arXiv:2103.10206v3 [cs.AI] 25 Mar 2021

DanceNet3D: Music Based Dance Generation with Parametric Motion Transformer

Buyu Li* [1,3] Yongchi Zhao* [1] Lu Sheng[2]
[1]Huiye Technology [2]College of Software, Beihang University [3]MMLab, CUHK
{libuyu, zhaoyongchi}@huiye.tech, lsheng@buaa.edu.cn

Abstract

In this work, we propose a novel deep learning framework that can generate a vivid dance from a whole piece of music. In contrast to previous works that define the problem as generation of frames of motion state parameters, we formulate the task as a prediction of motion curves between key poses, which is inspired by the animation industry practice. The proposed framework, named DanceNet3D, first generates key poses on beats of the given music and then predicts the in-between motion curves. DanceNet3D adopts the encoder-decoder architecture and the adversarial schemes for training. The decoders in DanceNet3D are constructed on MoTrans, a transformer tailored for motion generation. In MoTrans we introduce the kinematic correlation by the Kinematic Chain Networks, and we also propose the Learned Local Attention module to take the temporal local correlation of human motion into consideration. Furthermore, we propose PhantomDance, the first large-scale dance dataset produced by professional animators, with accurate synchronization with music. Extensive experiments demonstrate that the proposed approach can generate fluent, elegant, performative and beat-synchronized 3D dances, which significantly surpasses previous works quantitatively and qualitatively. The project link is https://huiye-tech.github.io/project/dancenet3d/

1. Introduction

Dance generation from music has become an increasingly attractive research topic in recent years, whereas it is still one of the most challenging tasks. Different from common motions like walking, jumping and sitting that are evaluated by their functionalities, dance is an important form of artistic performance that focuses more on power, fluency and elegance. Most of the previous works formulate the output of the dance generation task as a sequence of motion

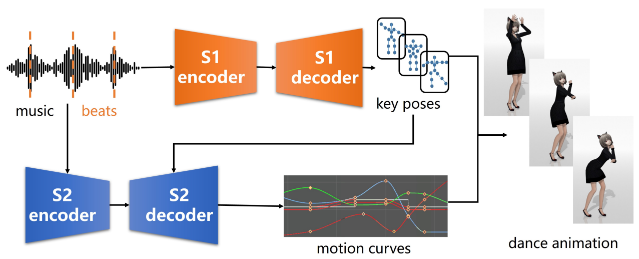

Figure 1. **Overview of the DanceNet3D pipeline.** The networks of stage 1 takes as input the music frequency spectrum coefficients around each beat. And an encoder-decoder model with transformer structure generates a sequence of key poses conditioned on the music feature sequences. Then the joints parameters of key poses and the spectrum features of the in-between music are feed into the encoder-decoder model of stage 2 to predict the in-between motion curves.

state parameters [8, 11, 9, 17, 10, 22, 1], e.g., the positions and rotations of joints at each frame. However, in animation industry the motions of characters are mostly animated by key frames and curves [7, 18]. According to the animation principles [20, 7], the curve representation can avoid unnecessary small flickers and make an action fluent. Inspired by these facts, we propose a new perspective to model the dance generation problem as key poses generation and subsequent motion curve generation.

Dance choreography strongly depends on the music, where movements are supposed to be designed based on the rhythm and beat of the song. Furthermore, recent study has demonstrated the co-occurrence of music beats and kinematic beats [8]. Therefore, it is convenient to assume that each key pose of the dance corresponds to a beat of the music and they have an identical occurrence time. Therefore, we apply a beat track algorithm [4] to detect the beats of a music, and we generate key poses based on the music spectrum features on the windows centered at the beats, where the poses are parametrically formulated as joint translation (t_x, t_y, t_z) and rotation (q_x, q_y, q_z, q_w) in the form of quaternion. Given a sequence of key poses, the motions between adjacent key poses are parametrically represented as Cubic Hermite Spline that interpolates the key poses with

*They contributed equally to this work

respect to time. The key benefit of using spline is that it enables various movements based on a small set of control parameters, which can be learned efficiently through a deep neural network.

To this end, we propose DanceNet3D, a two-stage framework with encoder-decoder architectures and adversarial training scheme. The encoders take the well-known transformer [19] as the basic models. While for the decoders, we propose a novel architecture named MoTrans, that is also based on transformer but has carefully been tailored to the dance generation task. Specifically, in MoTrans we propose the Kinematic Chain Networks (KCN) as the backbone, and the Learned Local Attention (LLA) module with structured multi-head attentions to replace the vanilla attentions.

Inspired from the kinematics that is widely used in robotics and motion control, we propose the Kinematic Chain Networks (KCN) with joints as nodes and bones as edges to model the spatial correlation of body parts. It encodes the features into the kinematic space and conducts feature fusion for transformation controlling forward-and-backward along the kinematic chains. The KCN involves spatial correlation and physics constraints into the model, which leads to more natural results.

Different from sentences in NLP that have long-term correlations, human poses and motions only have local correlations in a short time interval. That is, current pose is strongly related to the neighboring ones, but it only has weak correlation to the poses before or after many seconds. Therefore the global attention in the original transformer is not so reasonable in this task. And we introduce kernels in the attention function and propose the Learned Local Attention (LLA) module with the use of learned kernels. Owing to the LLA module, the training is more stable and the degradation to averaging actions is avoided.

Moreover, we propose PhantomDance, the first 3D dance dataset produced by professional animators. Existing 3D dance datasets are either by reconstruction from 2D videos [8, 11, 9] or by motion capture [22, 17]. The reconstructed data can not ensure the accuracy due to the limitation of recent 3D reconstruction algorithms. The motion capture methods are more accurate but still suffer from noise and misalignment to music. In contrast, our PhantomDance is produced by a team of experienced animators with the help of an expert dancer. The animated dances are more fluent, elegant and expressive, and they are strictly matched with the music rhythm. We will release 100 of the professionally animated dance data with aligned musics as PhantomDance-100 dataset to facilitate future studies.

Extensive experiments are conducted on PhantomDance and AIST++ [11], which demonstrates that DanceNet3D attains state-of-the-art results and significantly surpasses other works both qualitatively and quantitatively.

Above all, the contributions are summarized as follows:

1. We propose a new perspective to model the 3D dance generation as key poses and motion curves generation. This formulation takes animation principles into consideration and fits well with the fluency and elegance of dance performance.

2. *PhantomDance* is the first dance dataset that crafted by professional animators. Since existing datasets collect dance data either by 3D reconstruction from 2D videos or using motion capture devices, PhantomDance is unprecedented. The dances in it are much more fluent, expressive, and synchronized well with music.

3. We propose a novel framework named DanceNet3D, and the MoTran architecture as well. MoTrans consists of Kinematic Chain Networks (KCN) and Learned Local Attention (LLA) modules with structured multi-head attention. It introduces kinematic information and temporal locality into the networks to better handle the dance generation task.

2. Related Work

Dance Datasets: Large datasets are the cornerstone of recent advances in motion synthesis using deep learning. The widely used motion datasets, such as Human3.6M [6] and AMASS [14], collect large amount of common actions like walking, running, jumping and sitting. However, these datasets are hard to use for dance generation due to the limited motion data distribution and the absence of music-dance pairs. In reality, 3D dance movements of high quality with synchronized music are extremely hard to collect and many of existing datasets are limited by quantity or quality of data. Some of them just use 2D keypoints but no 3D data [8, 9]. AIST++ [11], as the largest dataset to date, presented a 5 hours long 3D dance dataset. But it also reconstructs 3D poses from 2D multi-view videos, which has limitation of accuracy especially for the rotation parameters. Other works use motion capture to build dataset [1, 17, 22]. The accuracy is ensured, but they suffer from the problem of disalignment between dance-music pair. The proposed PhantomDance is the first 3D dance dataset crafted by professional animators, which is high-quality, expressive, graceful and strictly aligned with music.

Dance Synthesis: Early works on dance generation are mostly retrieval based methods [16, 2]. Recent years methods based on deep learning attain more and more qualitatively pleasing results. The capability of deep neural networks to generate dances has been demonstrated by many works, such as those using long-short term memory networks (LSTM) [17], convolutional neural networks (CNN) [9, 22] and variational auto-encoder (VAE) [8]. CSGN [21] generates the entire dance sequence altogether with a graph neural network (GNN). The GNN in CSGN grows from root node to leaf nodes, while Kinematic Chain Networks proposed in this work adopt forward and inverse kinematic passing that better involves the physics constraints. And the generated dance of CSGN has no relation to music. As transformer [19] achieves great success in sequence-to-sequence generation tasks in language processing field, some of the most recent research works on the dance generation also bring it into use. The most related works to ours are those using transformers. Jiaman Li [10] presents a two-stream motion transformer generative model. And AI Choreographer [11] presents a cross-model transformer with future-N supervision mechanism for auto-regressive motion prediction. Both of the two works still use fully-connected layers for feature embedding and make no changes on the attention operation. While we propose Kinematic Chain Networks to embed the kinematic information and Learned Local Attention module to enable the model to adapt to the temporal locality of motions. Moreover, all the related works above take the frames of joints parameters as the output of the model, whereas we propose a new formulation that model the output as key poses and the in-between motion curves.

3. Data Collection and Formatting

3.1. PhantomDance Dataset

We first collect 300 popular dance videos on YouTube and NicoNico. Their categories are mainly in Otaku Dance, Hip-hop, Jazz and Old School. Then a team of experienced animators make the dance animations according to the collected music and videos. An expert dancer is also asked to provide instructions on dance acting for the animators, so that the animated dances are graceful and strictly matched with music rhythm. It costs about 18 months for the animators to produce the 300 dance animations, which have 12 hours (about 2.6M frames) in total. These data are named PhantomDance, and we will release 100 of them as the PhantomDance-100 Dataset, which has 794776 frames (about 3.7 hours).

The dances are animated in Maya [3] on a standard SMPL [13] character model with 24 body joints. Although the animations in Maya are edited by key-frames and curves, they are exported as 60-FPS pose sequences for convenient use of the dataset. A pose of character is represented as global position and rotation (in quaternion) for each of the 24 joints.

3.2. Key Pose Extraction

We first leverage the a dynamic programming algorithm [4] to track the time of each beat in the music. Previous study shows that music beats and kinematic inflexion points have strong consistency in time [8], and moreover our dance data are highly matched with rhythm. So we use the poses corresponding to the beat times directly as the key poses. A key pose is represented as position and rotation (in quaternion) of each joint, with the dimension of $(3 + 4) \times 24 = 168$.

3.3. Motion Curve Formulation

On the basis of key poses, the dance is split into several pieces. And the in-between motions are pose frames in original dataset. Our target is to model the motions as curves on the transformation parameters (x, y, z for position and x, y, z, w for rotation) of joints. Thus a motion in-between consists of 24 x 7 curves, and each curve has two axes, namely the horizontal axis of time and the vertical axis of parameter value. And we use a multi knots cubic Hermite spline [23] to fit each curve. It is simple in parametric form and can keep smooth at endpoints, so it is widely used in industrial design including animation.

Cubic Hermite spline (CHS) uses knots (endpoints) and their tangent (slope) to represent a curve. Let $k_i = (t_i, p_i)$ be the coordinate of the i-th knot and s_i be its slope. The blend form of the CHS between k_i and k_{i+1} is:

$$p(t) = h_{00}(\hat{t})p_i + h_{10}(\hat{t})\bar{s}_i + h_{01}(\hat{t})p_{i+1} + h_{11}(\hat{t})\tilde{s}_{i+1} \quad (1)$$

where $\hat{t} = \frac{t-t_i}{t_{i+1}-t_i}$ is the uniformed time. $\bar{s}_i = \frac{s_i}{t_{i+1}-t_i}$ and $\tilde{s}_{i+1} = \frac{s_{i+1}}{t_{i+1}-t_i}$ are slopes scaled by the interval length. The coefficient functions are specified in Hermite forms:

$$\begin{aligned} h_{00}(t) &= 2t^3 - 3t^2 + 1, \quad & h_{10}(t) &= t^3 - 2t^2 + t \\ h_{01}(t) &= -2t^3 + 3t^2, & h_{11}(t) &= t^3 - t^2 \end{aligned} \quad (2)$$

In implementation we use a 4-knot CHS to fit a motion curve between two key poses, as an optimal trade-off between fitting precision and representation simplicity. For a 4-knot CHS, we have 12 control parameters, namely $k_i = (t_i, p_i)$ and s_i, $i = 1, 2, 3, 4$, to determine. While k_1 and k_4 are just the points corresponding to the two key poses on both ends of the motion curve. So there are 8 control parameters to compute (which are also the output target of the neural networks in stage 2). Since the data in PhantomDance have a 60-FPS format of discrete poses, we just use the neighbor frame points on the two sides of a knot to estimate its slope. Then we set k_2 and k_3 to the two frame points that minimize the fitting error in the frames between the two key poses knots.

4. DanceNet3d

4.1. Pipeline

As illustrated in Fig.1, the networks in stage 1 (S1) generate a sequences of key poses based on the input music. And the networks in stage 2 (S2) predict the motion curves between the key poses with regard to the corresponding music. The encoder adopts a standard transformer following

[19]. While we propose the MoTrans architecture in the decoders of both stages. And adversarial learning is applied during the training in both stages. Details are specified in the following sections.

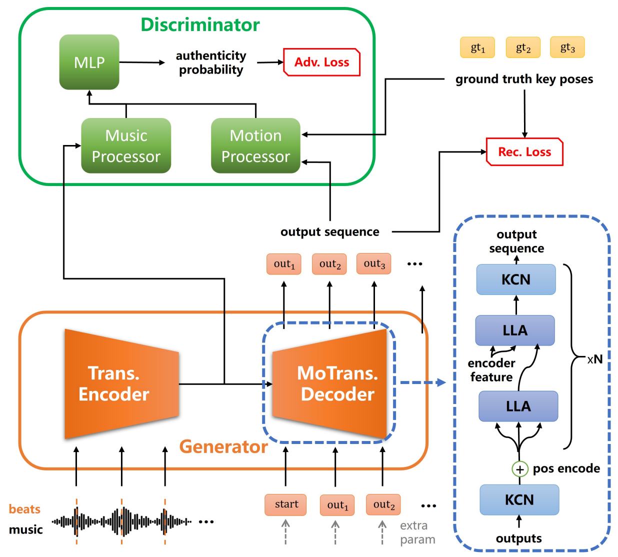

Figure 2. **Training framework of the two stages.** As the architectures of the two stages are quite similar, we summarize them in one figure. An adversarial learning schema is adopted for the training of the networks. The encoder-decoder networks are regarded as generators. The discriminator are targeted to discriminate between the ground truth and generated dance conditioned on the input music. The decoder adopts the proposed MoTrans architecture shown on the right (residual structures are omitted for better view). Note that extra parameters illustrated in gray dashed lines represent the generated key poses from S1, which are only used in S2.

4.1.1 Key Pose Generation

As described in Sec.3.2, we obtain the beat times by an off-the-shelf algorithm [4]. Our target in this stage is to generate a sequence of key poses at the same times of the beats. For each beat, we use a hamming window of 0.5s length centered at it and calculate MFCC [12] on the corresponding wave data. We adopt the 40-dimension form of MFCC that has 13 coefficients of Mel filters cepstrum coefficients, their first and second order differences (13 x 2 dimensions) and the energy sum (1 dimension). We further append a 12-dimension chroma feature to every beat features and finally attain the inputs to the encoder.

As shown in Fig.2, the encoder takes beat features as input and utilizes common self attention modules of the transformer. The attention modules enable the encoder to extract global features of the music (e.g. music style) across beats. The decoder takes right shifted outputs to perform self attention and then combines the features from encoder to perform vanilla attention. In S1 the start parameters refer to a given start pose before all the beats, which can be considered as an opening pose for the dance. During training the start pose is simply selected as the first frame of the dance sequence and the right shifted outputs are just taken from ground-truth with mask for future time steps. While in the test phase any reasonable pose can be used as the start pose, which brings out diversity for dance generation. Note that the network architecture in the decoder is MoTrans that consists of the proposed Learned Local Attention module and Kinematic Chain Networks structure. Details of MoTrans are specified in Sec.4.2.2 and Sec.4.2.1.

The encoder-decoder networks are supposed to map a distribution in the music spectrum space to a distribution in the body pose space. So we adopt an adversarial training architecture that regards the encoder-decoder as a generator. And the discriminator is aimed to determine whether the given key poses are suitable for the input music as well as authentic in human view. Thus we use a 3-layer fully-connected network to embedding the encoded music feature at each time in the sequence, and the embedded features are then average-pooled across the time dimension to obtain the global music features. A similar 3-layer network is employed to extract the global features of the generated/ground-truth key poses. The music and key pose global features are then concatenated and fed into a 3-layer MLP, and the networks finally output a probability of the authenticity of the key pose sequence conditioned on the music.

Let s and $\hat{s}$ represent the generated and ground-truth key pose sequence respectively and $D()$ be the function of discriminator networks. We apply the generally used binary cross entropy loss for the discriminator. Thus the adversarial loss for generator is:

$$L_{adv} = y \log D(\hat{s}) + (1-y)\log(1-D(s)) \tag{3}$$

where y is the authenticity label (1 for ground-truth and 0 for generation). And a L-2 loss of reconstruction is additionally applied to the generator:

$$L_{rec} = ||s - \hat{s}||^2 \tag{4}$$

So the final loss function for the key pose generator is:

$$L_G = L_{adv} + \lambda L_{rec} \tag{5}$$

where λ is the weight to control the contribution of the loss term.

4.1.2 Curve Prediction

Based on the generated key poses, networks in S2 are targeted to generate the curves in between. S2 has a similar adversarial training schema and network architecture as S1, as shown in Fig.2. For a curve to predict, we extract

MFCC features from the music wave data between the two beats corresponding to the two key poses on the ends of the curve. Similar to S1, the 40-dimension MFCC features with 12-dimension chroma information are fed into the encoder. The decoder in S2 outputs 8 control parameters specified in Sec.3.3, and it has a similar right shifted output curve parameters for self attention, except that the output curve parameters are concatenated with the two generated key poses (shown as the extra parameters in Fig.2). The loss function of the curve generator has the same formulation as that of the key pose generator.

4.2. MoTrans Architecture

Both the decoders in S1 and S2 are built with the MoTrans architecture. MoTrans has a similar construction to the original transformer as illustrated on the right of Fig.2. The proposed Kinematic Chain Networks (KCN) is used for output embedding and the fead-forward part in the layer groups. And the two attention sub-layers of layer groups adopt the proposed Learned Local Attention (LLA) modules in the form of structured multi-head attention. We employ a stack of $N = 6$ layer groups in MoTrans.

4.2.1 Kinematic Chain Network

The Kinematic Chain Networks (KCN) are constructed on the basis of body structure, i.e., the 24 joint nodes with a tree topology. The networks consist of a stack of core blocks (FK-IK block) as illustrated in Fig.3. The core block

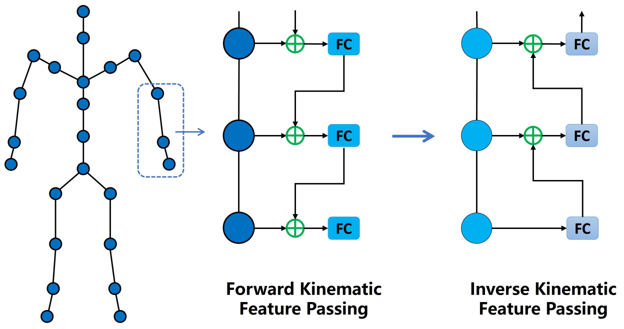
FC
FC
FC
Forward Kinematic Feature Passing
FC
FC
FC
Inverse Kinematic Feature Passing

Figure 3. **The FK-IK block of the Kinematic Chain Networks (KCN).** Features are embedded and passed along the kinematic chains forward and inversely with fusion operations. Note that the endpoints of feature passing are root and leaf nodes, and we just show a part of them for clearer view.

has two steps. It first performs a feature embedding on the root node and pass it to the neighbour child nodes along the kinematic chain. Receiving the features from the parent node, the child node performs feature fusion and then conducts feature embedding by linear projection (FC). This process is named forward kinematic feature passing, where fully-connected layers (FC) are utilized for feature embedding and the fusion adopts the addition operation. Then a similar feature embedding and passing process is performed from the leaf nodes to the root, completing the inverse kinematic feature passing. These procedures are similar to the operations of forward kinematics (FK) and inverse kinematics (IK), thus we name it as FK-IK block. The FK-IK block attempts to encode features into the motion controlling parameter space and to bring physics constraints into the model.

For the output embedding process, we use a KCN with one FK-IK block. And for fead-forward process in the core layers, we adopt a stack of 2 FK-IK blocks for KCN. All the KCNs have the same output feature size of 64 in implementation.

4.2.2 Learned Local Attention

To enable the networks to adapt to the locality property of motion states (joint parameters), we propose the Learned Local Attention (LLA) module. The generally used attention function proposed in the original transformer [19] is

$$Att(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{6}$$

where $Q \in \mathbb{R}^{l \times d_k}, K \in \mathbb{R}^{l \times d_k}, V \in \mathbb{R}^{l \times d_v}$ represent the matrix of queries, keys and values with a sequence length of l. Query vectors and key vectors have the same feature dimension d_k, and d_v is the feature dimension of value vectors. We use $\psi(\cdot)$ represents $softmax(\frac{\cdot}{\sqrt{d_k}})$ to focus on the effects of Q, K and V. Then the attention function for a certain query vector q_i is

$$Att(q_i, K, V) = \psi(q_i K^T)V \tag{7}$$

Fig.4 (left) provides an illustration of the mechanism of the kernel function. Here we introduce the kernel function ϕ that acts on the value vector sequence:

$$Att_\phi(q_i, K, V) = \psi(q_i \tilde{\phi}(K^T))\phi(V) \tag{8}$$

In this equation, $\tilde{\phi}$ is the index filter operation to select out the keys in the range of ϕ, because key and value vectors are pairwise and must have a same sequence length. The generally used attention in Eq.7 can also be regarded to have a uniform weighted kernel whose size is just the sequence length l. The kernel of the proposed LLA is centered at time step i with learnable parameters to weight the contribution of value vectors. The kernel size is set to 17 so as to involve the neighboring 8 beats before and after current beat in consideration of the dance principles.

The kernel parameters are shared across a sequence (the time dimension i), so there are 17 parameters for one LLA module to learn. In the training phase, the kernel functions are initialized with a Gaussian distribution $\mathcal{N} \sim (t_i, \sigma^2)$ where $\sigma = 5$ with the assumption that nearer motion state has stronger influence.

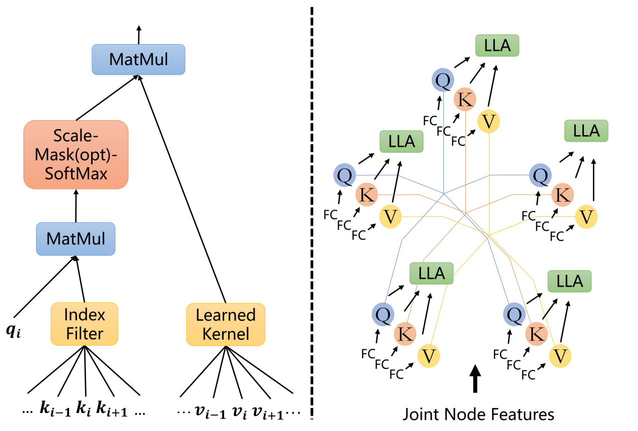

Figure 4. **(left) the Learned Local Attention (LLA) module.** We add the kernel functions to the value vectors and a index filter operation to select the corresponding key vectors. **(right) the structured multi-head attention.** We take the joint nodes as the heads in for attention. Joint node features are from KCN outputs in the self attention layers and both encoder features and KCN outputs in vanilla attention layers. Note that only 5 of the 24 joints are illustrated here for a clear view.

4.2.3 Structured Multi-Head Attention

Since features output by KCN are arranged in the joint nodes dimension, they are naturally compatible with the "multi-head" mechanism used in common transformers [19]. To maintain a unified network structure, the encoded features of the input music, which is used as V, K for the vanilla attention module (the upper LLA part in Fig.2), are processed by 24 paralleled fully-connected layers with the output size of 64 to obtain joint node features with the same structure.

With the nodes as "heads", we propose the structured multi-head attention process as illustrated on the right of Fig.4. For each node, 3 individual fully-connected layers embed the joint node features into Q, K and V vectors and then we perform 24 individual LLA modules on Q, K and V of the nodes. Note that no concatenation is needed because the features should maintain the body structure to be fed to the succeeding KCN. Thus this process actually saves computation compared to the typical multi-head attention that concatenates features from each head and use a big linear layer for feature embedding.

The structured multi-head attention briges the KCN and LLA modules, and the head here is the node in the body structure which has explicit physical significance. To some degree this design is more reasonable than the multi-heads in the original transformer.

5. Experiments

The experiments are performed on our *PhantomDance* dataset and the *AIST++* [11] dataset.

5.1. Implementation Details

Since most songs of the collected dances have the verse-chorus form, we divide the 300 dance animations into 1092 pieces according to their music structures. Among them 1000 pieces of music-dance pairs are used as input-supervision sequence pairs for model training, and the other 92 are split into test set.

The DanceNet3D is end-to-end trained using 4 TITAN Xp GPUs with a batch size of 8 on each GPU. We use Adam optimizer with betas (0.5, 0.999) and a learning rate of 0.0002. The learning rate drops to 2e-5, 2e-6 after $100k$, $200k$ steps. The model is trained with $300k$ steps for *AIST++* and $400k$ steps for *PhantomDance*.

5.2. Evaluation Metrics

Normalized Power Spectrum Simularity (NPSS) is a better evaluation metric for long-term motion synthesis compared to Mean Square Error (MSE). We just follow the official implementation in [5] and compute NPSS in the joint motion space $R^{T\times N\times 7}$ (4 for joint rotation represented as quaternion and 3 for joint position).

Frechet Distance (FD) is proposed by *AIST++* [11] that has two metrics, namely PFD for position and VFD for velocity. We also employ it to calculate the distribution distance of joints.

Position Variance (PVar) evaluates the diversity of the generated dance. In *AIST++*, a piece of music corresponds to more than one dance, but only one in *PhantomDance*. So we make a modification to the metric **PVar** in the experiments on *PhantomDance* where we compute it along different music pieces which have identical length.

Beat Consistency Score (BC) is a metric for motion-music correlation. We follow [11] to define kinematic beats as the local minima of the kinetic velocity. BC computes the average distance between every music beat and its nearest kinematic beat with the following equation:

$$BC = \frac{1}{n}\sum_{i=1}^{n}\exp\left(-\frac{\min_{\forall t_j^x \in B^x} ||t_i^x - t_j^y||^2}{2\sigma^2}\right) \qquad (9)$$

where $B^x = \{t^x\}, B^y = \{t^y\}$ are kinematic beats and music beats respectively and $\sigma = 3$. Note that BC has a very similar form to Beat Alignment Score (BA) proposed in [11], but they are different in essence. BA forces every kinematic beat to match a music beat, but in fact a dance usually has many small kinematic beats that occur between beats. Moreover a music synchronized dance just need to ensure the most music beats are accompanied with action emphasis (kinematic beats). So our proposed BC, which finds matched kinematic beat for each music beat, is a better metric.

Method	AIST++					PhantomDance				
	NPSS	PFD	VFD	PVar	BC	NPSS	PFD	VFD	PVar	BC
Li et al. [10]	16.31	5595.91	3.40	0.019	0.359	18.34	7944.78	5.84	0.014	0.175
Music2Dance [22]	14.74	2367.26	1.13	0.215	0.378	15.94	3147.89	3.74	0.267	0.223
AI Choreographer [11]	8.29	113.56	0.45	0.509	0.452	10.62	164.33	0.73	0.624	0.388
DanceNet3D	**5.81**	**81.34**	**0.33**	**0.744**	**0.872**	**7.43**	**104.30**	**0.51**	**0.931**	**0.835**

Table 1. **Dance generation evaluation on *AIST*++ and *PhantomDance* datasets.** Our method outperforms other baselines in term of dance quality, dance diversity and beat consistency. Especially, due to our two-stage prediction schema, our model has superior performance in term of BC indicating that our model can generate dances which better match the given music.

Metric	NPSS	PFD	VFD	PVar	BC
Baseline w/o Curve	15.11	678.49	2.97	0.343	0.327
Baseline	11.59	227.4	1.06	0.437	0.617
+GNN	10.34	146.43	0.69	0.647	0.692
+KCN	8.21	115.59	0.58	0.844	0.783
+Gauss	10.73	159.53	0.71	0.629	0.687
+LLA	8.93	127.14	0.64	0.759	0.764
+KCN +LLA	**7.43**	**104.30**	**0.51**	**0.931**	**0.835**

Table 2. **Ablation study about our proposed model (DanceNet3D) on the *PhantomDance* dataset.** **Baseline** refers to the typical transformer presented by [19]. **KCN** and **LLA** refers to the key components in **DanceNet3D** described in Sec.4. Here the component named **Gauss** denotes Gaussian kernel for local attention used as comparsion with LLA, and the **GNN** component denotes the typical graph neural network for our **KCN's** baseline.

5.3. Ablation Study

To evaluate the impact of the core ideas in DanceNet3D, we conduct the ablation study on our *PhantomDance* dataset. We use the typical transformer model [19] with 24 heads attention (same number as ours for fair comparison) as the baseline to compare with our MoTrans. The results are shown in Table 2.

Effectiveness of Curve Prediction: To evaluate the effectiveness of curve prediction, we perform experiments on the method that uses frame by frame generation for comparison, which is a common practice in previous works. The result is shown in the table as **Baseline w/o Curve**. The experimental result demonstrates the advantages of curve representation. Especially the improvement on the beat consistency score reveals the fluency property brings by motion curves.

Effectiveness of KCN: To study the impact of the KCN component, we introduce a comparison architecture which substitutes the FK-IK block with a 2-layer trivial graph neural network(**GNN**) [15]. This GNN fuses the joint features according to the adjacency matrix of undirected graph defined by the joint structure. The experimental result shows that GNN has better perforamce compared to the baseline, which proves that the representative ability of the model can be improved by introducing spatial information. While the proposed **KCN** surpasses the trivial GNN by a large margin. We claim that the proposed KCN, which involves spatial correlation and physics constraints into the model, brings significant improvements especially in generation quality (NPSS, PFD and VFD).

Effectiveness of LLA: Then we evaluate the effects of the **LLA** module. To verify the contribution of the learned kernel, we add experiments which applies a fixed Gaussian function with kernel size of 17 and standard variance of 5 for local attention. The results are shown as **+Gauss**, which proves the introducing of temporally local attention mechanism has improvements on the performance for the dance generation task. While the utilization of LLA module (the line of **+LLA**) demonstrates that a learned kernel can attain rather more gains. This is mainly owing to the rich represenitive ability brought by the LLA module, which enables the kernel to adapt to the specific feature distribution. We argue that concentrating on the adaptive local motion information makes the model generate more real transitions.

Finally, we perform the experiments on the complete model, which produces the optimal results in our study. It demonstrates that the kinematic controlling information introduced by KCN and the temporal locality property brought by LLA contribute in different aspects and altogether can improve the model greatly.

5.4. Comparison with Other Methods

We mainly compare our method with *AI Choreographer* [11] which to our knowledge obtains current state-of-the-art results for dance generation. Two other most related works, namely *Li et al.* [10] and *Music2Dance* [22] are also compared in this section. The comparison experiments are performed on both the *AIST*++ and our *PhantomDance* datasets.

The results are viewed in Table 1. Our method outperforms Li et al. [10] and Music2Dance [22] by a considerably large margin. And it also surpasses AI Choreographer [11] significantly in the quality related metrics (a 30% gain of NPSS, a 28% gain of PFD and a 27% gain of VFD), diversity metrics (a 46% gain of PVar) and beat consistency metrics (a 93% gain of BC).

The excellent results on the quality results are mainly owing to the appropriate architecture of KCN and LLA. While the generation diversity is ensured by the adversarial learning schema. Note that the upper bound of the metric

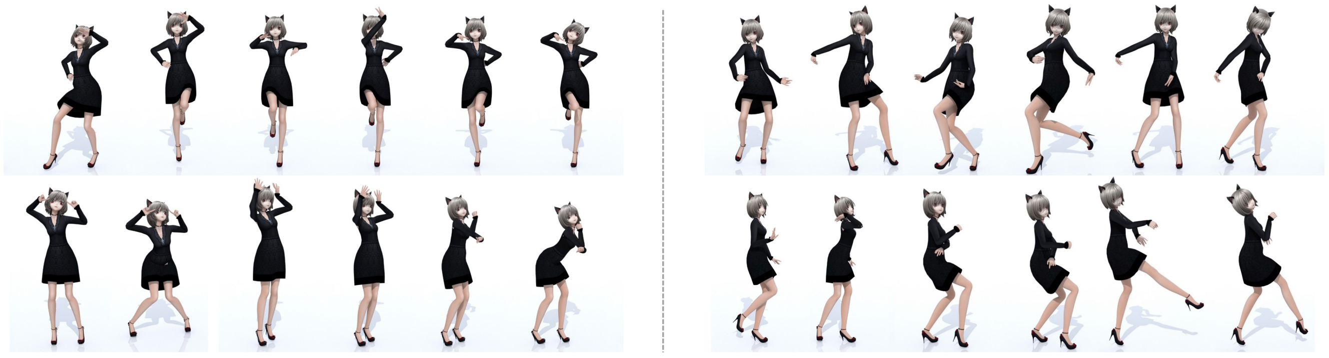

Figure 5. **Visualization of generated results of DanceNet3D.**

BC is 1, which means there exactly exists a kinematic beat at the time of each music beat. So our results have relatively high beat consistency score, which is due to the two-stage framework of DanceNet3D.

Since no metric emphasizes the fluency of the dance performance, the advantages of motion curve formulation can only be well revealed in qualitative results.

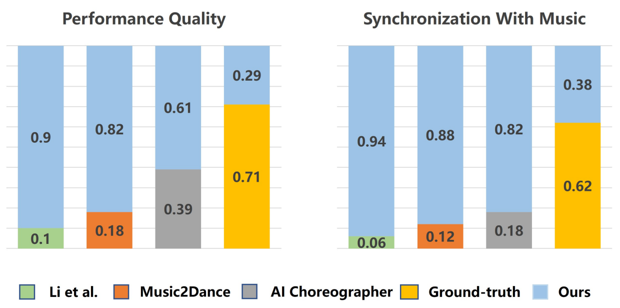

Figure 6. **Results of the user study on performance quality and synchronization with music.** We conduct a user study to ask participants to choose the better dances from pairwise comparisons. The criteria includes the performance quality and synchronization with music. The number denotes the percentage of preference on the comparison pairs.

5.5. Qualitative Results

Fig.5 provides a visualization of generated dances. Results in video form including comparison with other works can be found in the supplementary materials. Owing to curve representation, our results more fluent and graceful.

We also perform a user study to evaluate the authenticity and beat consistency of generated dances conditioned on music qualitatively. We first select 50 pieces of music from the validation set of PhantomDance. And then we collect the generated dances based on the music by conducting our method and three other works [11, 10, 22]. In addition, the ground-truth dances are also included. The user study is conducted using a pairwise comparison schema. For each of the 50 pieces musics, we provide 4 pairs in which our result occurs with result of other method or the ground truth. The $50 \times 4 = 200$ pairs are provided to the participants, and they are asked to make two choices for each pair: "Which dance is a better performance (more fluent, graceful and pleasing) regardless of music?" and "Which dance is more synchronized when played with the music?". There are 100 participants in the user study.

Fig.6 shows the user study results, where DanceNet3D outperforms the other methods on both criteria. Most of the participants think that our method generates better dances in performance quality compared with other works, owing to the curve representation to ensure the performance fluency. And even more participants hold the opinion that dances generated of our model are more synchronized with music. Compared to the ground truths (real dances), there are still 29% of users prefers our approach in quality and 38% in suitability for music. That is, to a great extent, a demonstration of the effectiveness of DanceNet3D.

6. Conclusion

In this work, we propose a new perspective to model the 3D dance generation task. Different from previous works that define the outputs as frames of poses, we formulate them as key poses and in-between motion curves. The curve representation makes the generated results more fluent and graceful. Based on the formulation we propose a dance generation framework, DanceNet3D with the novel MoTrans architecture consisting of the KCN and LLA module. KCN involves kinematic controlling information and LLA enables the attention function to focus on the nearby time steps that have strong influence on current time. And the structured multi-head attention naturally enriches the feature expressions and bridges the KCN and LLA. Due to the careful design of MoTrans, DanceNet3D yields high-quality results in experiments. Moreover, we propose PhantomDance-100 Dataset, the first 3D dance dataset proposed by professional animators. Future works based on the high-quality PhantomDance-100 can pay more attention to the artistic performance.

References

[1] Omid Alemi, Jules Françoise, and Philippe Pasquier. Groovenet: Real-time music-driven dance movement generation using artificial neural networks. *networks*, 8(17):26, 2017. 1, 2

[2] Shih-Pin Chao, Chih-Yi Chiu, Jui-Hsiang Chao, Shi-Nine Yang, and T-K Lin. Motion retrieval and its application to motion synthesis. In *24th International Conference on Distributed Computing Systems Workshops, 2004. Proceedings.*, pages 254–259. IEEE, 2004. 2

[3] Dariush Derakhshani. *Introducing Autodesk Maya 2013*. John Wiley & Sons, 2012. 3

[4] Daniel PW Ellis. Beat tracking by dynamic programming. *Journal of New Music Research*, 36(1):51–60, 2007. 1, 3, 4

[5] Anand Gopalakrishnan, Ankur Mali, Dan Kifer, Lee Giles, and Alexander G Ororbia. A neural temporal model for human motion prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12116–12125, 2019. 6

[6] 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. 2

[7] John Lasseter. Principles of traditional animation applied to 3d computer animation. In *Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, pages 35–44, 1987. 1

[8] Hsin Ying Lee, Xiaodong Yang, Ming Yu Liu, Ting Chun Wang, Yu Ding Lu, Ming Hsuan Yang, and Jan Kautz. Dancing to music. *Advances in Neural Information Processing Systems*, 32, 2019. 1, 2, 3

[9] Juheon Lee, Seohyun Kim, and Kyogu Lee. Listen to dance: Music-driven choreography generation using autoregressive encoder-decoder network. *arXiv preprint arXiv:1811.00818*, 2018. 1, 2

[10] Jiaman Li, Yihang Yin, Hang Chu, Yi Zhou, Tingwu Wang, Sanja Fidler, and Hao Li. Learning to generate diverse dance motions with transformer. *arXiv preprint arXiv:2008.08171*, 2020. 1, 3, 7, 8

[11] Ruilong Li, Shan Yang, David A Ross, and Angjoo Kanazawa. Learn to dance with aist++: Music conditioned 3d dance generation. *arXiv preprint arXiv:2101.08779*, 2021. 1, 2, 3, 6, 7, 8

[12] Beth Logan et al. Mel frequency cepstral coefficients for music modeling. In *Ismir*, volume 270, pages 1–11. Citeseer, 2000. 4

[13] Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J Black. Smpl: A skinned multi-person linear model. *ACM transactions on graphics (TOG)*, 34(6):1–16, 2015. 3

[14] 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. 2

[15] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE transactions on neural networks*, 20(1):61–80, 2008. 7

[16] Wataru Takano, Katsu Yamane, and Yoshihiko Nakamura. Retrieval and generation of human motions based on associative model between motion symbols and motion labels. *Journal of the Robotics Society of Japan*, 28(6):723–734, 2010. 2

[17] Taoran Tang, Jia Jia, and Hanyang Mao. Dance with melody: An lstm-autoencoder approach to music-oriented dance synthesis. In *Proceedings of the 26th ACM international conference on Multimedia*, pages 1598–1606, 2018. 1, 2

[18] Frank Thomas, Ollie Johnston, and Frank Thomas. *The illusion of life: Disney animation*. Hyperion New York, 1995. 1

[19] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *arXiv preprint arXiv:1706.03762*, 2017. 2, 3, 4, 5, 6, 7

[20] Richard Williams. *The animator's survival kit: a manual of methods, principles and formulas for classical, computer, games, stop motion and internet animators*. Macmillan, 2012. 1

[21] Sijie Yan, Zhizhong Li, Yuanjun Xiong, Huahan Yan, and Dahua Lin. Convolutional sequence generation for skeleton-based action synthesis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4394–4402, 2019. 2

[22] Wenlin Zhuang, Congyi Wang, Siyu Xia, Jinxiang Chai, and Yangang Wang. Music2dance: Music-driven dance generation using wavenet. *arXiv preprint arXiv:2002.03761*, 2020. 1, 2, 7, 8

[23] Dennis G Zill. *Advanced engineering mathematics*. Jones & Bartlett Publishers, 2020. 3