given start and target character states. The majority of existing works require multiple (often $\geq$ 10) frames as input, which are not always available. In addition, they produce results that lack diversity, which may not fulfill artists' requirements. Addressing these gaps, our work deals with a focused yet challenging problem: generating diverse and high-quality transitions given exactly two frames (only the start and target frames). To cope with this challenging scenario, we propose a bi-directional motion generation and stitching scheme which generates forward and backward transitions from the start and target frames with two adversarial autoregressive networks, respectively, and stitches them midway between the start and target frames. In contrast to stitching at the start or target frames, where the ground truth cannot be altered, there is no strict midway ground truth. Thus, our method can capitalize on this flexibility and generate high-quality and diverse transitions simultaneously. Specifically, we employ conditional variational autoencoders (CVAEs) to implement our autoregressive networks and propose a novel stitching loss to stitch the bi-directional generated motions around the midway point.

Extensive experiments demonstrate that our method achieves higher motion quality and more diverse results than existing methods on the LaFAN1, Human3.6m and AMASS datasets.

Index Terms—Animation, Transition Generation, In-betweening, Deep Learning

1 Introduction

Motion in-betweening, or keyframe interpolation, is a technique widely used in film production, video games, etc. Thanks to the introduction of deep learning techniques, modern motion in-betweening methods [11], [12], [13], [19], [29], [32], [34], [38] have achieved significant improvements in the naturalness and diversity of long-gap interpolation tasks, thereby significantly saving manpower and speeding up the animation production process.

Autoregressive models like LSTMs have become a natural choice for motion in-betweening due to their sequence modeling capabilities. However, they face two key challenges that conflict with each other: i) transition ambiguity, as there are infinitely many valid transitions between the start and target frames; and ii) constraints imposed by the target frame, which restrict the output sequence to

- *Tianxiang Ren, Jubo Yu, Shihui Guo, Ying Ma, Yutao Ouyang are with the School of Informatics, Xiamen University, Xiamen, China.*
- *Zijiao Zeng and Yazhan Zhang are with Tencent Technology.*
- *Yipeng Qin is with Cardiff University.*
- *Corresponding author: Shihui Guo (guoshihui@xmu.edu.cn)*

a fixed endpoint. Between them, the first challenge can be addressed by generative neural sequence models like variational autoencoders (VAEs) that can produce diverse outputs from the same input. However, increasing diversity makes restricting the output to a fixed endpoint more challenging, resulting in discontinuities that substantially degrade the visual quality. Existing solutions address this issue in two ways: i) Reduce transition ambiguity by using dense start frames ($\geq$10) [19], [29], [32], [34]. Transformers are commonly used in these solutions to effectively model the global context from the additional input. However, using such dense input comes at the cost of limiting diversity in the generated transitions. Moreover, acquiring 10 or more start frames (usually needs to be created by artists) is expensive and often infeasible in real-world applications. ii) Introduce post-processing techniques like blending to bridge the differences between the generated transitions and target poses. However, this post-processing has two key problems: Firstly, blending the transitional motion with the target motion can alter the original data, which animators may wish to preserve. Secondly, artifacts like foot sliding and body floating can be introduced, which are unacceptable for high-quality results. These artifacts are challenging to completely rectify. In some cases, manual cleanup by professional animators is needed to fix these issues, an often time-consuming process. To the best of our knowledge, there are currently no methods capable of simultaneously generating diverse, high-quality transition motions.

Addressing the gap mentioned above, in this paper, we propose a bi-directional motion stitching scheme, which relaxes the strict constraint imposed by the target frame (fixed endpoint) to a loose one (same connecting motion from both sides) at the midway point of the transition. Specifically, given sparse input frames (*i.e.*, only one start and one target frames), our method first generates forward and backward motion sequences from the start and target frames respectively, and then stitch them together (*i.e.* "align" them) in the middle of the transition, where there is no strict ground truth. A byproduct of our method is the *zero* error at the target frame, yielding exceptionally smooth and natural transitions that previous methods have never achieved. To implement our bi-directional scheme, we use two conditional variational autoencoder (CVAE) networks to build the mapping between motion data and their corresponding latent spaces, and generate the forward and backward motion sequences by sampling in their respective latent space. We argue that CVAE is well-fitted to our bi-directional scheme as it diversifies motion generation with the randomness in its sampling process, and thus capable of successfully modeling the diversity of transition animations. We then implement the stitching by identifying a pair of latent codes that minimizes a novel stitching loss in the two latent spaces of the CVAEs, respectively. In addition, we adapt CVAE to our stitching task with several novel techniques (*i.e.* Stitching-CVAE), including latent interpolation, bi-directional aligning and phase modulation.

Our contributions are summarized as follows:

- We propose a novel bi-directional stitching scheme for diverse and high-quality motion in-betweening from spare keyframes (*i.e.,* one start and one target frame). Our method generates highly smooth and natural transitions with *zero* error at the target frame, an achievement not attained by previous methods.
- We propose a novel Stitching-CVAE network that adapts CVAE to our stitching task with several novel techniques, including latent interpolation, bi-directional aligning and phase modulation.
- Extensive experimental results on the LaFAN1, Human3.6m and AMASS datasets justify the effectiveness of our method in natural and diverse motion in-betweening.

2 Related Work

2.1 Motion Prediction

Motion prediction generates future frames of motion based on the character states in the past few frames. Motion prediction tasks can be divided into deterministic and stochastic prediction. In deterministic motion prediction, existing works often use recurrent neural network (RNN) or their variants to capture temporal dependencies [16], [27], [45]. Researchers [7] proposed two LSTM-based structures to model temporal patterns and learn feature representations of sequences. Another work proposed structured RNN (S-RNN), a stacked RNN structure incorporating human motion semantic information [16]. S-RNN captures rich human-object interactions and makes significant improvements on human motion modeling. However, the RNN-based methods may cause frame skipping (the last input frame is not continuous with the predicted first frame) and face the problem of model collapse, which leads to average movements when capturing long-term dependencies [4], [8], [40]. PFNN [14] strengthens the control of character animation by introducing the phase feature and abandons traditional RNN-based methods. The phase feature was crafted to indicate the current motion cycle, eliminating motion ambiguity. [35], [36], [37] improved the phase feature and achieved more robust motion prediction. The graph-structure is typically used to represent skeletons. Graph Convolution Network (GCN) [6], [50] is employed to more effectively model the movement spatial relationships among skeleton parts in motion prediction. Mao *et al.* [25] firstly utilized GCN to exploit motion patterns to predict the future motions. It treats a human pose as a generic graph and designs a new GCN to learn the graph connectivity automatically. It leverages discrete cosine transform (DCT) to encode temporal information. Rather than employing DCT for encoding motion sequences, Ma *et al.* [23] used two separate GCNs to extract spatial and temporal features.

Compared with deterministic motion prediction, the result of stochastic motion prediction is not required to be close to ground truth [3], [10], [13], [20], [22], [39], [44]. It is required to generate diverse results given the same input. CVAE is widely used in stochastic motion prediction for its ability to learn data distribution and generate diverse results by sampling [2], [17], [31], [49]. A recent work [49] used CVAE for stochastic motion prediction, using marker-based locations instead of joint positions as human state representation and skinned multi-person linear model (SMPL) to generate more realistic human motions. A few works [17], [31] combined transformer with VAE to perform prediction in parallel and achieved an excellent performance. The

introduction of discrete cosine transform (DCT) improves the diversity of stochastic motion prediction [17].

In this work, we follow best practices from previous works and employ CVAE to model the diversity of transitional motions.

2.2 Transition Generation

The goal of transition generation (motion in-betweening) is to interpolate between two separate frames or motion clips. More priors are given in transition generation task than motion prediction, including past few frames and target frames information. In general motion prediction, only past few frames are provided without the constraints of target frames. In earlier research, [33], [42] adopted a physics-based strategy to generate motion between keyframes by solving an optimization problem with spatio-temporal constraints. Statistical models have also been used for generating transition animations, including Maximum A Posteriori (MAP) [28], Gaussian Process [41] and Markov models [21]. Over the past decade, deep neural networks have been applied to motion in-betweening. Based on the different historical frame lengths used, we categorize the learning based methods into single-frame and multi-frame required in-betweening methods.

Single-frame required in-betweening methods. RNN has been demonstrated to have excellent performance in time series prediction. [48] utilized an RNN conditioned on keyframes to generate jumping motions for a 2D model. [11] used RNN to generate transitions. As a following work, [12] proposed ERD with GAN network to achieve variable-length transition generation, with the assistance of time-to-arrival and scheduled-target embeddings. [38] proposed a new natural motion manifold model and a new transition sampler for real-time motion in-betweening. It increases the controllability of the in-betweening synthesis, and achieves good performance and high motion quality. But it can not guarantee tracking of target and its results lack diversity, especially of its lower body. RNNs are often used in conjunction with an autoregressive approach for motion in-betweening. The autoregressive approach can conduct motion in-betweening starting with only one historical frame. We call it single-frame required methods.

Multi-frame required in-betweening methods. However, the majority of current methods require multiple historical frames for better results. Methods in image inpainting have been applied to transition generation, considering the similarity between two tasks [13], [51]. These methods transformed time-series motion data into two-dimensional image-like features. Researchers proposed to apply progressive learning to transition tasks and gradually increase the length of transition during training to accelerate it [18]. However, this conversion of motion sequences into images lacks interpretability, and commonly produces artifacts such as jittering and foot sliding. Another work [46] only interpolates the body joint trajectory and generates the corresponding pose based on the interpolated trajectory. It generates animations for hundreds of characters simultaneously. [43] used a global and local hierarchical model for transition generation. First, it uses the route information to find small fragments to fill the gap through motion matching. It then generates the transition between each neighboring short sequence. Finally, Bi-LSTM predicts the transition between short sequences, and the prediction results are blended.

Recently, Transformer-based methods have proven its effectiveness in in-betweening. [34] use Transformer encoder and 1D temporal convolution to generate transitions. [29] use a Transformer-based Encoder-Decoder structure to generate transitions in delta mode. The delta means the offset between the spherical linear interpolation (Slerp) between keyframes and ground truth. [32] employs a two-stage generation process. One context transformer performs the first interpolation, followed by a refinement using one detail transformer structure. This approach excels in generating longer transition animations. All Transformer-based methods are trained with multiple (often ≥ 10) past frames as input and can't handle the extremely sparse cases, where there are only one past frame and one target frame given. However, methods requiring multiple frames exhibit a noticeable performance drop when the available historical frames are reduced. It restricts the use of these methods in practical scenarios.

Our work falls under the single-frame required in-betweening methods. Our key idea is to relax the strict constraint imposed by the target frame (fix endpoint) to a loose one (same connecting motion from both sides) at the middle point of the transition, thus generates diverse and high-quality transitions at the same time.

3 METHOD

3.1 Data Formatting

Given one start keyframe f_0 and one target keyframe f_L, our method generates intermediate transitions $\{\hat{f}_t\}_{t=1}^{L-1}$. The pose of each keyframe is composed of the 3-dimensional global position of the root joint r_t and local quaternion vectors q_t for the other joints relative to the root joint. t represents the timestep index. We extract feet contact information as a binary vector c_t of 4 dimensions when working with the LaFAN1 dataset. We also calculate the offset vectors o_t^r and o_t^q containing respectively the global root position's offset and local-quaternions' offsets from the target keyframe at time t [12]. The offset vector is the element-wise linear difference between the current pose and the target pose. When using the forward kinematics (FK) loss, we get the global positions of all joints $\hat{p}_{t+1}$ with predicted global root position $\hat{r}_{t+1}$ and local quaternions $\hat{q}_{t+1}$ by performing FK.

3.2 Motion Stitching Scheme

Figure 2 shows the diagram of our motion stitching scheme. Previous uni-directional methods [11], [12], [38] synthesized the motion sequence from the start frame to the target frame. We propose a new framework that *bi-directionally* synthesizes the motion sequence from both the start and the target frame simultaneously, and blends them in the *intermediate* region. This is similar to the procedure of *stitching* in the domain of garment making, in which edges of two clothes are sewn together.

As Figure 2 shows, we implement the proposed scheme with two generators: a forward generator $G_f(\cdot)$ synthesizing the forward motion sequence from the start frame

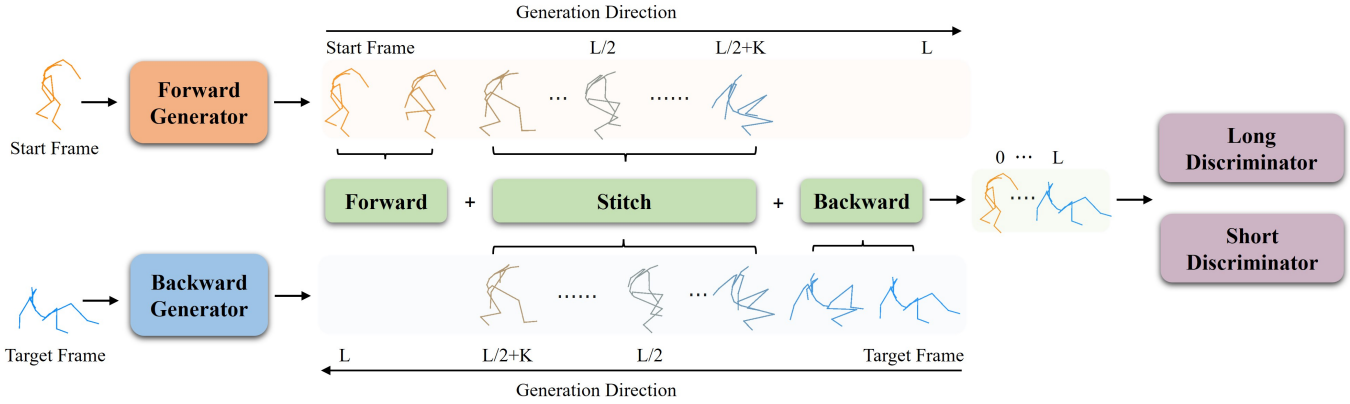


Fig. 2: Illustration of our bi-directional motion stitching scheme. We synthesize forward and backward motion sequences from the start and target frame respectively and stitch them together in the intermediate region. We also employ a pair of long-short discriminators [12] to improve the naturalness of synthesized motions. The black boxes in the figure mean there are no frames at these timestamps.

and a backward generator $G_b(\cdot)$ synthesizing the backward motion sequence from the target frame. Let L be the frame length of the entire transition period, K be the length of the synthesis buffers allowing for smoother blending results, we first make the two generators synthesize a motion sequence of length $L/2 + K$ each and linearly blend the overlap of the two sequences at each timestamp. We then concatenate the blended results with the remaining parts of both the forward and backward sequences to obtain the final motion sequence. To improve the naturalness of the synthesized motion sequence, we further employ a pair of long-short discriminators [12] $D_{ls}(\cdot)$ to enhance the transition details.

In contrast to uni-directional methods, our bi-directional scheme eliminates the necessity of trade-off between naturalness and fidelity for motion blending by shifting the blending operation from the target frame to the middle of the transition. Since the middle part of the transition is far from the strict motion ground truths at the start and target frames, the fidelity requirement is significantly relaxed and we only need to ensure that the blended motions are natural at the intermediate frames. In other words, our framework allows diverse motions to be blended, constituting a large and diverse motion space in the middle of the transition.

The application of the proposed bi-directional scheme is non-trivial as it requires efficient exploration of a large and diverse motion space, which poses a challenge for the design of the motion generators. We tried to build the bi-directional scheme with the model proposed in [12]. But the stitching result is terrible (shown in section 5.3.3). It's because the method can't generate diverse results. If generated motion sequences in forward and backward directions differ significantly in the synthesis buffers, the stitching results will be awful. To tackle this, we propose a novel stitching-CVAE (S-CVAE) network as described in the following section. Forward and backward generators are two independent S-CVAE structures and don't share the same parameters.

3.3 Stitching-CVAE

Similar to the vanilla CVAE, stitching-CVAE consists of an encoder and a decoder: the encoder encodes the character state of the current frame and the target frame, and maps them to a latent code z; the decoder decodes z sampled in the latent space and generates the character state in the next frame. We adapt the vanilla CVAE to our stitching task by re-designing its encoder.

Figure 3 shows the architecture of the S-CVAE encoder. In S-CVAE, the encoder involves three inputs: the current frame, the target frame and their offset. We train the model in two stages. Firstly, we concatenate the embeddings of the current frame, the target frame and their offset and feed it into the LSTM. Then, we pass the LSTM output through a fully-connected network to obtain the current latent space distribution $\mathcal{N}(\mu, \theta)$. Then, we replace the current frame with the target frame so that the offset is 0, then recalculate the connection embedding and feed it into the LSTM again. Then, we pass the LSTM output through another fully-connected network to obtain the latent space distribution for the target frame $\mathcal{N}(\mu_t, \theta_t)$. Note that this stage indicates the arrival of the current frame at the target frame, thus producing the target latent space distribution. Finally, we perform a latent interpolation operation on the above two latent space distributions.

To adapt the encoder to our stitching task, we propose several novel techniques as follows.

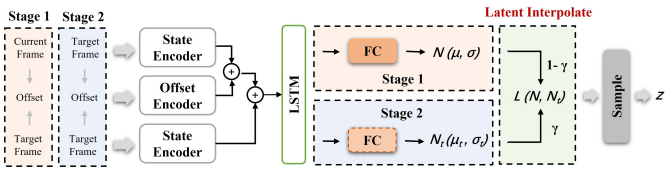


Fig. 3: Illustration of the encoder of the stitching-CVAE. The latent interpolation operation linearly blends the distributions of the current frame and the target frame. Orange: stage 1; Blue: stage 2.

Latent Interpolation (Figure 3). To facilitate stitching, we design a Latent Interpolation operation to linearly blend the latent distributions of the current frame and the target

frame:

$$L(\mathcal{N}, \mathcal{N}_t) = (1-\gamma)\mathcal{N}(\mu, \theta) + \gamma\mathcal{N}(\mu_t, \theta_t) \tag{1}$$

$$\gamma = \begin{cases} t/\lfloor L/2 \rfloor, & 0 <= t < \lfloor L/2 \rfloor \\ t/(\lfloor L/2 \rfloor + K), & \lfloor L/2 \rfloor <= t <= \lfloor L/2 \rfloor + K \end{cases} \tag{2}$$

where $\mathcal{N}(\mu, \theta)$ denotes the distribution of the current frame and $\mathcal{N}(\mu_t, \theta_t)$ denotes the distribution of the target frame. γ linearly increases from 0 to 1 (0 at the start of the transition(0), 1 at the middle of the transition($\lfloor L/2 \rfloor$)) with the latest-of-opposite-generator as the target frame. When the target frame switches back to the end-of-transition(L), it linearly increases again from a lower value to 1 as illustrated in Eq.2. This ensures that reasonable weights are assigned to the forward and backward generators at different stitching positions.

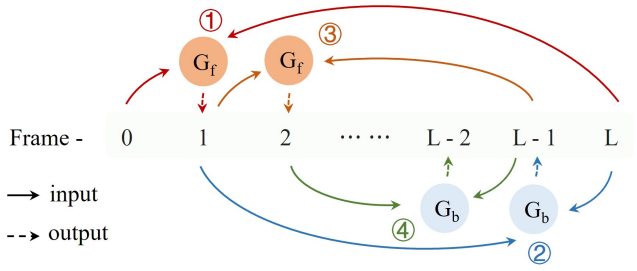


Fig. 4: Illustration of bi-directional aligning. $G_{f/b}(\cdot)$ represents the forward/backward motion generator. The ① represents the i-th step generation.

Bi-directional Aligning (Figure 4). Starting from the start and target frames, we generate new frames in turn using the forward and backward generators respectively. The process can be described as below:

$$\begin{aligned} f_{i+1} &= G_f(f_i | f_j) \\ f_{j-1} &= G_b(f_j | f_{i+1}) \end{aligned} \tag{3}$$

where f_i represents the i-th frame. $G_{f/b}(\cdot)$ represents the forward/backward generator. $G(a|b)$ means the generator generates the next frame with the a-th frame as current frame and the b-th frame as the target frame.

To facilitate the stitching when the two sequences meet, we condition the generation of the current frame with the latest frame synthesized by the other generator, thereby aligning the generation processes of the forward and backward motion sequences together. After the current frame crosses the current target (middle of the transition), the forward generator conditions on the last frame of the transition instead of the backward generator's last output. The same to the backward generator.

Stitching Loss. We design a stitching loss as the average L1 distance of the overlap of the two generated sequences, which regularizes the two sequences to be consistent with each other:

$$\mathcal{L}_{stitch} = \frac{1}{2K} \sum_{t=\lfloor L/2 \rfloor - K + 1}^{\lfloor L/2 \rfloor + K} \left\| \mathbf{p}_t^f - \mathbf{p}_{L-t}^b \right\|_1 \tag{4}$$

where $\lfloor \cdot \rfloor$ is a floor function, L is the length of sequence generated by each motion generator, $\mathbf{p}^f$ and $\mathbf{p}^b$ represent the global positions of the forward and backward motion generators calculated by forward kinematics (FK). K is the length of the synthesis buffers allowing for smoother blending results.

We also adapt the decoder to our stitching task with a novel phase modulation technique.

Phase Modulation. Observing the periodicity of many common motions, we propose that the incorporation of phase information can eliminate action ambiguity, improve animation quality, and reduce flutter. Specifically, we use a phase prediction network to extract the phase information from the current frame and use it to modulate the CVAE decoder. The phase prediction network is pre-trained on a dataset labelled using local phase method introduced in [36], in which we can automatically extract phase variables at local level. It takes local rotations in quaternions, the root velocity, foot contact information and the phase value at the current frame as input and outputs the phase updates. The phase value is updated in an auto-regressive manner as in [36]. It indicates which phase of a motion cycle the character is currently in and helps to generate accurate motions.

Remark. S-CVAE not only increases the diversity of the results, but also facilitates stitching with its diverse motion space. Specifically, the forward and backward sequences can be smoothly stitched if we can find a pair of matching latent codes in their corresponding motion spaces. The more diverse such motion spaces, the higher likelihood that we can find such a pair of latent codes.

3.4 Overall Loss Function

In addition to the stitching loss (Eq. 4), we use several other loss functions to constrain the learning process to guarantee the stability of the training and the quality of generated results. Since in our method, the character state is represented by its global root position and local rotations of other joints relative to their parent joints respectively, we denote the local rotations in the form of quaternions as $\mathbf{q}_t$, the root joint velocity as $\mathbf{v}_t$, the foot contact information extracted using the method provided in LaFAN1 [12] as $\mathbf{c}_t$, and define the loss functions as follows.

State Loss. State loss represents the reconstruction loss of three different types of character state. It consists of quaternion loss, root velocity loss, and contact loss. Each loss is a L1 norm between the predicted results and the ground truth. The losses are summarized weighting by β_1, β_2 and β_3, and averaged across all time frames. The state loss function is:

$$\begin{aligned} \mathcal{L}_{state} = \frac{1}{L} \sum_{t=0}^{L-1} (\beta_1 \left\| \hat{\mathbf{q}}_t - \mathbf{q}_t \right\|_1 + \beta_2 \left\| \hat{\mathbf{v}}_t - \mathbf{v}_t \right\|_1 + \\ \beta_3 \left\| \hat{\mathbf{c}}_t - \mathbf{c}_t \right\|_1) \end{aligned} \tag{5}$$

KL Loss. As common in CVAE, we regularize the posterior distribution to normal distribution by optimizing the Kullback-Leibler divergence:

$$\mathcal{L}_{kl} = KLD(q(\boldsymbol{Z} \mid \boldsymbol{X}_t) || \mathcal{N}(0, \boldsymbol{I})) \tag{6}$$

where $q(\cdot \mid \cdot)$ denotes the inference posterior (encoder).

FK loss. FK loss is proposed in [30] to alleviate the accumulative errors of rotations in local coordinates. We calculate global positions by local quaternions with forward kinematics (FK) and get the average L1 norm between the calculated positions and real positions. The FK loss function is:

$$\mathcal{L}_{fk} = \frac{1}{L}\sum_{t=0}^{L-1} \|FK(r, \hat{\mathbf{q}}_t) - \mathbf{p}_t\|_1) \tag{7}$$

where r represents the global root position.

Adversarial Loss. We use a generator-discriminator architecture and employ a pair of long-short discriminators [12] to improve the motion quality. The discriminator is in the form of Least Square GAN [26]. Each discriminator takes different lengths of generated motions and ground truth motions as input. The adversarial loss function is defined as follows:

$$L_G = \frac{1}{2}\mathbb{E}_{Z\sim p_Z}\left[\left(D\left(G\left(Z\right)\right)-1\right)^2\right] \tag{8}$$

$$\begin{aligned} L_{\mathrm{D}} =& \frac{1}{2}\mathbb{E}_{\mathrm{X}\sim p_{\mathrm{Data}}}\left[\left(D\left(\mathrm{X}\right)-1\right)^2\right] + \\ & \frac{1}{2}\mathbb{E}_{\mathrm{Z}\sim p_Z}\left[\left(D\left(G\left(Z\right)\right)\right)^2\right] \end{aligned} \tag{9}$$

where X and Z represent the ground truth frames and sampled latent codes respectively. G is the transition generator network. D is the discriminator network.

Overall Loss Function. The overall loss is made up of i) the average of the forward and backward losses consisting of their own state, KL and FK losses respectively ii) a stitching loss and an adversarial loss:

$$\begin{aligned} \mathcal{L} =& \mathcal{L}_{state} + \alpha_1\mathcal{L}_{kl} + \alpha_2\mathcal{L}_{stitch} + \alpha_3\mathcal{L}_{fk} + \\ & \alpha_4\mathcal{L}_D + \alpha_5\mathcal{L}_G \end{aligned} \tag{10}$$

Please see Sec. 4.3 for the choices of weights.

4 Implementation Details

4.1 Network Details

Encoder. We use separate encoders to learn from different state features, including a state encoder, an offset encoder, and a target encoder. All encoders consist of three fully-connected layers. The inputs of the encoders are in different sizes as described below. The hidden size of all of them is 512, and the output size is 256. The state encoder encodes the character state of the current frame, including local rotation information in the form of quaternions, foot contact information, and velocity information of the root joint. The input size of state encoder is 95 on LAFAN1 datasets. It is different when using different datasets, because they have various skeletons with different joint numbers. The velocity information of the root joint plays an essential role in alleviating the mode collapse problem of LSTM in our experiments. The offset encoder encodes the local quaternion offset and the global root position offset between the current frame and the target frame. In the in-betweening task, the prior information of the offset between the current and target frames is critical [12]. The input size of offset encoder is 91. The target encoder encodes the state of the target frame, which is taken as the condition signal of CVAE to guide the prediction of the generator. The input size of target encoder is 88. To make networks aware of the time until target, we achieve time encoding by introducing the time-to-arrival embedding [12] in our method. The time-to-arrival embedding has 256 dimensions and is added to all input embeddings separately. With the time-to-arrival embeddings, our method is able to gracefully handle transitions of variable lengths. It can be defined as:

$$\mathbf{z}_{tta,2i} = \sin\left(\frac{tta}{\text{basis}^{\,2i/d}}\right) \tag{11}$$

$$\mathbf{z}_{tta,2i+1} = \cos\left(\frac{tta}{\text{basis}^{\,2i/d}}\right) \tag{12}$$

where tta is the timesteps until the target. The second subscript of the vector $\mathbf{z}_{tta,\_}$ represents the dimension index. d represents the the dimensionality of the input embeddings. $basis$ influences the rate of change in frequencies along the embedding dimensions. We set it to 10,000 as in [12].

All output embeddings of the encoders are concatenated as the input embedding of LSTM, which helps to capture temporal dependencies. The hidden size of LSTM is 768. And then we use two fully-connected layers of width 768 and 16 to get the distribution of the current and target frame separately. Finally, the distributions will be blended by the linearly blending operation to form the final latent variable space.

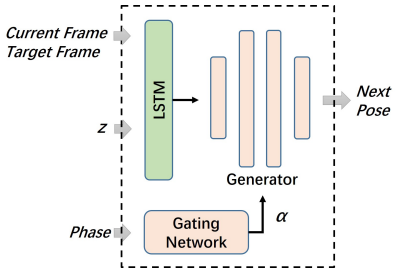


Fig. 5: Illustration of S-CVAE decoder.

Decoder. The decoder is composed of LSTM and mixture of experts (MoE [47]) network as illustrated in Fig. 5. Its architecture llustration is shown in the supplementary material. It takes the random latent variable $\mathbf{z}$ and the character state of the current frame and the target frame as input to predict character states in the next frame. The predicted character states include the root joint velocity, the quaternion updates of the other joints and the foot contact information. We predict local-quaternion updates and root velocities to prevent the frame skipping problem in RNN as in [27]. The local-quaternion updates are element-wise linear differences of quaternions at frame t and frame t+1.

All feature embeddings are concatenated as the input embedding of LSTM of width 528. MoE network includes a gating network and multiple expert networks. We set the number of experts as 4 The gating network generates a set of blending coefficients α suitable for the current motion according to the phase and then blends the weights of multiple expert networks to form the generator. The gating network and expert networks are based on a multi-layer perception (MLP) model. The gating network is a four layers fully-connected networks. The size of each layer is 4, 64, 32

and 4. All expert networks are also all four layers fully-connected networks, with sizes of 768, 512, 256 and 88.

Discriminators. The pair of long-short discriminators are two variants of a relatively simple feed-forward architecture [12]. Each of the discriminators is comprised of three fully-connected layers, with the final layer serving as a 1D linear output layer. The long discriminator examines consecutive motion frames in sliding windows of 10 frames, while the shor one examining in sliding windows of 2 frames. The size of each layer is 512 and 256. To generate a single scalar loss, we calculate the average of discriminator scores over time.

4.2 Datasets

We train and evaluate our model on three public datasets[1], LAFAN1 [12], Human3.6m [15] and AMASS [24]. The experiments result on AMASS dataset is shown in the supplementary material.

Human3.6m. Human3.6m is a large-scale dataset with diverse action types, often used for motion prediction and pose estimation. It contains the data of 7 subjects performing 15 types of actions, including "Direction", "Sitting", "Sitting Down", "Walking", "Taking Photos", "Smoking" and "Eating", etc. Following the standard setting in [1], [12], we take subject1, subject5, subject6, subject7, and subject8 as training sets and subject9 and subject11 as test sets. We refer to the experimental setting of RMIB [12] and use data of specific action types for training, including walking, walking-dog, and walking-together. The other action types are short-term ones that are not suitable for long-term motion prediction. To adapt the motion sequences to our motion in-betweening task, we create the training and test sets by sampling the sequences with a window size of 50 and a offset of 20. Our resulting training set contains 8,451 motion fragments and our test set contains 2,635 fragments.

LAFAN1. LAFAN1 dataset contains 78 long motion sequences performed by 5 subjects, consisting of 496,672 frames sampled at 30Hz. Following RMIB [12], we take subject1, subject2, subject3, and subject4 as training sets and subject5 as the test set. Similar to Human3.6m, we create the training and test sets by sampling the sequences with a window size of 50 and a offset of 20. Our resulting training set contains 20,212 motion fragments and our test set contains 2,232 fragments.

4.3 Training Details

We conduct experiments on a PC with a Intel i7-7700 CPU and a Nvidea TESLA P40 GPU. We implement our method with PyTorch. We train our model using an AdamW optimizer with a learning rate η =0.0001, β_1=0.5, β_2=0.9, weight decay λ =0.00001, and batch size n_{batch}=32. We set the number of expert networks in MoE as 4. We use β_1=1.0, β_2=1.0, β_3=0.1 in Eq. 5 and $\alpha_1 = 1.0$, $\alpha_2 = 0.5$, $\alpha_3 = 0.5$, $\alpha_4 = \alpha_5 = 0.1$ in Eq. 10. To accelerate training, we adopt a progressive training strategy: we gradually increase the length of the transition by 1 for every 2 epochs, from 5 to 50, during training.

1. Note that we reverse the motion data to train our backward generator.

5 Experiments

5.1 Metrics

We evaluate motion in-betweening methods from three aspects: diversity, accuracy, and naturalness, using seven metrics. We provide detailed descriptions and equations for all metrics used in the paper in the supplementary material.

Average Pairwise Distance (APD): the average L2 distance of global positions of multiple motions generated under the same input and constraints.

Accuracy - Average Displacement Error (ADE): the average L2 distance of global positions between the reconstructed motion and the ground truth.

Accuracy - L2P: L2P reports average L2 distances of global global positions.

Accuracy - L2Q: L2Q reports average L2 distances of global quaternions.

Accuracy - Second to Last Displacement Error (SLDE): the average L2 distance of global positions between the *second to last frame* of the reconstructed motion and the ground truth.

Naturalness - Normalized Power Spectrum Similarity (NPSS [9]): the similarity of the distribution of the generated motion and the ground truth.

Naturalness - Foot Sliding per Frame (Foot Slide): the average sliding distance of the stance foot, *i.e.*, the ankle and toe joints, per frame.

Among them, SLDE complements ADE by highlighting the accuracy of motion reconstruction at the second to last frame where the ground truth requirement is strict; NPSS and Foot Slide show the naturalness of motions from both the statistics distribution and critical events respectively.

5.2 Qualitative Experiments

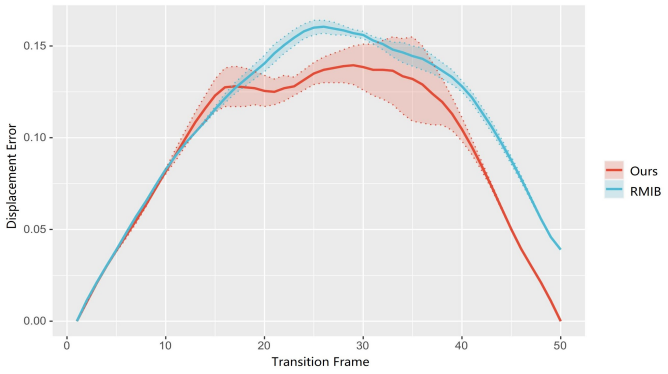


Fig. 6: Displacement error along time. Blue shadow: the displacement error of results generated three times by RMIB [12]. Red shadow: the displacement error of results generated three times by our method. The test is conducted across the entire test set of the LaFAN1 dataset. The solid lines represent the average displacement error. The bigger shadow area means that our method can generate more diverse results.

Diversity of Motion In-betweening. As Fig. 6 shows, our model produces more diverse transitions with the same inputs and constraints, especially at the middle of the transition. Additionally, our method generates more accurate

results. Because the FK loss converges to a lower level with the assistance of the stitching loss, the generated transitions are more similar to the ground truth transitions. The lack of smoothness in the curves is because the entire sequence is optimized in segments. The reconstruction loss is applied to the entire sequence, whereas the stitching loss only affects the intermediate buffer region, precisely corresponding to the less smooth part of the curve. The asymmetry in the curves is attributed to the different training data used by the forward and backward motion generators. Although both are trained on the same dataset, the backward motion generator uses reversed motion data. We also show the visible comparison results in Fig. 7.

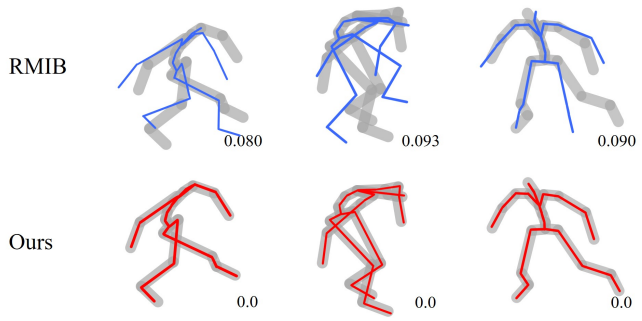


Fig. 7: Fidelity of motions generated at the end (target) frames by RMIB [12] (top) and our method (bottom). Blue and red skeletons: generated motions. Gray shading: ground truth. The numbers at the right-bottom corner of each sample are their corresponding FDE scores. FDE means final displacement error. It calculates the L2 distance between the pose in the last time step of ground truth motion and the motion from a generated set of K motions that is the closest to the ground truth.

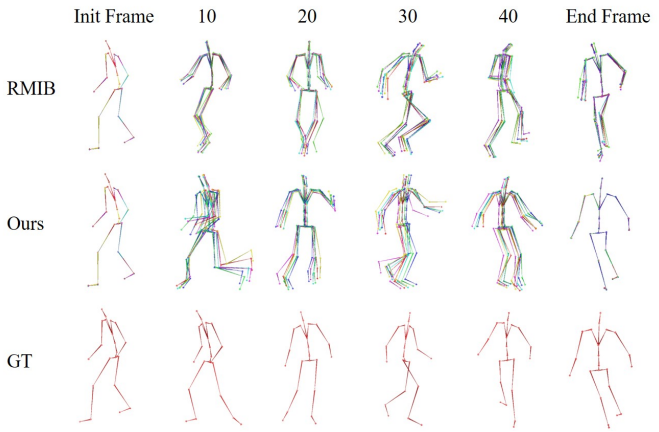


Fig. 8: Diversity of samples generated by RMIB [12] (top) and our method (middle) and ground truth transition (bottom). All samples are generated six times repeatedly with the same constraints and target motions.

Fidelity at the End (Target) Frame. As Fig. 7 shows, the proposed bi-directional stitching method guarantees a perfect fit at the end (target) frame, which resolves a longstanding challenge in previous methods [38]. Besides, the Fig. 6 shows the displacement error along with time. Our method achieves more diverse results and guarantees a perfect fit at the target frame.

5.3 Quantitative Experiments

5.3.1 Evaluation on the LaFAN1 dataset

As Table 1 shows, we compare our method with the classic interpolation method Slerp[2] [5], RMIB method [12], Δ-Interpolator [29], and τ_{det} [32] on three different lengths in-betweening motion generation tasks on the LaFAN1 dataset. Δ-Interpolator and τ_{det} both are Transformer-based motion in-betweening methods. τ_{det} reports state-of-the-art motion in-between benchmark results on LAFAN1. It consists of two Transformer Encoder-based networks (Context Transformer and Detail Transformer) operating in two stages. In the first stage the Context Transformer generates rough transitions based on the context and in the second stage the Detail Transformer is employed to refine motion details. This work proposed two in-between networks (τ_{con} and τ_{det}). τ_{con} only leverages the Context Transformer. τ_{det} leverages both the Context Transformer and Detail Transformer. The Detail Transformer is independent from the Context Transformer, so we try to involve it in our method (Ours$_{det}$) to refine motion details. All methods were trained and tested with exactly two frames given. Among them, the short-term one is considered "resolved" by Slerp as it is relatively simple due to its smaller number of possible motion variations. It can be observed that: i) For the short-term synthesis task, our method is comparable to Slerp in accuracy and naturalness, but with smaller Foot Slide scores, which demonstrate the effectiveness of our method in short-term motion in-betweening synthesis. ii) For the medium-term and long-term synthesis tasks, our method significantly outperforms Slerp and RMIB in accuracy, diversity and naturalness. Note that our method achieves a much smaller SLDE and a perfect alignment with the target frame. iii) The diversity (APD) of our method increases as the number of frames to be generated, which indicates that our method successfully captures the increasing number of possible motion variations with time.

The transformer-based methods can predict multiple missing frames within a single forward propagation. It achieves high speed performance and high quality with multiple past frames as input. However, as shown in Table 1, the quality drops dramatically when the number of past frames decreases to one. But ours performs well. The distinct difference between the transformer-based methods and ours is that ours address a more challenging but more valuable case where the past frames before the source frame are unavailable. Unlike transformer-based methods that use 10 such past frames, ours uses none but achieves comparable performances. In addition, our method can generate diverse results but the transformer-based methods can't.

Besides, we compared our methods with Δ-Interpolator [29] and τ_{det} [32], all Transformer-based in-between methods trained with 10 past frames. We show all the comparison results in Table 2, which shows our method is comparable

2. In Slerp, We interpolate the root position linearly and the quaternions spherically.

TABLE 1: In-betweening on the LaFAN1 dataset. We use different methods to generate three types of lengths transitions, given only 1 historical frame as input.

Frames	Method	APD ↑	L2P ↓	L2Q ↓	ADE ↓	SLDE ↓	NPSS ↓	Foot Slide ↓
10	Slerp	0.000	0.53	0.42	**0.030**	0.020	**0.006**	3.689
	RMIB	0.537	0.44	0.31	0.043	0.049	0.029	2.221
	Δ-Interpolator	0.000	0.95	0.83	0.203	0.033	0.041	3.319
	τ_{det}	0.000	0.68	0.72	0.141	0.020	0.035	2.936
	Ours	**1.415**	0.37	**0.30**	**0.032**	**0.012**	**0.008**	**2.146**
	Ours$_{det}$	1.407	**0.36**	0.30	0.032	0.013	0.008	2.231
30	Slerp	0.000	2.32	0.98	0.135	0.029	0.178	4.743
	RMIB	14.499	1.28	0.69	0.143	0.053	0.132	0.939
	Δ-Interpolator	0.000	4.47	3.11	0.793	0.039	0.220	2.343
	τ_{det}	0.000	3.58	2.33	0.517	0.023	0.184	1.896
	Ours	**25.123**	1.08	0.60	0.099	0.015	0.121	0.863
	Ours$_{det}$	24.244	**1.06**	**0.59**	**0.093**	**0.013**	**0.120**	**0.830**
50	Slerp	0.000	4.97	1.98	0.252	0.034	0.739	3.984
	RMIB	36.860	2.73	1.21	0.172	0.057	0.432	0.592
	Δ-Interpolator	0.000	6.27	4.39	0.938	0.036	0.613	2.946
	τ_{det}	0.000	5.22	3.96	0.807	0.021	0.557	2.539
	Ours	**63.269**	2.37	**1.13**	0.123	0.016	0.311	0.468
	Ours$_{det}$	62.899	**2.32**	1.13	**0.113**	**0.015**	**0.310**	**0.465**

to Transformer-based methods even if trained with only one past frame, even better than some of them. And our method combined with the Detail Transformer [32] gets the best L2P and NPSS score.

TABLE 2: Comparisons of different methods on LaFAN1 dataset. All Transformer-based methods are trained with 10 past frames as input, while RMIB and our method only use 1 historical frame as input.

Frames	Method	APD ↑	L2P ↓	L2Q ↓	ADE ↓	SLDE ↓	NPSS ↓	Foot Slide ↓
15	RMIB	4.322	0.65	0.42	0.097	0.055	0.0258	2.411
	Δ-Interpolator	0.000	0.47	0.32	0.073	0.016	0.0217	1.624
	τ_{det}	0.000	**0.39**	**0.28**	**0.070**	0.016	**0.0188**	**1.562**
	Ours	**8.153**	0.53	0.35	0.076	**0.014**	0.0223	1.627
	Ours$_{det}$	8.150	0.45	0.32	0.073	0.014	0.0201	1.601
30	RMIB	14.499	1.28	0.69	0.143	0.053	0.1328	0.939
	Δ-Interpolator	0.000	1.00	0.57	0.091	0.021	0.1217	0.845
	τ_{det}	0.000	**0.89**	**0.54**	**0.084**	0.019	**0.1124**	**0.792**
	Ours	**25.123**	1.08	0.60	0.099	0.015	0.1210	0.863
	Ours$_{det}$	24.244	0.99	0.57	0.093	**0.013**	0.1196	0.830
45	RMIB	28.773	2.24	0.94	0.158	0.032	0.3311	0.572
	Δ-Interpolator	0.000	3.23	1.15	0.145	0.024	0.4359	0.531
	τ_{det}	0.000	1.68	**0.87**	**0.106**	0.018	0.3217	**0.427**
	Ours	**55.748**	1.81	0.92	0.113	**0.015**	0.3051	0.468
	Ours$_{det}$	55.465	**1.59**	0.92	0.110	0.015	**0.3040**	0.452

We compared our method with τ_{con} and τ_{det} [32] with different input historical frames length. As shown in Fig.9, our method always performs better than τ_{con} and is comparable to τ_{det} when input historical frames decrease to 4.

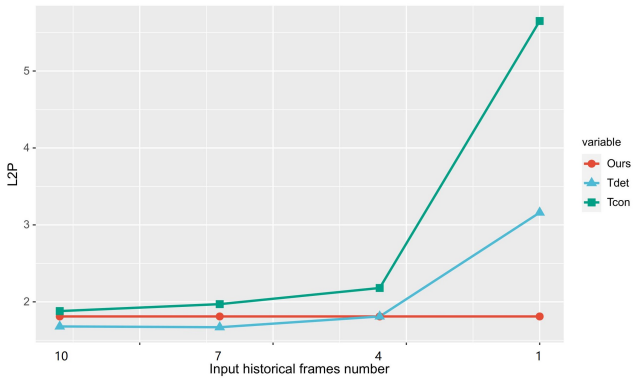


Fig. 9: Comparison with τ_{con} and τ_{det} with different input historical frames length. Our method only uses 1 historical frame as input.

TABLE 3: In-betweening on the Human3.6m dataset. We use different methods to generate three types of lengths transitions, given only 1 historical frame as input.

Frames	Method	APD ↑	L2P ↓	L2Q ↓	ADE ↓	SLDE ↓	NPSS ↓	Foot Slide ↓
10	Slerp	0.000	0.67	0.41	0.16	0.233	0.0054	3.239
	RMIB	0.823	0.48	0.38	0.217	0.598	0.0102	3.557
	Δ-Interpolator	0.000	1.01	0.97	0.611	0.325	0.0116	8.031
	τ_{det}	0.000	0.74	0.69	0.517	0.295	0.0108	7.283
	Ours	1.445	0.41	0.32	0.159	0.225	0.0059	3.002
	Ours$_{det}$	**1.449**	**0.40**	**0.32**	**0.154**	**0.218**	**0.0059**	**2.981**
30	Slerp	0.000	2.13	1.09	1.232	0.499	0.132	3.887
	RMIB	14.879	1.18	0.78	0.654	0.848	0.096	2.865
	Δ-Interpolator	0.000	4.07	3.52	3.259	0.593	0.192	7.874
	τ_{det}	0.000	3.31	2.59	2.547	0.328	0.16	7.084
	Ours	26.314	1.07	0.69	0.501	0.273	0.087	2.412
	Ours$_{det}$	**26.423**	**1.01**	**0.67**	**0.494**	**0.265**	**0.085**	**2.395**
50	Slerp	0.000	2.55	1.05	1.552	0.532	0.356	2.352
	RMIB	36.069	1.68	0.93	0.851	0.904	0.252	1.550
	Δ-Interpolator	0.000	3.85	3.37	4.543	0.654	0.452	7.428
	τ_{det}	0.000	3.20	3.05	3.784	0.382	0.421	7.101
	Ours	**63.751**	1.35	0.91	0.622	0.291	0.229	1.309
	Ours$_{det}$	63.387	**1.31**	**0.90**	**0.601**	**0.273**	**0.228**	**1.300**

5.3.2 *Evaluation on the Human3.6m dataset*

We evaluate our method on the Human3.6m dataset using the same setups as those on the LaFAN1 dataset. As Table 3 shows, similar to the results on the LaFAN1 dataset, our method achieves the best scores in APD, ADE, SLDE, NPSS and Foot Slide, which demonstrates that our method outperforms previous methods in all accuracy, diversity and naturalness metrics.

As shown in 4, we also tested the performance of the model trained on the LaFAN1 dataset using the Human3.6M test set. To enable the model trained on LaFAN1 dataset to be tested on different datasets, we retargeted the motions from Human3.6m to the LaFAN1 skeleton. The results show that our approach generalizes well across datasets.

TABLE 4: Evaluate our method trained on LaFAN1 dataset (Ours_la) on Human3.6m test set.

Frame	Method	APD ↑	L2P ↓	L2Q ↓	ADE ↓	SLDE ↓	NPSS ↓	Foot Slide ↓
10	Ours_la_hm	1.033	0.46	0.39	0.045	0.015	0.017	2.336
	Ours_la_la	1.407	0.37	0.30	0.032	0.013	0.008	2.231
30	Ours_la_hm	23.725	1.97	0.82	0.121	0.030	0.137	1.257
	Ours_la_la	25.123	1.08	0.60	0.099	0.015	0.121	0.863
50	Ours_la_hm	60.237	3.12	1.55	0.153	0.028	0.351	0.591
	Ours_la_la	63.269	2.37	1.13	0.123	0.016	0.311	0.468

5.3.3 *Implementation of Bi-directional Framework*

We tried to build the bi-directional framework with the model proposed in [12] and our method. As shown in Table 5, the stitching result of the bi-directional scheme with RMIB [12] is terrible. It's because RMIB can't generate diverse results. If generated motion sequences in forward and backward directions differ significantly in the synthesis buffers, the stitching results will be awful. But our method can provide much more diverse motion spaces and find such a pair of latent codes to make smooth stitching.

TABLE 5: Implement Bi-directional Scheme with different methods.

Method	APD ↑	ADE ↓	SLDE ↓	NPSS ↓	Foot Slide ↓
Bi-directional in RMIB	36.319	0.254	0.019	0.573	1.627
Ours	**63.269**	**0.123**	**0.016**	**0.311**	**0.468**

5.4 Ablation Study

We perform an ablation study on the LaFAN1 dataset to explore the effectiveness of each module. We conduct a 50-frame length transition generation on the following baselines:

- Ours w/o BS (Bi-directional Scheme): directly generate the transition only in the forward direction;
- Ours w/o SL (Stitching Loss): train the network without the stitching loss.
- Ours w/o LI (Latent Interpolation): directly sample latent code from the latent space of the current frame.
- Ours w/o BA (Bi-directional Aligning): don't take the opposite generator's last prediction as target frame. Keep the given target frame as the target.
- Ours w/o PM (Phase Modulation): replace the MoE module with MLP.
- ours w/o D: generate the transition with out discriminators;
- ours w/o LSTM: replace LSTM in the encoder with MLP.

In ablation study, We add a metric, namely Mean Middle Pose Error (MMPE), to measure the pose error between the two different generated poses at L/2. It is defined as:

$$MMPE = \left\| \widehat{\mathbf{p}}^{f}_{L/2} - \widehat{\mathbf{p}}^{b}_{L/2} \right\|_2 \tag{13}$$

where $\widehat{\mathbf{p}}^{f}_{L/2}$ and $\widehat{\mathbf{p}}^{b}_{L/2}$ represent the generated global positions at L/2 generated by the forward and backward generator.

TABLE 6: Ablation Study on LAFAN1 dataset.

Method	APD ↑	ADE ↓	SLDE ↓	NPSS ↓	Foot Slide ↓	MMPE ↓
Ours (w/o BS)	60.467	0.190	0.237	0.419	0.548	/
Ours (w/o SL)	63.476	0.130	0.018	0.313	0.517	0.0371
Ours (w/o LI)	**71.260**	0.126	0.017	0.318	0.493	0.0184
Ours (w/o BA)	62.443	0.124	0.018	0.315	0.541	0.0353
Ours (w/o PM)	62.880	0.128	0.016	0.325	0.554	0.0143
Ours (w/o D)	34.647	0.207	0.016	0.453	1.436	0.0154
Ours (w/o LSTM)	60.329	0.154	0.018	0.373	0.627	0.0145
Ours	63.269	**0.123**	**0.016**	**0.311**	**0.468**	**0.0142**

The results are shown in Table 6. It can be observed that: 1) The bi-directional schema solves the problem of generated results not fitting with the target frame. 2) S-CVAE increases the diversity of results. The integrated bi-directional aligning, stitching loss, and latent interpolate operation lead to a smooth stitching result. Additionally, latent interpolating contributes to more natural results at the expense of harming the diversity of results due to its average operation. 3) The phase modulation is beneficial for improving motion quality. Introducing the phase into in-betweening tasks is a good practice. 4) LSTM and discriminators are beneficial for improving motion quality and discriminators contribute to increasing the diversity of results. 5) Though latent interpolation process hurts the diversity, we still keep it. Our aim is to increase the diversity without hurting quality, which is achieved by the proposed latent interpolation. And our method outperforms RMIB significantly even when latent interpolation is applied. 6) It can be observed that the novel techniques proposed in our work effectively reduce MMPE, which suggests that they bring the two generated poses closer together at L/2, resulting in better stitching results.

6 Limitations and Future Work

In this work, we focus on improving the diversity of transition animations rather than how to control the generation process of them. Our method does not allow intuitive control of the generation process due to the random sampling method used in the CVAE latent space. We hope to explore the use of high-level semantic information (*e.g.* motion styles or action types) to control the generation in future work, which will meet the animators' wills better.

A major limitation of our approach is that it inherits the inherent challenges of data-driven methods and does not generalize well to rare or unseen motions (*e.g.*, turning around and pushing adversaries), *a.k.a*, the imbalanced dataset problem.

Our method is frame rate-sensitive – it can only generate results at the same frame rate as the training set. If we want a model capable of generating results at different frame rates, we have to retrain it on datasets with different frame rates. We demonstrated it through experimentation (included in the supplementary material). We think it may be due to the influence of different frame rates on position encoding. An in-betweening method effective for different frame rates would be worth exploring in future work.

Last but not least, in inference, the length of the transition is determined by the user. However, animators can not directly judge how long transition are in real cases. So we need the model to help us to predict lengths of transitions, which is helpful when applying in real cases.

7 Conclusion

In this paper, we propose an in-betweening method. It can generate diverse, high-quality transition motions in extreme cases where only two frames are given (one past frame and one target frame). Our method generates the forward and backward segments, respectively from both ends, and then stitches both segments at the intermediate seam region. This strategy solves the problem that the generated transitions deviate from the target frames. Experiments demonstrate that our method can generate more diverse and higher-quality results than previous work on both long and short sequences. The success of our method is rooted in the elegant design of bi-directional generation and intermediate stitching. Components in this complete recipe are indispensable in order to satisfy the *conflicting* requirements of both motion diversity and conformity.

Acknowledgments

The authors would like to thank Shihui Guo, Jubo Yu, Zijiao Zeng, Yazhan Zhang, Ying Ma, Yutao Ouyang, and Yipeng Qin for their contributions to this study. This work is supported by National Natural Science Foundation of China (62072383, 61702433), the Fundamental Research Funds for the Central Universities (20720210044), and Royal Society (IEC\NSFC\211022).

REFERENCES

[1] Haoye Cai, Chunyan Bai, Yu-Wing Tai, and Chi-Keung Tang. Deep video generation, prediction and completion of human action sequences. In *Proceedings of the European conference on computer vision (ECCV)*, pages 366–382, 2018.

[2] Yujun Cai, Yiwei Wang, Yiheng Zhu, Tat-Jen Cham, Jianfei Cai, Junsong Yuan, Jun Liu, Chuanxia Zheng, Sijie Yan, Henghui Ding, 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*, pages 11645–11655, 2021.

[3] Zhe Cao, Hang Gao, Karttikeya Mangalam, Qi-Zhi Cai, Minh Vo, and Jitendra Malik. Long-term human motion prediction with scene context. In *European Conference on Computer Vision*, pages 387–404. Springer, 2020.

[4] Qiongjie Cui, Huaijiang Sun, Yue Kong, Xiaoqian Zhang, and Yanmeng Li. Efficient human motion prediction using temporal convolutional generative adversarial network. *Information Sciences*, 545:427–447, 2021.

[5] Erik B Dam, Martin Koch, and Martin Lillholm. *Quaternions, interpolation and animation*, volume 2. Citeseer, 1998.

[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 *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 11467–11476, 2021.

[7] Katerina Fragkiadaki, Sergey Levine, Panna Felsen, and Jitendra Malik. Recurrent network models for human dynamics. In *Proceedings of the IEEE international conference on computer vision*, pages 4346–4354, 2015.

[8] S. Ghorbani, C. Wloka, A. Etemad, M. A. Brubaker, and N. F. Troje. Probabilistic character motion synthesis using a hierarchical deep latent variable model. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '20, Goslar, DEU, 2020. Eurographics Association.

[9] Anand Gopalakrishnan, Ankur Arjun Mali, Daniel Kifer, C. Lee Giles, and Alexander Ororbia. A neural temporal model for human motion prediction. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12108–12117, 2019.

[10] Ikhsanul Habibie, Daniel Holden, Jonathan Schwarz, Joe Yearsley, and Taku Komura. A recurrent variational autoencoder for human motion synthesis. In *28th British Machine Vision Conference*, pages 1–12, 2017.

[11] Félix G. Harvey and Christopher Pal. Recurrent transition networks for character locomotion. In *SIGGRAPH Asia 2018 Technical Briefs*, SA '18, New York, NY, USA, 2018. Association for Computing Machinery.

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

[13] Alejandro Hernandez, Jurgen Gall, and Francesc Moreno-Noguer. Human motion prediction via spatio-temporal inpainting. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7134–7143, 2019.

[14] Daniel Holden, Taku Komura, and Jun Saito. Phase-functioned neural networks for character control. *ACM Transactions on Graphics (TOG)*, 36(4):1–13, 2017.

[15] Catalin Ionescu, Dragos Papava, Vlad Olaru, and Cristian Sminchisescu. Human3. 6m: Large scale datasets and predictive methods for 3d human sensing in natural environments. *IEEE transactions on pattern analysis and machine intelligence*, 36(7):1325–1339, 2013.

[16] Ashesh Jain, Amir R Zamir, Silvio Savarese, and Ashutosh Saxena. Structural-rnn: Deep learning on spatio-temporal graphs. In *Proceedings of the ieee conference on computer vision and pattern recognition*, pages 5308–5317, 2016.

[17] Kacper Kania, Marek Kowalski, and Tomasz Trzciński. Trajevae–controllable human motion generation from trajectories. *arXiv preprint arXiv:2104.00351*, 2021.

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

[19] Jihoon Kim, Taehyun Byun, Seungyoun Shin, Jungdam Won, and Sungjoon Choi. Conditional motion in-betweening. *Pattern Recognition*, 132:108894, dec 2022.

[20] Jogendra Nath Kundu, Maharshi Gor, and R. Venkatesh Babu. Bihmp-gan: Bidirectional 3d human motion prediction gan. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence and Thirty-First Innovative Applications of Artificial Intelligence Conference and Ninth AAAI Symposium on Educational Advances in Artificial Intelligence*, AAAI'19/IAAI'19/EAAI'19. AAAI Press, 2019.

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

[22] Xiao Lin and Mohamed R Amer. Human motion modeling using dvgans. *arXiv preprint arXiv:1804.10652*, 2018.

[23] 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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6437–6446, 2022.

[24] Naureen Mahmood, Nima Ghorbani, Nikolaus F Troje, Gerard Pons-Moll, and Michael J Black. Amass: Archive of motion capture as surface shapes. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 5442–5451, 2019.

[25] Wei Mao, Miaomiao Liu, Mathieu Salzmann, and Hongdong Li. Learning trajectory dependencies for human motion prediction. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9489–9497, 2019.

[26] Xudong Mao, Qing Li, Haoran Xie, Raymond Y.K. Lau, Zhen Wang, and Stephen Paul Smolley. Least squares generative adversarial networks. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[27] Julieta Martinez, Michael J Black, and Javier Romero. On human motion prediction using recurrent neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2891–2900, 2017.

[28] Jianyuan Min, Yen-Lin Chen, and Jinxiang Chai. Interactive generation of human animation with deformable motion models. *ACM Transactions on Graphics (TOG)*, 29(1):1–12, 2009.

[29] Boris N. Oreshkin, Antonios Valkanas, Félix G. Harvey, Louis-Simon Ménard, Florent Bocquelet, and Mark J. Coates. Motion inbetweening via deep Δ-interpolator. *CoRR*, abs/2201.06701, 2022.

[30] Dario Pavllo, David Grangier, and Michael Auli. Quaternet: A quaternion-based recurrent model for human motion. *ArXiv*, abs/1805.06485, 2018.

[31] Mathis Petrovich, Michael J Black, and Gül Varol. Action-conditioned 3d human motion synthesis with transformer vae. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10985–10995, 2021.

[32] Jia Qin, Youyi Zheng, and Kun Zhou. Motion in-betweening via two-stage transformers. *ACM Trans. Graph.*, 41(6), nov 2022.

[33] Charles Rose, Brian Guenter, Bobby Bodenheimer, and Michael F Cohen. Efficient generation of motion transitions using spacetime constraints. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 147–154, 1996.

[34] Seyed Sina Mirrazavi Salehian, Nadia Figueroa, and Aude Billard. A unified framework for coordinated multi-arm motion planning. *The International Journal of Robotics Research*, 37(10):1205–1232, 2018.

[35] Sebastian Starke, He Zhang, Taku Komura, and Jun Saito. Neural state machine for character-scene interactions. *ACM Trans. Graph.*, 38(6):209–1, 2019.

[36] Sebastian Starke, Yiwei Zhao, Taku Komura, and Kazi Zaman. Local motion phases for learning multi-contact character movements. *ACM Transactions on Graphics (TOG)*, 39(4):54–1, 2020.

[37] Sebastian Starke, Yiwei Zhao, Fabio Zinno, and Taku Komura. Neural animation layering for synthesizing martial arts movements. *ACM Transactions on Graphics (TOG)*, 40(4):1–16, 2021.

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

[39] Jacob Walker, Kenneth Marino, Abhinav Gupta, and Martial Hebert. The pose knows: Video forecasting by generating pose futures. In *Proceedings of the IEEE international conference on computer vision*, pages 3332–3341, 2017.

[40] Hongsong Wang, Jian Dong, Bin Cheng, and Jiashi Feng. Pvred: A position-velocity recurrent encoder-decoder for human motion prediction. *IEEE Transactions on Image Processing*, 30:6096–6106, 2021.

[41] Jack M Wang, David J Fleet, and Aaron Hertzmann. Gaussian process dynamical models for human motion. *IEEE transactions on pattern analysis and machine intelligence*, 30(2):283–298, 2007.

[42] Andrew Witkin and Michael Kass. Spacetime constraints. *ACM Siggraph Computer Graphics*, 22(4):159–168, 1988.

[43] Jingwei Xu, Huazhe Xu, Bingbing Ni, Xiaokang Yang, Xiaolong Wang, and Trevor Darrell. Hierarchical style-based networks for

motion synthesis. In *European conference on computer vision*, pages 178–194. Springer, 2020.
[44] Yi Tian Xu, Yaqiao Li, and David Meger. Human motion prediction via pattern completion in latent representation space. In *2019 16th conference on computer and robot vision (CRV)*, pages 57–64. IEEE, 2019.
[45] Dongseok Yang, Doyeon Kim, and Sung-Hee Lee. LoBSTr: Real-time lower-body pose prediction from sparse upper-body tracking signals. *Computer Graphics Forum*, 40(2):265–275, may 2021.
[46] Moonwon Yu, Byungjun Kwon, Jongmin Kim, Shinjin Kang, and Hanyoung Jang. Fast terrain-adaptive motion generation using deep neural networks. In *SIGGRAPH Asia 2019 Technical Briefs*, SA '19, page 57–60, New York, NY, USA, 2019. Association for Computing Machinery.
[47] He Zhang, Sebastian Starke, Taku Komura, and Jun Saito. Mode-adaptive neural networks for quadruped motion control. *ACM Trans. Graph.*, 37(4), jul 2018.
[48] Xinyi Zhang and Michiel van de Panne. Data-driven autocompletion for keyframe animation. In *Proceedings of the 11th ACM SIGGRAPH Conference on Motion, Interaction and Games*, pages 1–11, 2018.
[49] Yan Zhang, Michael J Black, and Siyu Tang. We are more than our joints: Predicting how 3d bodies move. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3372–3382, 2021.
[50] Chongyang Zhong, Lei Hu, Zihao Zhang, Yongjing Ye, and Shihong Xia. Spatio-temporal gating-adjacency gcn for human motion prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6447–6456, 2022.
[51] Chi Zhou, Zhangjiong Lai, Suzhen Wang, Lincheng Li, Xiaohan Sun, and Yu Ding. Learning a deep motion interpolation network for human skeleton animations. *Computer Animation and Virtual Worlds*, 32(3-4):e2003, 2021.

Tianxiang Ren received his master's degree from School of Informatics, Xiamen University. He is a algorithm research engineer at SenseTime, China. His research interests include animation synthesis, human motion control and 3D vision.

Jubo Yu received his master's degree from School of Informatics, Xiamen University. His main research direction is action generation.

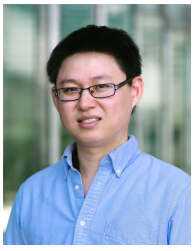
Shihui Guo is an associate professor at School of Informatics, Xiamen University. He received his Ph.D. in computer animation from National Centre for Computer Animation, Bournemouth University, UK and B.s. in Electrical Engineering from Peking University, China. His research interests include virtual reality, computer graphics and vision, human–computer interaction.

Ying Ma is a master student from Communication University of China. Her field of study revolves around Intelligent Science and Technology, with a specific focus on computer vision and virtual reality technology.

Yutao Ouyang is studying in the Department of Artificial Intelligence, School of Informatics, Xiamen University. His research interests include reinforcement learning, robotics and 3D vision.

Zijiao Zeng is currently the leader of the Animation Technology Research Team at Tencent Games. His research focuses on 3D vision, computer graphics, and motion synthesis.

Yazhan Zhang is a senior research engineer at Tencent Ltd, China. His research interests include animation synthesis, automation toolchain, robotic tactile sensing and manipulation.



Yipeng Qin received his BSc degree in Electrical Engineering at Shanghai Jiao Tong University, China and went on to complete a PhD at the National Centre for Computer Animation (NCCA), Bournemouth University, UK. He was a postdoctoral research fellow at the Visual Computing Center (VCC), King Abdullah University of Science and Technology (KAUST), Saudi Arabia and is now working as a Lecturer at the School of Computer Science and Informatics, Cardiff University, UK. His current research interests include deep learning, computer vision, computer graphics, and human-computer interaction (HCI), with a focus on generative modeling and visual content creation.