Pacific Graphics 2021
E. Eisemann, K. Singh, and F.-L Zhang
(Guest Editors)

Volume 40 (2021), Number 7

Diverse Dance Synthesis via Keyframes with Transformer Controllers

Junjun Pan[1,2] and Siyuan Wang[1] and Junxuan Bai[1,2] and Ju Dai[2†]

[1]Beihang University, State Key Laboratory of Virtual Reality Technology and Systems, Beijing, China
[2]Peng Cheng Laboratory, Shenzhen, China

Abstract

Existing keyframe-based motion synthesis mainly focuses on the generation of cyclic actions or short-term motion, such as walking, running, and transitions between close postures. However, these methods will significantly degrade the naturalness and diversity of the synthesized motion when dealing with complex and impromptu movements, e.g.*, dance performance and martial arts. In addition, current research lacks fine-grained control over the generated motion, which is essential for intelligent human-computer interaction and animation creation. In this paper, we propose a novel keyframe-based motion generation network based on multiple constraints, which can achieve diverse dance synthesis via learned knowledge. Specifically, the algorithm is mainly formulated based on the recurrent neural network (RNN) and the Transformer architecture. The backbone of our network is a hierarchical RNN module composed of two long short-term memory (LSTM) units, in which the first LSTM is utilized to embed the posture information of the historical frames into a latent space, and the second one is employed to predict the human posture for the next frame. Moreover, our framework contains two Transformer-based controllers, which are used to model the constraints of the root trajectory and the velocity factor respectively, so as to better utilize the temporal context of the frames and achieve fine-grained motion control. We verify the proposed approach on a dance dataset containing a wide range of contemporary dance. The results of three quantitative analyses validate the superiority of our algorithm. The video and qualitative experimental results demonstrate that the complex motion sequences generated by our algorithm can achieve diverse and smooth motion transitions between keyframes, even for long-term synthesis.*

CCS Concepts

• ***Computing methodologies*** → ***Motion processing; Motion capture;***

1. Introduction

Character animation is one of the essential research topics in computer graphics. Animators usually use motion capture systems or keyframe-based techniques to obtain high-quality animation data. However, massive editing and modification still need to be executed for the final animation production, which is quite tedious and time-consuming. With the development of deep learning techniques, scholars have made significant efforts to simplify the motion synthesis process. A prevailing trend is to utilize deep neural networks to generate natural and diverse human movements [H-SK16, ZvdP18]. More recently, there has been some work focusing on how to achieve a smoother transition [HYNP20]. These studies mainly focus on predicting and controlling cyclic actions such as walking and running, and they often achieve better results in those simple movements. However, when applied to complex and impromptu activities such as dance performance or martial arts, the generated animations are far from satisfactory.

Compared with walking, running, and other cyclic locomotion, learning to synthesize diverse and artistically-elegant dance movements from keyframes is more challenging. Firstly, dance performance is highly irregular with complex kinetics, *e.g.*, body rotation in contemporary dance can have various speeds, different strengths, and larger amplitudes. Secondly, dance movements are inherently diversified, *e.g.*, the current motion can be followed by a wide range of possible movements. Thirdly, in long-term motion sequence synthesis, the entire dance movements may be composed of different dance action units or various combinations of them. Lastly, it is more difficult to obtain a dance performance than walking or running. A well-choreographed dance animation requires collaborative efforts between animators, dancers, and choreographers, which is an expensive and tedious process. However, little research investigates efficient keyframe-based motion synthesis for an impromptu dance performance at present. The topic is extremely valuable, which can significantly reduce the demand for professional motion capture systems, the dependence on professional choreographers, and the workload of animation designers when creating new dance animation.

† The corresponding author: daij@pcl.ac.cn.

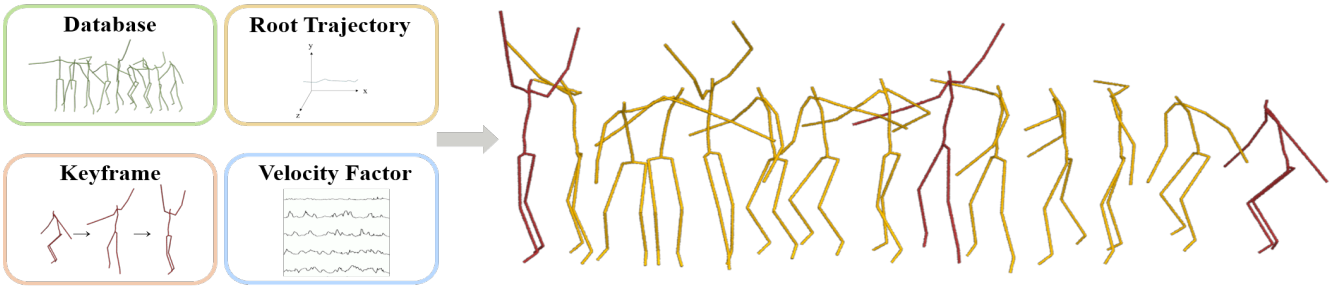


Figure 1: *Dance generation from keyframes conditioned on the root trajectory and the velocity factor constraints. The postures marked in red are the keyframes, and the postures marked in yellow are the synthesized frames.*

In this paper, we propose a novel keyframe-based motion generation network based on multiple constraints, which can achieve diverse dance synthesis via learned knowledge. The constraints include the given keyframes, the root trajectory, and the velocity factor. Similar to [ECC*20], the velocity factor is designed to constrain the motion synthesis. Specifically, the approach is mainly formulated based on the recurrent neural network (RNN) and the Transformer architecture. The core of our network is a hierarchical RNN module composed of two long short-term memory (LSTM) units, in which the first LSTM is utilized to embed the posture information of the historical frames into a latent space, and the second one is employed to predict the human posture for the next frame. We also design two Transformer-based controllers to model the constraints of the root trajectory and the velocity factor respectively. The self-attention layers in the Transformer encourage a model considering the broad context in a given sequence by learning the relationships between different elements [VSP*17]. Therefore, the proposed Transformer-based controllers enable our network to utilize the temporal context of the frames to achieve fine-grained motion control. We verify the proposed algorithm on a dance dataset containing a variety of contemporary dance. The video and quantitative analyses prove the superiority of our algorithm. Moreover, the demo and qualitative experimental results demonstrate that the complex motion sequence generated by our algorithm is capable of producing diverse and smooth motion transitions between keyframes, even for long-term synthesis.

In summary, **our main contributions** are listed as follows:

- We propose a novel neural network based on LSTM and Transformer for complex motion generation via keyframes. The model is elaborately controlled under the root trajectory and the velocity factor constraints, and can generate complex dance movements satisfying the control conditions.
- We design the velocity factor constraint for fine-grained dance motion synthesis. By specifying the velocities of different body parts, our model is able to enhance the diversity and smoothness in long-term motion generation.
- Compared with the state-of-the-art motion transition methods, the data synthesized by our technique on the dance dataset obtain better accuracy in terms of various evaluation criteria, and the quality of character animation is also higher.

2. Related work

In this section, we briefly review the literatures closely related to our work, including human motion modeling, dance motion synthesis, and motion transition generation.

2.1. Human motion modeling based on deep learning

Recently, deep learning has gained remarkable success in the field of both CV and CG. To synthesis realistic and natural human motion, there has been a surge in modeling human motion with neural networks. For example, Holden *et al.* [HSK16] propose a deep convolutional network to perform human motion synthesis and resolve the ambiguity via incorporating foot contact information. However, the proposed framework concentrates on simple character movements regarding walking, running, and punching. Inspired by the image inpainting techniques, Hernandez *et al.* [RGM19] reformulate motion prediction as an inpainting task to complete the masked joints in spatiotemporal volumes. It is well-known that recurrent neural networks (RNNs) have inherent advantages in modeling sequential data. Hence recent work employs RNNs to model human motion. Martinez *et al.* [MBR17] make several modifications to the standard RNN models and develop a sequence-to-sequence architecture with residual connections for short-term human motion prediction. Lee *et al.* [LLL18] present a multi-layer RNN conditioned to handle spatiotemporal constraints and structural variabilities for interactive character animation. Wang *et al.* [WHSZ21] formulate a new spatiotemporal RNN framework to investigate the motion manifold. The proposed network avoids the generation of average posture and eliminates the need for a separate disambiguous network. Also, Wang *et al.* [WCX21] construct an RNN-based generator for human motion synthesis and utilize a refiner network with adversarial training loss to refine motion sequences. In our work, the RNN is utilized for sequence modeling and long-term motion synthesis. The constrained conditions in our method is modeled using two additional Transformers, which enables the network to synthesize diverse dance motion sequence.

2.2. Dance motion synthesis

Dance motion synthesis can be viewed as a typical conditional motion generation task. Due to the high correlation between dance and music, extensive works have been dedicated to the music-oriented dance generation. In the early research, dance-to-music is regarded

as the problem of template matching, which attempts to generate dance movements according to musical similarity [SNI06, LLP13]. By analyzing the rhythmic patterns of motions, Kim *et al.* [KPS03] facilitate the rhythmic motion generation synchronized with an input sound signal. However, the above methods are limited by their capacity of the provided dataset. Recently, neural networks dominate dance motion synthesis. Tang *et al.* [TJM18] design an LSTM-autoencoder model to extract the mappings between music and motion, which can largely enhance choreography in accordance with the music. Lee *et al.* [LYL*19] and Ye *et al.* [YWJ*20] attempt to synthesize dance from music through a two-stage procedure, which firstly learn basic dance units and then organizes the basic units into dance sequence. Nevertheless, two-stage generation methods lack enough flexibility and scalability. To alleviate error accumulation of autoregressive model in long-term motion generation, Huang *et al.* [HHW*21] formulate the music-conditioned dance generation as a sequence-to-sequence learning problem and utilize the curriculum learning strategy to enhance the training process. Considering that the motion manifolds of classical convolutional and recursive neural models are non-Euclidean geometry, Ferreira *et al.* [FCG*21] design a novel method based on graph convolutional networks to synthesize human motion from music. The goal of our research is different from the above work. We attempt to synthesize dance motion sequences conditioned on temporal sparse keyframes with user-specified root trajectory and velocity factor. It is a rarely explored but rather valuable topic for its significance in character animation and entertainment.

2.3. Motion transition generation

In computer animation, there have been intensive investigations on motion transition generation. We limit the task as synthesizing intermediate movements between user-specified keyframes. It is quite challenging as significant motion gaps must be filled under sparse temporal constraints. Pioneering approaches are mainly based on retrieval paradigms, which search the matched motion clips from a database and blend them, such as motion graphs [KGP02, ZS09]. After that, methods based on probabilistic models have been widely adopted for human motion. Chai *et al.* [CH07] and Min *et al.* [MCC09] formulate motion synthesis as maximum a posterior (MAP) problem. Wang *et al.* [WFH08] apply Gaussian process dynamical models (GPDMs) for learning models of human pose and motion. However, the above methods are designed for designated actions, making the algorithms look like pre-arranged scripts and rules that will fail in confronting complex human movements. Because of the impressive scalability and expressiveness of deep neural networks, recent studies tend to use deep learning for motion transition prediction. Gaisbauer *et al.* [GLSR19] present a fully-connected feed-forward neural network for generating feasible postures from given input postures. Zhang *et al.* [ZvdP18] formulate an autoregressive recurrent neural network (ARNN) that is conditioned on the target keyframes for motion-aware animation with fixed interpolated frames. Harvey *et al.* [HP18] propose the recurrent transition networks (RAT) based on LSTM to synthesize missing data between keyframes. However, the method can still only generate motion transition sequences with definite lengths and is limited to periodic routine actions. Later, Harvey *et al.* [HYNP20] improve the RAT method [HP18] by adding the time-to-arrival embedding to the network, which allows the method generates variable transition lengths. They verify the method on a periodic dance dataset and obtain good animation results. In summary, existing methods mainly focus on the motion transition of periodic actions with relatively short-term synthesis, while our work concentrates on the long-term and impromptu contemporary dance generation. For diverse motion synthesis, besides the conventional root trajectory constraint, we introduce the velocity factor for the first time in motion modeling, which preserves the naturalness and diversity of the generated dance movements.

3. Method

Throughout the paper, we denote *keyframes* as the user-specified temporal sparse representative frames in a motion sequence. Our generative model can be regarded as a time series modeling and synthesis problem. The overall framework of dance synthesis is illustrated in Figure 1. With the current frame and the constrained conditions, the model can automatically predict the pose and the associated information of the next frame. Given the keyframes and the length of the sequence to be predicted, the system can automatically synthesize the intermediate motion sequence between the keyframes. Moreover, users can specify the trajectory of the root joint and the moving speeds of different body parts, which are employed as control signals to guide the network to generate the desired animation.

3.1. Variable definitions

The dance dataset we use comes from [dat, AZS*17]. We remove the finger joints, and the resulted human skeleton contains 23 joints. We preprocess the dataset by rotating the postures at different timesteps toward the positive direction of z-axis and extract the related information based on the rotated postures. The definition of the symbols used in this paper is listed in Table 1.

We use $\mathbf{X}_t = \{\mathbf{p}_t, \mathbf{o}^r_t, \mathbf{c}_t, \mathbf{v}_t\}$ to represent the ground truth information at time t and $\tilde{\mathbf{X}}_t = \{\tilde{\mathbf{p}}_t, \tilde{\mathbf{o}}^r_t, \tilde{\mathbf{c}}_t, \tilde{\mathbf{v}}_t\}$ to denote the prediction results of the network at time t. The given inputs and conditions can be expressed as $\{\mathbf{p}_{k_1}, \mathbf{o}^r_{k_1}, \mathbf{p}_{k_2}, \mathbf{o}^r_{k_2}, \hat{\mathbf{p}}^r, \hat{\mathbf{f}}\}$. The output of the model contains the position sequence $\{\tilde{\mathbf{p}}_{k_1+1}, \cdots, \tilde{\mathbf{p}}_{k_2}\}$ of all joints, the rotation sequence $\{\tilde{\mathbf{o}}^r_{k_1+1}, \cdots, \tilde{\mathbf{o}}^r_{k_2}\}$ of the root joint, the contact label sequence $\{\tilde{\mathbf{c}}_{k_1+1}, \cdots, \tilde{\mathbf{c}}_{k_2}\}$ of the toe joints and heel joints, as well as the velocity sequence $\{\tilde{\mathbf{v}}_{k_1+1}, \cdots, \tilde{\mathbf{v}}_{k_2}\}$ of all joints.

3.2. The proposed motion synthesis network

The proposed network aims to receive historical information and control signals, then produce smooth and diverse dance sequences. As shown in Figure 2, we demonstrate the generation process in one timestep. Given two keyframes k_1 and k_2, the problem is how to generate a realistic motion sequence between the two keyframes with a natural transition. In essence, this is a time-series-prediction problem. Inspired by [HYNP20], we utilize the LSTM unit to predict the motion sequence.

Overview of the network. The whole framework consists of three encoders, three controllers, and two decoders. The three encoders are comprised of a state encoder, a velocity encoder, and a position

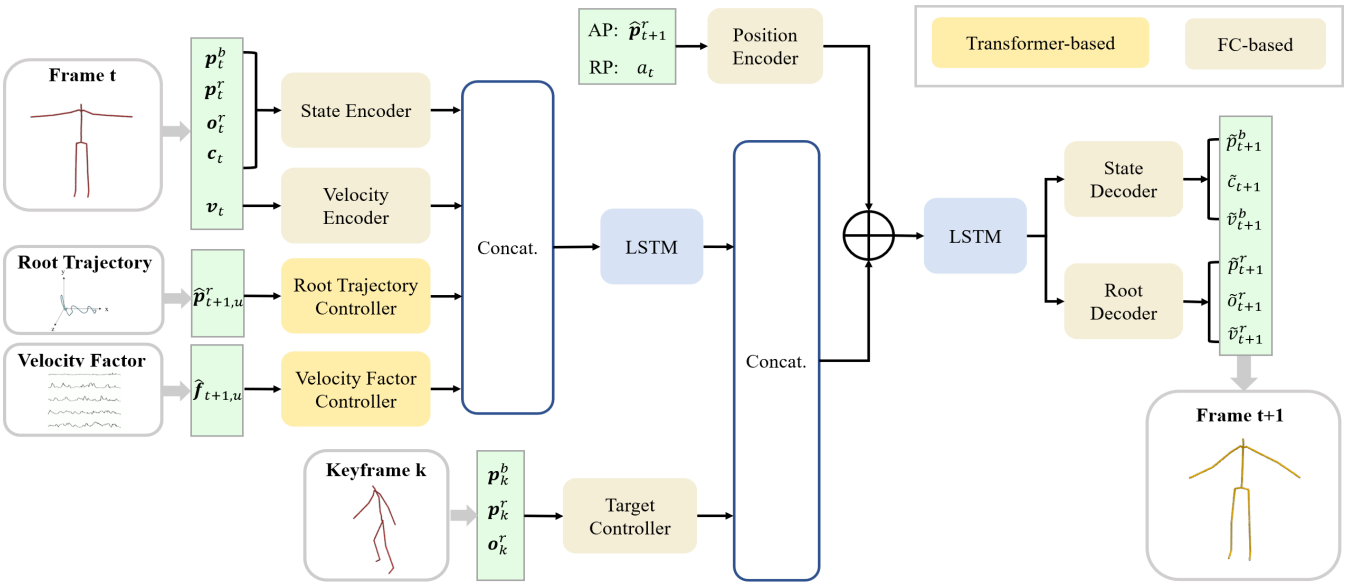


Figure 2: *The framework of our motion synthesis network. It shows all the computations for a single timestep.*

Table 1: *A table with variable definitions.*

Symbol	Definition
t	The time mark of the current frame.
J	The total number of joints.
k_1, k_2	The indexes of the given keyframes, where $k_1 < k_2$.
N	Number of frames between keyframes, $N = k_2 - k_1$.
$\mathbf{o}_t^r$	The rotation angle to rotate a posture toward the positive direction of the z axis.
$\mathbf{p}_t^r$	The global position of the root joint.
$\mathbf{p}_t^b$	The root-relative position of other joints except for root joint. The dimension is $(J-1)\times 3$.
$\mathbf{p}_t$	The set of the global position of root joint and the root-relative positions of other joints, that is $\mathbf{p}_t = \{\mathbf{p}_t^r, \mathbf{p}_t^b\}$.
$\mathbf{v}_t^r$	The velocity of the root joint.
$\mathbf{v}_t^b$	The velocities of other joints except for root joint. The dimension is $(J-1)\times 3$.
$\mathbf{v}_t$	The set of velocities of all the joints with $\mathbf{v}_t = \{\mathbf{v}_t^r, \mathbf{v}_t^b\}$.
$\mathbf{c}_t$	The foot contact labels of two toe joints and two heel joints with dimension of 4.
$\mathbf{f}_t$	The velocity factor vector with dimension of 5.
$\hat{\mathbf{p}}^r$	The given root trajectory sequence with dimension of $N\times 3$.
$\hat{\mathbf{f}}$	The given velocity factor sequence with dimension of $N\times 5$.
$\hat{\mathbf{p}}_{t,u}^r$	A subsequence of the given root trajectory sequence. The subsequence is centered at frame t, the window size is u, and the dimension is $u\times 3$.
$\hat{\mathbf{f}}_{t,u}$	A subsequence of the given velocity factor sequence. The subsequence is centered at frame t, the window size is u, and the dimension is $u\times 5$.
M	The number of divided body parts.

encoder. Specifically, the state encoder receives the posture information, *i.e.*, the root-relative positions $\mathbf{p}_t^b$, the root's global position $\mathbf{p}_t^r$, the root's rotation angle $\mathbf{o}_t^r$, and the foot contact information $\mathbf{c}_t$. The position encoder takes the next frame's root position as input to make the network aware of the temporal embedding. The velocity encoder projects the dynamics of the movement to our network. The three controllers are composed of a root trajectory controller, a velocity factor controller, and a target controller. The root trajectory controller takes the positions of the root's trajectory as input to ensure the generated motion moves towards the specified trajectory. The velocity factor is proposed to achieve fine-grained motion control by specifying the moving speeds of different body parts. The target controller receives the posture information of the keyframe. The goal is to make the network perceive the distance between the predicted frame and the target keyframe. The decoders contain a root decoder and a state decoder. The root decoder is in charge of decoding the root position, rotation, and velocity information, while the state decoder predicts the relative positions and velocities of other joints except for the root, as well as the foot contact states at time $t+1$. In summary, the three encoders, two decoders, and the target controller are constructed based on the fully connected layer (FC-based) with different depths, while the root trajectory controller and the velocity factor controller are formulated based on the Transformer structure (Transformer-based).

The root trajectory controller. To make the generated motion consistent with the given trajectory, it is crucial to let the network capture the temporal context information from the past to the future. Since Transformer [VSP*17] can well model long-range dependency by leveraging the query-key correlation to different tokens, it has shown outstanding results in natural language processing [MZZ*19], neural machine translation [BDK20], and various vision tasks [ZZT*20, CMS*20]. Therefore, we construct the root trajectory controller based on the Transformer mechanism.

We illustrate the detailed structure of the root trajectory con-

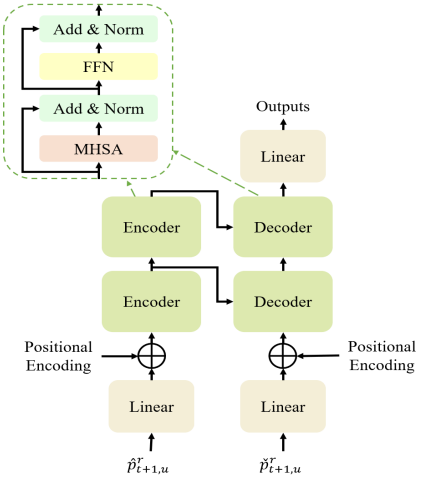


Figure 3: *The structure diagram of the Transformer-based root trajectory controller.*

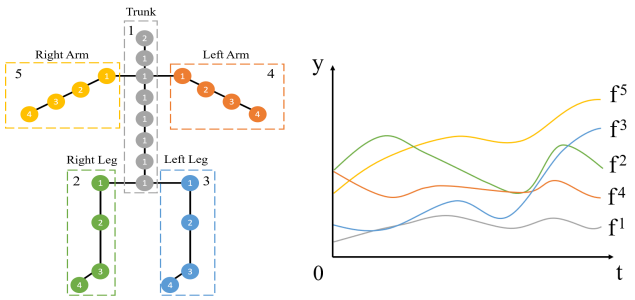
troller in Figure 3, where $\hat{\mathbf{p}}^r_{t+1,u}$ refers to the target root trajectory segment with frame $t+1$ as the center and the window size of u. $\check{\mathbf{p}}^r_{t+1,u}$ is the modification of $\hat{\mathbf{p}}^r_{t+1,u}$, that is, the data before the frame $t+1$ is replaced by the predicted root trajectory. Considering that the root trajectory predicted by the network cannot be completely consistent with as the target trajectory, and the current predicted position is not only required to smoothly connect with the past predicted results, but also move along the future target trajectory, it is not enough to only encode the given target trajectory. Therefore, we introduce the $\check{\mathbf{p}}^r_{t+1,u}$ and define it as the mixture root trajectory segment. The encoders and decoders in the Transformer are the standard combinations of the multi-head self-attention (MHSA) and the feed forward networks (FFN). The MHSA comprises multiple self-attention blocks and explicitly models the interactions between all entities of a sequence. The FFN is used to perform information transformation. The residual add and layer norm are implemented after both the MHSA and the FFN.

For the constraint coding process, the controller first accepts $\hat{\mathbf{p}}^r_{t+1,u}$ as input, projects it into the embedding space, and then extracts the trajectory representation with two encoders. We also project $\check{\mathbf{p}}^r_{t+1,u}$ into the feature space and then decode it with two decoders. Skip connections are used between the encoders and the decoders to increase the flexibility of information flow. As pointed out in [VSP*17], the Transformer has a permutation invariance. We follow its procedure, incorporating the positional encoding into $\hat{\mathbf{p}}^r_{t+1,u}$ and $\check{\mathbf{p}}^r_{t+1,u}$ to make the controller aware of the trajectory segments' positions. The technical details can be found in the literature above. Finally, the decoded result is transformed by a linear layer to acquire the representation of the root trajectory constraint.

The velocity factor controller. To achieve fine-grained control, the velocity factor controller is introduced. As the speeds of different body parts can vary significantly when dancing, we divide the human body into five parts: the trunk, the left and right arms, the left and right legs. We get five velocity factors by weighted average over the moving speeds of the joints related to the body parts. Figure 4 (a) displays the structure diagram of body division, where different colors represent different body parts. For different joints, the weights are set according to their distances from the end effector. The numbers in the circle of Figure 4 (a) are the weights used in this paper. In general, the end effectors are more influential in describing the dynamics of a movement, so joints closer to them have larger weights. The velocity factor of body part i, $i \in \{1,2,3,4,5\}$ at frame t can be calculated as:

$$f_t^i = \frac{\sum_j^{J_i}(w^{ji} \times \|\mathbf{v}_t^{ji}\|_2)}{\sum_j^{J_i} w^{ji}}, \tag{1}$$

where J_i is the joint set of part i, w^{ji} is the weight of joint j in part i, and $\mathbf{v}_t^{ji}$ is the velocity of joint j in part i at time t. The obtained velocity factor sequences of different body parts for a given motion sequence are illustrated in figure 4 (b).



(a) Body part division (b) Velocity factor sequences

Figure 4: *The structure diagram representation of the velocity factor. (a) Different colors represent different body parts. The numbers in the circles represent the joint weights, while the numbers in the dotted boxes denote the body-part index. (b) The values of the corresponding velocity factors are displayed in this figure.*

Considering human motion at consecutive moments should be continuous without sudden changes, it is important for the velocity factor controller to also perceive the long-range context information, so as to learn the smooth temporal context representation. Therefore, we construct the velocity factor controller with the same structure as the trajectory controller. The differences are that the inputs $\hat{\mathbf{p}}^r_{t+1,u}$ and $\check{\mathbf{p}}^r_{t+1,u}$ are replaced by the $\hat{\mathbf{f}}_{t+1,u}$ and $\check{\mathbf{f}}_{t+1,u}$. Here, the $\hat{\mathbf{f}}_{t+1,u}$ is a sequence segment of the given velocity factor with the time $t+1$ as the center and the window size u, and the $\check{\mathbf{f}}_{t+1,u}$ replaces the data of $\hat{\mathbf{f}}_{t+1,u}$ before time $t+1$ with the network predicted results. Similar to $\check{\mathbf{p}}^r_{t+1,u}$, $\check{\mathbf{f}}_{t+1,u}$ is defined as the mixture velocity factor segment.

The LSTM backbone. If we directly send all the coding formation of the encoders and the controllers to the LSTM network, the information received by the network will be too much and too complex. However, one LSTM unit may not be able to process so much data at one time. Therefore, we employ two LSTM units hierarchically to receive and process the encoded information. The first LSTM receives the outputs of the state and velocity encoders, as well as the root trajectory and velocity factor controllers. After the information is processed by the first LSTM, we concatenate its output with the keyframe embedding of the target controller and send the concatenated results to the second LSTM for further processing. The sequential processing inputs and control conditions make the

network achieve better results. We will validate it in the ablation study. In experiments, we employ a scheduled sampling mechanism to select the input of LSTM in each timestep. Specifically, we first define a sampling probability rate β. When predicting actions in different timesteps, we select the ground truth as the input with the probability of β. At the beginning, β is set as 1. With the progress of the training, β decays in an exponential decay manner. The learning strategy can make the convergence of the network more stable and make the motion synthesis and prediction more smooth.

When the pose information of the target keyframe is directly sent to the network, the network cannot know the temporal distance between the frame to be predicted and the target keyframe. Referring to the insight in Transformer, we encode the position information of the frame to be predicted and add it with the input of the second LSTM unit. We use one fully connected layer to embed the position information. Specifically, as pointed out in [ZJJ*20], the 3D joint coordinate is a kind of natural position coding. We define the root's position as the absolute position (AP) representation. The root joint of frame $t+1$ extracted from the root trajectory constraint is used as input of the position encoder. Since the trajectory sequence of the root joint may have the same position at different times, we design an additional relative position (RP) representation a_t, which is calculated as follows:

$$a_t = \frac{t+1-k_1}{k_2-k_1}. \tag{2}$$

At last, the state decoder and the root decoder are used to predict the posture information of the next movement. However, when all the information is decoded by the state decoder, the predicted position sequence of the root joint is discontinuous with relatively large fluctuation. Inspired by the 3D human body pose estimation algorithms [ZFS*20,LL20], we separate the information of the root joint and use the root decoder to predict its state information. This practice can make the generated trajectory sequence smoother.

3.3. Loss functions

According to the network structure and goal, we define several loss functions, including the reconstruction loss $\mathcal{L}_{rec}$, posture consistency loss $\mathcal{L}_{con}$, root trajectory smooth loss $\mathcal{L}_{root}$, keyframe consistency loss $\mathcal{L}_{key}$, and velocity factor consistency loss $\mathcal{L}_{vfac}$. The complete loss function is defined as follows:

$$\mathcal{L} = w_{rec}\mathcal{L}_{rec} + w_{con}\mathcal{L}_{con} + w_{root}\mathcal{L}_{root} + w_{key}\mathcal{L}_{key} + w_{vfac}\mathcal{L}_{vfac}, \tag{3}$$

where w_{rec}, w_{con}, w_{root}, w_{key}, w_{vfac} are the corresponding loss weights. We set the loss weights as 0.3, 0.2, 0.15, 0.2, and 0.15 respectively in training phase. These parameters are obtained through a number of experiments. We give the details of each loss below.

Reconstruction loss. The mean square error (MSE) loss is taken to construct the reconstruction loss, which can force the network to generate motion sequences satisfying the designed control constraints. The reconstruction loss $\mathcal{L}_{rec}$ can be expressed as:

$$\mathcal{L}_{rec} = \frac{1}{N}\sum_{t=k_1+1}^{k_2} \|\tilde{\mathbf{X}}_t - \mathbf{X}_t\|^2, \tag{4}$$

where N is the sequence length. The terms of $\mathcal{L}_{rec}$ include the reconstruction of joint positions, root joint rotation angles, foot contact labels, and joint velocities at each moment.

Posture consistency loss. When the network is trained only with individual joints, the correlations between the connected joints are neglected to a certain extent. Hence, we introduce the bone length consistency loss $\mathcal{L}_{bone}$ to force the network to generate posture consistent with the ground truth bone length. In the meantime, the foot contact labels and the joint velocities can be inferred from the predicted posture information. Thus, we also introduce the foot contact consistency loss $\mathcal{L}_{contact}$ and the joint velocity consistency loss $\mathcal{L}_{velocity}$ to avoid the contradiction of prediction information. Therefore, the consistency loss $\mathcal{L}_{con}$ consists of $\mathcal{L}_{bone}$, $\mathcal{L}_{contact}$ and $\mathcal{L}_{velocity}$, and they are expressed as follows:

$$\mathcal{L}_{con} = \mathcal{L}_{bone} + \mathcal{L}_{contact} + \mathcal{L}_{velocity}, \tag{5}$$

$$\mathcal{L}_{bone} = \frac{1}{N}\sum_{t=k_1+1}^{k_2}\Big(\sum_{(i,j)\in\mathcal{B}} \big\|\|\tilde{\mathbf{p}}_t^i - \tilde{\mathbf{p}}_t^j\|_2 - l_{ij}\big\|^2\Big), \tag{6}$$

$$\mathcal{L}_{contact} = \frac{1}{N}\sum_{t=k_1+1}^{k_2}\Big(\sum_{i}^{\mathcal{F}} \tilde{c}_t^i \|\tilde{\mathbf{v}}_t^i\|_2\Big), \tag{7}$$

$$\mathcal{L}_{velocity} = \frac{1}{N}\sum_{t=k_1+1}^{k_2}\Big(\sum_{i}^{\mathcal{J}} \|\tilde{\mathbf{v}}_t^i - (\tilde{\mathbf{p}}_t^i - \tilde{\mathbf{p}}_{t-1}^i)\|^2\Big), \tag{8}$$

where $\mathcal{B}$ in $\mathcal{L}_{bone}$ is the index set consisted of all natural connected joint pairs in the human skeleton. l_{ij} is the original length of the bone segment formed by joint i and j, the length of which can also be obtained by calculating the distance between $\tilde{\mathbf{p}}_t^i$ and $\tilde{\mathbf{p}}_t^j$. We penalize the length inconsistency between the ground truth and the inferred to force the correctness of the predicted posture information. In $\mathcal{L}_{contact}$, $\mathcal{F}$ is the index set of the foot contact joints, $\tilde{c}_t^i$ is the predicted contact label for foot joint i at frame t, $\tilde{c}_t^i = 1$ if there is foot contact and 0 otherwise. We impose a penalty on the product of the L2-norm of $\tilde{\mathbf{v}}_t^i$ and the corresponding contact label $\tilde{c}_t^i$ to force the consistency of the predicted foot contact label and the predicted velocity [WHSZ21]. In $\mathcal{L}_{velocity}$, $\mathcal{J}$ is the index set of all joints. The velocities of different joints at frame t can be inferred by subtracting skeleton positions in the previous frame from the current frame t. The information consistency can be effectively guaranteed by punishing the differences between the inferred velocities and the predicted velocities of our network.

Root trajectory smooth loss. We refer to the long horizon loss function in [WHSZ21] and extract the loss term of root joint to form our root trajectory smooth loss $\mathcal{L}_{root}$:

$$\mathcal{L}_{root} = \frac{1}{N}\left(\sum_{t=k_1+1}^{k_2} \|\tilde{\mathbf{p}}_t^r - \tilde{\mathbf{p}}_{t-1}^r\|^2 + \sum_{t=k_1+1}^{k_2} \|\tilde{\mathbf{o}}_t^r - \tilde{\mathbf{o}}_{t-1}^r\|^2\right), \tag{9}$$

We minimize the differences between the root's spatial positions and rotation angles at frame t and t -1 to enforce temporal consistency. When we impose a similar constraint on other joints, the network tends to overfit, and the final result converges to an average posture. Therefore, we only use the joint trajectory smooth loss to constrain the natural smooth transition for the root joint.

Keyframe consistency loss. One of the main tasks of the network is to build natural connections of the generated dance with the given keyframes, which means that we need to ensure the continuity of the predicted movements near keyframes and at keyframes. Hence, the keyframe consistency loss $\mathcal{L}_{key}$ is introduced to achieve

the goal:

$$\mathcal{L}_{key}=\begin{cases}\frac{1}{2m}\left(\sum_{t=k_1+1}^{k_1+m}\|\tilde{\mathbf{p}}_t-\mathbf{p}_{k_1}\|^2+\sum_{t=k_2-m+1}^{k_2}\|\tilde{\mathbf{p}}_t-\mathbf{p}_{k_2}\|^2\right), & N>2m,\\ \frac{1}{N}\sum_{t=k_1+1}^{k_2}\left(\frac{t-k_1}{N}\|\tilde{\mathbf{p}}_t-\mathbf{p}_{k_1}\|^2+\left(1-\frac{t-k_1}{N}\right)\|\tilde{\mathbf{p}}_t-\mathbf{p}_{k_2}\|^2\right), & N\leq 2m,\end{cases} \tag{10}$$

where m is the number of frames affected by the keyframes. When $N>2m$, we impose constraints on the m frames near the keyframe k_1 or keyframe k_2; When $N\leq 2m$, the in-between N frames are constrained to be affected by the mixture results of the keyframes k_1 and k_2. We use $\frac{t-k_1}{N}$ as the impact factor to determine the influence weights of two keyframes on the predicted pose at time t. However, when we only use the information of keyframes to calculate the keyframe consistency loss, there is a significant discontinuity at keyframes. By also imposing a temporal consistency constraint on the postures near the keyframes, the generated dance sequence can achieve smooth transitions at the given keyframes. In the training process, we set $m=5$. In experiments, we will validate the superiority of the proposed loss function.

Velocity factor consistency loss. In order to make the velocity factors of the synthesized dance sequences consistent with the given control condition, the velocity factor consistency loss $\mathcal{L}_{vfac}$ is proposed. After obtaining the network's outputs, we calculate the velocity factor $\tilde{\mathbf{f}}_t$ for each frame of the generated dance motion. We formulate the velocity factor consistency loss as follows:

$$\mathcal{L}_{vfac}=\frac{1}{N}\sum_{t=k_1+1}^{k_2}\|\tilde{\mathbf{f}}_t-\hat{\mathbf{f}}_t\|^2, \tag{11}$$

where $\hat{\mathbf{f}}_t$ is the constrained velocity factor given by the user.

3.4. Traning details

The dance database we use contains a total of 123 pieces of contemporary dance. Among them, 80% of the data set, a total of 98 dance segments, 93347 frames are used for training, and the remaining 20% of data, a total of 25 segments, 20897 frame samples are used for testing. For the network, the state encoder, the velocity encoder, and the target controller are all composed of two fully connected layers. The hidden units for the two layers are set as 512 and 256, respectively. The position encoder consists of one fully connected layer with 512 hidden units. For the Transformer-based controllers, *i.e.*, the root trajectory controller and the velocity factor controller, the temporal window size u is 7, the head number and the embedding dimension for the MHSA block are 8 and 32 respectively, and the dimensions for the three linear layers are set as 256. For the two LSTM blocks, the hidden units are set to 256. Both the state decoder and the root decoder consist of three fully connected layers, where the dimensions for the two hidden layers are 512 and 256, respectively. We use the Parametric Rectified Linear Unit (PReLU) [HZRS15] as the activation function for all the encoders, decoders, and the target controller, while for the two Transformer-based controllers, we follow the same activation function and structure as the typical Transformer does [VSP*17].

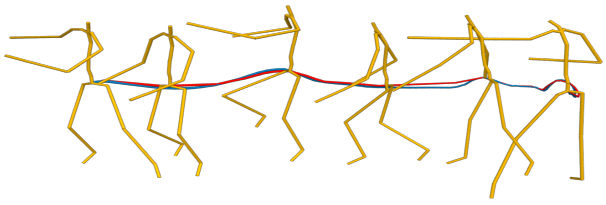
We implement our model using PyTorch, and all models used in this paper are trained on a GeForce RTX 2080 Ti GPU. When training the model, we use the Adam optimizer [KB15] and set the learning rate to 0.0001 and the batch size to 128. It takes 43 hours to train the proposed network and 0.0085 seconds to generate a single frame during testing. Since our model can generate variable-length motion transition sequences, we also use variable-length dance sequences to train the network. The minimum sequence length *min_len* used in training is 5 and the maximum sequence length *max_len* is 70. For each epoch, we set the current *min_len* and *max_len* to represent the sequence length range of the current epoch. At the early stages of training, both *min_len* and *max_len* are set to 5, and the *max_len* increases by 1 after each epoch. In this way, the amount of training data in each round gradually increases as the training progresses. To prevent repeated learning on dance sequences of small lengths, we let *min_len* increase by 4 every five epochs. When *max_len* is greater than 70, the training process is terminated.

4. Experiments and results

4.1. Different control constraints

Our model can achieve fine-grained control for dance movement generation through the skillfully designed control constraints. We conduct experiments to verify the effects of different control constraints with only one of the constraints changed at a time.

Different root trajectories. In order to verify the effect of the root trajectory, we visualize the generated root trajectory and the specified control trajectory to compare the difference between them in Figure 5. It can be observed that the motion sequence generated by our model can well fit the root trajectory conditions specified by the user. Meanwhile, we also defined a quantitative index to evaluate the accuracy of the predicted root trajectory. The quantitative index is explained in the ablation experiment (Section 4.2).

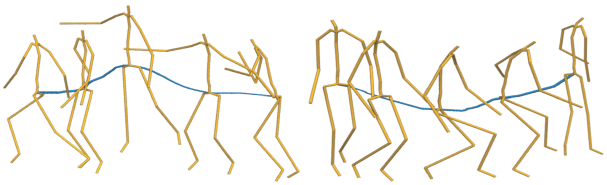


(a) Motion sequence

(b) Root trajectory

Figure 5: *Visualization of the generated motion sequence (a) and the related root trajectories (b), where the target trajectory is blue and the generated trajectory is red.*

When the user provides different root trajectories or different root heights, it should produce different actions. For example, if the root joint reaches a high position, a jump action should be generated, and a squat action should be produced if the root joint is low. The phenomenons can be observed in our results (Figure 6), which demonstrates the close correlations between the generated motions



(a) Jump (b) Squat

Figure 6: *Visualization of the generated motion sequences under different root trajectories.*

(a) Root trajectory 1

(b) Root trajectory 2

Figure 7: *Visualization of the generated motion sequences (the first and last postures are the keyframe postures) under different root trajectories (the red one is the generated motion and the blue one is the ground truth) with other conditions the same.*

and the height of the root joint. Furthermore, we conduct experiments with totally different root trajectories while maintaining the same keyframes and the velocity factors. The results are illustrated in Figure 7. We can observe that various transition movements are generated, and the diversity of dance motion is significantly enhanced under the control of the root trajectory. The above experiments validate that the root trajectory constraint can well control the global position of joints, affect the types of actions, and promote the diversity of dances.

Different velocity factors. When other conditions are the same, that is, the keyframes are the same, and the root trajectory constraint is also the same, we change the values of velocity factors to generate different motion sequences. When using different velocity factor sequences, motion sequences with different speeds will be generated. In order to enable users to control the overall speed better, we allow users to use a constant value to achieve multiple changes to the velocity factor sequences. Figure 8 illustrates the generated dance sequences with the velocity factors varied according to the multiples of 1.0, 0.5, and 1.5. We visualize the generated motion movements every five frames. From Figure 8 and the supplementary video, we observe that when the velocity factors become half of the original, the overall speed of the motion sequence becomes very slow, and the variety of actions is significantly reduced. Besides, the motion movements tend to go directly from one keyframe to another keyframe. However, when the velocity factors increase 1.5 times, the overall movements become faster and change dramatically. Because of the enlarged velocity factors, the generated action will not slide directly from one keyframe to another keyframe, and the diversities of intermediate movements have been promoted to a large extent.

The above experimental results illustrate that the same multiple changes are applied to the five velocity factors. Beyond that, users can assign different velocity factors to different body parts to achieve more fine-grained motion control. We implement experiments to change the velocity factors of different body parts. The generated dance movements are displayed every five frames in Figure 9, where Figure 9 (a) is the synthesized motion sequence with the original velocity factors, while Figure 9 (b) is the generated dances with the blue body parts (right arm and left leg) higher velocity factors. It can be observed that the blue parts generate more active results compared with the yellow parts. The experiments further validate that the diversities of synthesized dances can be enhanced by varying the velocity factor constraint.

(a) Original velocity factors

(b) 1.5 $\times$ velocity factors (blue parts)

Figure 9: *Visualization of the generated motion sequences under different velocity factors with other conditions the same.*

Motion transitions with different lengths. To verify the generation ability of our model and the diversities of generated motions, we use the same keyframes to generate long-term motion transition sequences of 100, 150, 200, and 250 frames, respectively. We visualize the four generated dance sequences in 25 frames in Figure 10. It turns out that the intermediate actions are still diversified when the length of the generated sequence is 200 frames. When the length exceeds 200 frames, repeated meaningless actions begin to appear, which can be seen in the last row of Figure 10 and the supplementary video. The results further reflect the importance of the two control conditions, *i.e.*, the root trajectory and the velocity factor, to the generation of the action. The existence of the two terms can make the synthesized action more diversified and realistic.

4.2. Ablation study

We conduct ablation experiments to verify the superiority of different modules of our network. We define three quantitative metrics: the evaluation criterion of joint position, the evaluation criterion of velocity factor, and the evaluation criterion of root trajectory. The units of the three metrics are cm, cm/frame and cm, respectively.

Inspired by the L2P criteria [HYNP20], we propose the average L2 distances of root-relative position (LRP) to measure the deviation between the predicted motions and their ground truth:

$$LRP = \frac{1}{|\mathcal{D}|}\frac{1}{N}\sum_{c\in\mathcal{D}}\sum_{t=k_1+1}^{k_2}\|\mathbf{p}_t^b(c)-\mathbf{p}_t^b(c)\|_2, \qquad (12)$$

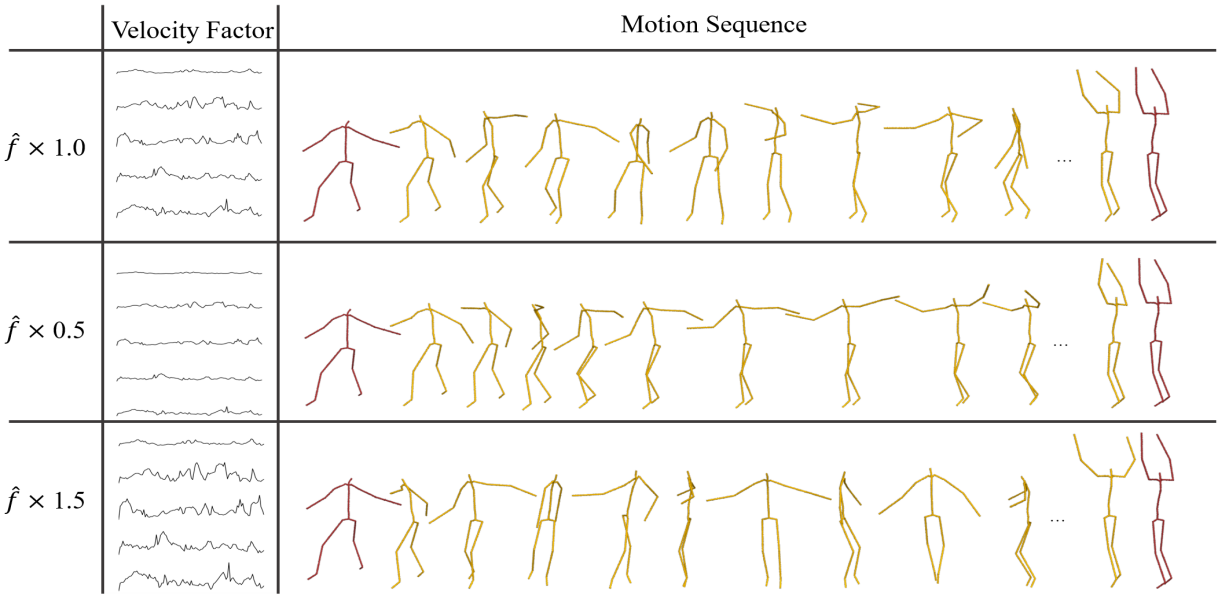


Figure 8: *Visualization of the generated motion sequences under different velocity factors with other conditions the same, where the first and last postures are the keyframe postures.*

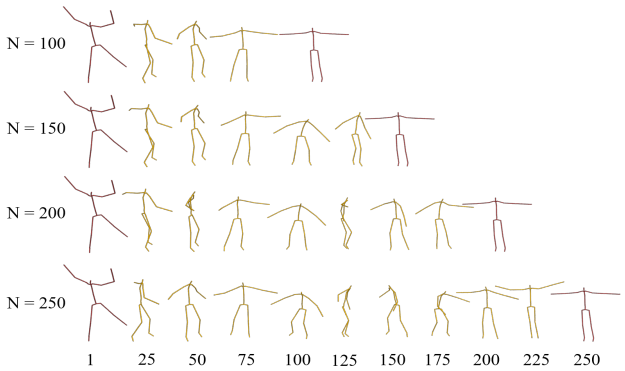


Figure 10: *Visualization of the generated motion sequences (the first and last postures are the keyframe postures) under different lengths with other conditions the same.*

where $\mathcal{D}$ is a test set, $|\mathcal{D}|$ is the number of $\mathcal{D}$, and c is a transition sequence of $\mathcal{D}$. $\tilde{\mathbf{p}}_t^b(c)$ refers to the root-relative positions of skeleton joints at time t in sequence c, and $\mathbf{p}_t^b(c)$ is the related ground truth. The smaller the distance error is, the more accurate the predicted movements are.

We use the accuracy rates for the evaluation indicators for both velocity factor and root trajectory. We define two fault-tolerant thresholds, δ_v (in cm/frame) and δ_r (in cm), to calculate the corresponding accuracies. We calculate the difference between the ground truth and the inferred result from the predicted motion sequence for the velocity factor. If the difference is less than δ_v, the result is correct; otherwise, it is considered to be out of the range of correct values. The average accuracy of the velocity factor (AVF) can be obtained by:

$$AVF = \frac{1}{|\mathcal{D}|}\sum_{c\in\mathcal{D}}\begin{cases}1, & if \quad g_v(c) < \delta_v,\\ 0, & otherwise,\end{cases} \tag{13}$$

$$g_v(c) = \frac{1}{N}\frac{1}{M}\sum_{t=k_1+1}^{k_2}\sum_{i=1}^{M}|\tilde{f}_t^i(c) - \hat{f}_t^i(c)|, \tag{14}$$

where $g_v(c)$ is the deviation of the velocity factor for dance sequence c, $\tilde{f}_t^i(c)$ is the predicted value of body part i at timestep t in sequence c, and $\hat{f}_t^i(c)$ is the corresponding ground truth.

The average accuracy of the root trajectory (ART) can be obtained similarly as the AVF:

$$ART = \frac{1}{|\mathcal{D}|}\sum_{c\in\mathcal{D}}\begin{cases}1, & if \quad g_r(c) < \delta_r,\\ 0, & otherwise,\end{cases} \tag{15}$$

$$g_r(c) = \frac{1}{N}\sum_{t=k_1+1}^{k_2}\|\tilde{\mathbf{p}}_t^r(c) - \hat{\mathbf{p}}_t^r(c)\|_2, \tag{16}$$

where $g_r(c)$ indicates the deviation of root trajectory for dance sequence c, $\tilde{\mathbf{p}}_t^r(c)$ is the predicted root joint position at timestep t of sequence c, and $\hat{\mathbf{p}}_t^r(c)$ is the ground truth. During the experiment, we take δ_v as 1.0 and δ_r as 7.0.

We report the ablation results of the three quantitative metrics under different model settings in Table 2, Table 3, and Table 4. Specifically, the "One LSTM" model represents that the generation model contains one LSTM unit to deal with postural information and control conditions. "Condition-FC" refers to replace the Transformer-based controllers with two fully connected layers to encode the constraints of root trajectory and velocity factor. "One decoder" means that the state decoder decodes the posture information of all skeleton joints, including the root joint. "Without Velfac

constraint" stands for the model without the velocity factor constraint. "Without $\mathcal{L}_{key}$" denotes we train the whole model without the keyframe consistency loss $\mathcal{L}_{key}$. "Whole model" is the overall framework learned with all loss functions. "Interpolation" is the model that utilizes the interpolation strategy to synthesize transition action between keyframes. At last, Harvey's method [HYNP20] in terms of LRP in Table 2 has also been reported.

Table 2: *The LRP evaluation for the transition sequence at the length of 10, 50, 100, 150, and the average result (AVG). The best results are shown in bold.*

Models	Frames				AVG
	10	50	100	150	
Interpolation	**13.83**	87.59	113.59	124.47	84.87
Harvey *et al.* [HYNP20]	141.79	116.40	201.43	297.22	189.21
One LSTM	32.81	70.49	88.11	100.81	73.06
Condition-FC	37.30	67.03	85.20	96.39	71.48
One decoder	28.36	51.15	65.04	77.24	55.45
Without Velfac constraint	34.37	62.30	83.81	97.52	69.50
Without $\mathcal{L}_{key}$	41.59	82.95	97.31	105.46	81.83
Whole model	27.07	**50.06**	**63.01**	**74.37**	**53.63**

Table 3: *The AVF evaluation for transition sequence at the length of 10, 50, 100, 150, and the average result (AVG). The best results are shown in bold.*

Models	Frames				AVG
	10	50	100	150	
One LSTM	0.68	0.75	0.70	0.67	0.70
Condition-FC	0.68	0.71	0.68	0.65	0.68
One decoder	0.71	0.75	0.73	0.70	0.72
Without Velfac constraint	0.60	0.60	0.53	0.49	0.56
Without $\mathcal{L}_{key}$	0.66	0.68	0.66	0.64	0.66
Whole model	**0.72**	**0.76**	**0.75**	**0.72**	**0.74**

Table 4: *The ART evaluation for transition sequence at the length of 10, 50, 100, 15, and the average result (AVG). The best results are shown in bold.*

Models	Frames				AVG
	10	50	100	150	
One LSTM	0.63	0.74	0.76	**0.74**	0.72
Condition-FC	0.47	0.64	0.67	0.67	0.61
One decoder	0.63	0.55	0.48	0.39	0.51
Without Velfac constraint	0.62	0.59	0.45	0.34	0.50
Without $\mathcal{L}_{key}$	0.62	0.42	0.33	0.27	0.41
Whole model	**0.86**	**0.85**	**0.80**	0.72	**0.81**

Effectiveness of two LSTM units. To verify the effectiveness of using two LSTM, we conduct experiments by learning the model with one LSTM. It turns out that the dance sequence generated with one LSTM performs less well than the "Whole model", leading to much larger LRP errors. The results can be observed in Table 2. For the velocity factor, the accuracy produced by the "Whole model" is much higher than the "One LSTM", as shown in Table 3. As for the root trajectory, when the sequence length is less than 100, the accuracy of the "Whole model" far outstrips the model learned with one LSTM. However, when the predicted length starts larger than 150, the "Whole model" is slightly worse than "One LSTM". The reason may be that one LSTM unit cannot reach a balance between various control conditions. Instead, it tends to dominate the control of the root trajectory. Hence the results of posture and velocity factor become worse. On the contrary, by processing different information hierarchically and sequentially, the whole model with two LSTM reaches a good balance between different constraints and achieves better results on the three criteria.

Effectiveness of Transformer controller. Due to the multi-head self-attention layer mechanism in Transformer, the Transformer-based controllers can well leverage the context information in the time window of the frame to be predicted to encode the control signals. We use the fully connected layer to replace the Transformer to validate its importance on constraint control. By comparing the "Condition-FC" with the "Whole model" in Table 2, Table 3, and Table 4, we observe that the performance of the complete model is far better than that of the model constraint with the fully connected layer in terms of the joint position error, velocity factor and root joint prediction accuracy, demonstrating the superiority of Transformer structure in constraints modulation.

Effectiveness of the root decoder. Existing methods usually employ the state decoder to predict the posture information of all skeleton joints [HYNP20, HP18]. However, for dance, the action space of dance movements is vast. Only one decoder may cause large jitter in the generated root trajectory. To verify the effectiveness of the root decoder, we conduct experiments to obtain the prediction results with only the state decoder. Comparing the experimental results of "One decoder" and "Whole model" in Table 2, Table 3, and Table 4, we can observe that the LRP and the AVF of "One decoder" are similar to those of the complete model, but the ART is inferior. We also illustrate the synthesized dance sequences and the corresponding root trajectories of "One decoder" and "Whole model" in Figure 11. It can be observed that the root trajectory obtained by using one decoder has a larger deviation from the target root trajectory, while the result obtained by using two decoders is more consistent with the target root trajectory, which validates the advantages of the root decoder.

(a) One decoder

(b) Two decoders

Figure 11: *Visualization of the generated motion sequences (the first and last postures are the keyframe postures) and root trajectories (the red one is the generated motion and the blue one is the ground truth).*

Effectiveness of velocity factor constraint. We implement experiments to verify the affect of the velocity factor constraint in motion synthesis by removing the corresponding controller. It can be seen

from Table 3 when velocity factors are not used for control, the AVF evaluation metric is 56%. After adding the velocity factor constraint, the AVF performance has improved to 74%. In addition, the whole model has significantly improved the model "Without Velfac constraint" in terms of the evaluation criteria of LRP and AVT as illustrated in Table 2 and Table 4. Three quantitative experimental results confirm the importance of the velocity factor constraint.

Effectiveness of keyframe consistency loss. Keyframe consistency loss aims to constrain the motion sequence transited natural and smooth near keyframes. To verify its importance, we conduct experiments by learning the network without the $\mathcal{L}_{key}$ function. In the experiment, we find that the performance of the three quantitative evaluation indexes of the model is greatly improved by adding the keyframe consistency loss. The detailed results are illustrated in Table 2, Table 3, and Table 4. The achieved significant gains demonstrate that the keyframe consistency loss has a positive effect on the optimization of the network.

4.3. Comparison with other methods

Experiments on dance dataset. We also compare our model with two related methods to validate its superiority. Specifically, we make a comparison with Harvey's method [HYNP20] and the results obtained by interpolation strategy. When training Harvey's model [HYNP20], we employ the same learning strategy on the dance dataset we use. It turns out that the generated motion sequences are far from satisfactory, many synthesized actions are unnatural, and the continuity at the keyframes is poor. The main reason for the results may be that the network is mainly designed for walking, running, and other simple movements, while the dance movements are relatively complex. Its variability makes it impossible to uniquely describe the movements using only keyframes, which may easily cause ambiguity. Therefore, the final synthesized results are performed poorly. For the interpolation strategy, we take the root trajectory and keyframes as the control conditions. We spherically interpolate the quaternions between the keyframes to complete the motion transition.

We also calculated the LRP errors of the two methods. The comparison results are reported in Table 2. It can be observed from Table 2 that within short-term motion transition, *e.g.* $N \leq 10$, the interpolation method shows a decisive advantage. It is partly because the motion becomes almost linearly in a sufficiently short timescale. However, with the increasing length of the prediction sequence, our method obviously exceeds the interpolation-based strategy. In addition, the results in Table 2 also prove that our method is always better than Harvey's model [HYNP20] under different prediction sequence lengths and far exceeds it by a large margin.

We show the visual comparison of the dance sequences generated by our algorithm and the above two algorithms, as well as the ground truth (GT) sequences in Figure 12. It can be seen that the motion generated by Harvey's method [HYNP20] are noisy and unnatural, and there are obvious discontinuities at the keyframes. The results obtained by interpolation lose the diversity of actions, and the steps are always floating. In contrast, our method improves the problem above and can generate more continuous and smooth motion movements. The qualitative and quantitative comparative experiments demonstrate that our model has better generation ability in complex dance synthesis and can achieve fine-grained control through the root trajectory and velocity factor constraints.

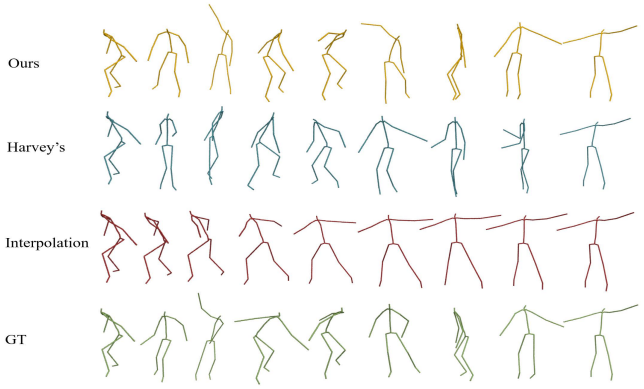


Figure 12: *Visualization of the generated dance sequences by different methods and the ground truth.*

Experiments on cyclic motion dataset. We select 97,123 frames related to walking and running from the dataset used by Harvey. To make a fair comparison, we train our model and Harvey's method [HYNP20] under the same environments using the selected motion samples. The generated results are illustrated in Figure 13, as well as in the supplementary video. Since Harvey's method does not control the global position, we modify the root joint's position and rotation of the synthesized results to avoid the influence of global information on the visual effect. From Figure 13 and the supplementary video, we observe that Harvey's method can generate natural and smooth motion transitions between keyframes for running motion. In contrast, our method has achieved comparable performance to Harvey's method on the cyclic motion dataset from the visual effect. However, when it comes to the challenging non-cyclic dance synthesis, our model is far better than Harvey's method in the naturalness and variety of dance movements, as shown in Figure 12 and the supplementary video. The experimental results validate that our model can not only control complex non-cyclic dance generation, but is also suitable for simple cyclic locomotion synthesis. Therefore, our approach demonstrates good robustness on a variety of datasets.

5. Limitations and discussions

Our method can achieve fine-grained control of complex movements, but there are still some limitations.

Firstly, the generated dance motion have the footstep floating problem. We introduce the foot contact loss in Eq. 7 to constrain footsteps, which is inspired by [WHSZ21]. The authors have verified its effectiveness on periodic simple actions, which is consistent with our model on running motion. However, dance movements are more complicated compared with cyclic motion, Eq. 7 can only alleviate the foot skating to some extent, it hardly solves this problem entirely. Besides, since there are many loss terms in our network, and the foot contact labels and footstep speeds in Eq. 7 appear in multiple loss terms, the network needs to achieve equilibrium among all the loss terms. Therefore, the importance of Eq. 7 may be

Figure 13: *Visualization of the generated running sequences by different methods and the ground truth.*

reduced. Thus, the final generated dance sequences have a footstep floating problem. At present, the most effective method to the problem is to use some post-processing methods, such as IK [ALCS18] or refer to Harvey's method to use GAN [HYNP20], which will be the focus of our future work.

Secondly, when providing control conditions that differ significantly from the training set, our method produces poor results similar to many data-driven tasks. This problem can be avoided by expanding the dataset or restricting the control conditions.

Last but not least, since our model has many control constraints, in rare cases, the user may give some extremely contradictory control conditions, which will also cause the unreality of the results. In the future, the relationship between different control conditions can be explored to generate more realistic actions.

6. Conclusions

In this paper, we propose a complex motion generation network from keyframes, which can realize fine-grained control for complex motion through the trajectory sequence of the root joint and the velocity factors of different body parts. LSTM and Transformer are combined to handle control conditions so that our method is able to generate motion sequences in variable length. We have carried out a large number of comparative experiments to demonstrate the results under different constraints. Meanwhile, we have conducted three categories of quantitative evaluation and ablation experimental analysis, and have compared our algorithm with the state-of-the-art methods, which proves the superiority of our network. In the future, we will focus on the footstep floating problem and multiple constraint contradiction issue in complex motion synthesis. The source code will be released after the paper is accepted. [https://github.com/godzillalla/Dance-Synthesis-Project].

Acknowledgments. This work was supported in part by National Key R&D Program of China (NO.2018YFC0115102), National Natural Science Foundation of China (No.61872020, U20A20195), Beijing Natural Science Foundation Haidian Primitive Innovation Joint Fund (L182016), Shenzhen Research Institute of Big Data, Shenzhen, 518000, China Postdoctoral Science Foundation (2020M682827), Baidu academic collaboration program, and Global Visiting Fellowship of Bournemouth University.

References

[ALCS18] ARISTIDOU A., LASENBY J., CHRYSANTHOU Y., SHAMIR A.: Inverse kinematics techniques in computer graphics: A survey. *Computer Graphics Forum 37*, 6 (2018), 35–58. 12

[AZS*17] ARISTIDOU A., ZENG Q., STAVRAKIS E., YIN K., COHEN-OR D., CHRYSANTHOU Y., CHEN B.: Emotion control of unstructured dance movements. In *Proceedings of the ACM SIGGRAPH / Eurographics Symposium on Computer Animation* (Los Angeles, CA, USA, 2017), Eurographics Association / ACM, pp. 9:1–9:10. 3

[BDK20] BANAR N., DAELEMANS W., KESTEMONT M.: Character-level transformer-based neural machine translation. In *International Conference on Natural Language Processing and Information Retrieval* (Seoul, Republic of Korea, 2020), ACM, pp. 149–156. 4

[CH07] CHAI J., HODGINS J. K.: Constraint-based motion optimization using a statistical dynamic model. *ACM Transactions on Graphics 26*, 3 (2007), 8. 3

[CMS*20] CARION N., MASSA F., SYNNAEVE G., USUNIER N., KIRILLOV A., ZAGORUYKO S.: End-to-end object detection with transformers. In *European Conference on Computer Vision* (Glasgow, UK, 2020), Vedaldi A., Bischof H., Brox T., Frahm J., (Eds.), vol. 12346, Springer, pp. 213–229. 4

[dat] http://dancedb.eu/main/performances. 3

[ECC*20] EOM H., CHOI B., CHO K., JUNG S., HONG S., NOH J.: Synthesizing character animation with smoothly decomposed motion layers. *Computer Graphics Forum 39*, 1 (2020), 595–606. 2

[FCG*21] FERREIRA J. P. M., COUTINHO T. M., GOMES T. L., NETO J. F., AZEVEDO R., MARTINS R., NASCIMENTO E. R.: Learning to dance: A graph convolutional adversarial network to generate realistic dance motions from audio. *Computers and Graphics 94* (2021), 11–21. 3

[GLSR19] GAISBAUER F., LEHWALD J., SPRENGER J., RUKZIO E.: Natural posture blending using deep neural networks. In *Motion, Interaction and Games* (Newcastle upon Tyne, UK, 2019), Shum H. P. H., Ho E. S. L., Cani M., Popa T., Holden D., Wang H., (Eds.), ACM, pp. 2:1–2:6. 3

[HHW*21] HUANG R., HU H., WU W., SAWADA K., ZHANG M., JIANG D.: Dance revolution: Long-term dance generation with music via curriculum learning. In *International Conference on Learning Representations* (2021). 3

[HP18] HARVEY F. G., PAL C. J.: Recurrent transition networks for character locomotion. In *SIGGRAPH Asia 2018 Technical Briefs* (Tokyo, Japan, 2018), Zafar N. B., Zhou K., (Eds.), ACM, pp. 4:1–4:4. 3, 10

[HSK16] HOLDEN D., SAITO J., KOMURA T.: A deep learning framework for character motion synthesis and editing. *ACM Transactions on Graphics 35*, 4 (2016), 138:1–138:11. 1, 2

[HYNP20] HARVEY F. G., YURICK M., NOWROUZEZAHRAI D., PAL C. J.: Robust motion in-betweening. *ACM Transactions on Graphics 39*, 4 (2020), 60. 1, 3, 8, 10, 11, 12

[HZRS15] HE K., ZHANG X., REN S., SUN J.: Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *IEEE/CVF International Conference on Computer Vision* (Santiago, Chile, 2015), IEEE, pp. 1026–1034. 7

[KB15] KINGMA D. P., BA J.: Adam: a method for stochastic optimization. In *International Conference on Learning Representations (ICLR)* (San Diego, CA, USA, 2015). 7

[KGP02] KOVAR L., GLEICHER M., PIGHIN F. H.: Motion graphs. *ACM Transactions on Graphics 21*, 3 (2002), 473–482. 3

[KPS03] KIM T., PARK S. I., SHIN S. Y.: Rhythmic-motion synthesis based on motion-beat analysis. *ACM Transactions on Graphics 22*, 3 (2003), 392–401. 3

[LL20] LIN J., LEE G. H.: Hdnet: Human depth estimation for multi-person camera-space localization. In *European Conference on Computer Vision* (Glasgow, UK, 2020), Vedaldi A., Bischof H., Brox T., Frahm J., (Eds.), vol. 12363, Springer, pp. 633–648. 6

[LLL18] LEE K., LEE S., LEE J.: Interactive character animation by learning multi-objective control. *ACM Transactions on Graphics 37*, 6 (2018), 180:1–180:10. 2

[LLP13] LEE M., LEE K., PARK J.: Music similarity-based approach to generating dance motion sequence. *Multimedia tools and applications 62*, 3 (2013), 895–912. 3

[LYL*19] LEE H., YANG X., LIU M., WANG T., LU Y., YANG M., KAUTZ J.: Dancing to music. In *Conference on Neural Information Processing Systems* (Vancouver, BC, Canada, 2019), Wallach H. M., Larochelle H., Beygelzimer A., d'Alché-Buc F., Fox E. B., Garnett R., (Eds.), pp. 3581–3591. 3

[MBR17] MARTINEZ J., BLACK M. J., ROMERO J.: On human motion prediction using recurrent neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition* (Honolulu, HI, USA, 2017), IEEE Computer Society, pp. 4674–4683. 2

[MCC09] MIN J., CHEN Y., CHAI J.: Interactive generation of human animation with deformable motion models. *ACM Transactions on Graphics 29*, 1 (2009), 9:1–9:12. 3

[MZZ*19] MA X., ZHANG P., ZHANG S., DUAN N., HOU Y., ZHOU M., SONG D.: A tensorized transformer for language modeling. In *Conference on Neural Information Processing Systems* (Vancouver, BC, Canada, 2019), Wallach H. M., Larochelle H., Beygelzimer A., d'Alché-Buc F., Fox E. B., Garnett R., (Eds.), pp. 2229–2239. 4

[RGM19] RUIZ A. H., GALL J., MORENO F.: Human motion prediction via spatio-temporal inpainting. In *IEEE/CVF International Conference on Computer Vision* (Seoul, Korea (South), 2019), IEEE, pp. 7133–7142. 2

[SNI06] SHIRATORI T., NAKAZAWA A., IKEUCHI K.: Dancing-to-music character animation. *Computer Graphics Forum 25*, 3 (2006), 449–458. 3

[TJM18] TANG T., JIA J., MAO H.: Dance with melody: An lstm-autoencoder approach to music-oriented dance synthesis. In *ACM Multimedia Conference on Multimedia Conference* (Seoul, Republic of Korea, 2018), Boll S., Lee K. M., Luo J., Zhu W., Byun H., Chen C. W., Lienhart R., Mei T., (Eds.), ACM, pp. 1598–1606. 3

[VSP*17] VASWANI A., SHAZEER N., PARMAR N., USZKOREIT J., JONES L., GOMEZ A. N., KAISER L., POLOSUKHIN I.: Attention is all you need. In *Conference on Neural Information Processing Systems* (Long Beach, CA, USA, 2017), Guyon I., von Luxburg U., Bengio S., Wallach H. M., Fergus R., Vishwanathan S. V. N., Garnett R., (Eds.), pp. 5998–6008. 2, 4, 5, 7

[WCX21] WANG Z., CHAI J., XIA S.: Combining recurrent neural networks and adversarial training for human motion synthesis and control. *IEEE Transactions on Visualization and Computer Graphics 27*, 1 (2021), 14–28. 2

[WFH08] 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 (2008), 283–298. 3

[WHSZ21] WANG H., HO E. S. L., SHUM H. P. H., ZHU Z.: Spatio-temporal manifold learning for human motions via long-horizon modeling. *IEEE Transactions on Visualization and Computer Graphics 27*, 1 (2021), 216–227. 2, 6

[YWJ*20] YE Z., WU H., JIA J., BU Y., CHEN W., MENG F., WANG Y.: Choreonet: Towards music to dance synthesis with choreographic action unit. In *ACM Multimedia Conference on Multimedia Conference* (Seattle, WA, USA, 2020), Chen C. W., Cucchiara R., Hua X., Qi G., Ricci E., Zhang Z., Zimmermann R., (Eds.), ACM, pp. 744–752. 3

[ZFS*20] ZHEN J., FANG Q., SUN J., LIU W., JIANG W., BAO H., ZHOU X.: SMAP: single-shot multi-person absolute 3d pose estimation. In *European Conference on Computer Vision* (Glasgow, UK, 2020), Vedaldi A., Bischof H., Brox T., Frahm J., (Eds.), vol. 12360, Springer, pp. 550–566. 6

[ZJJ*20] ZHAO H., JIANG L., JIA J., TORR P. H. S., KOLTUN V.: Point transformer. *arXiv preprint arXiv:2012.09164* (2020). 6

[ZS09] ZHAO L., SAFONOVA A.: Achieving good connectivity in motion graphs. *Graphical Models 71*, 4 (2009), 139–152. 3

[ZvdP18] ZHANG X., VAN DE PANNE M.: Data-driven autocompletion for keyframe animation. In *In Proceedings of International Conference on Motion, Interaction, and Games* (Limassol, Cyprus, 2018), ACM, pp. 10:1–10:11. 1, 3

[ZZT*20] ZHANG D., ZHANG H., TANG J., WANG M., HUA X., SUN Q.: Feature pyramid transformer. In *European Conference on Computer Vision* (Glasgow, UK, 2020), Vedaldi A., Bischof H., Brox T., Frahm J., (Eds.), vol. 12373, Springer, pp. 323–339. 4