PC-Dance: Posture-controllable Music-driven Dance Synthesis

Jibin Gao
gaojb5@mail2.sysu.edu.cn
School of Computer Science and Engineering, Sun Yat-Sen University
Guangzhou, China

Junfu Pu
jevinpu@tencent.com
ARC Lab, Tencent PCG
Shenzhen, China

Honglun Zhang
honlanzhang@tencent.com
ARC Lab, Tencent PCG
Shenzhen, China

Ying Shan
yingshan@tencent.com
ARC Lab, Tencent PCG
Shenzhen, China

Wei-Shi Zheng*
zhwshi@mail.sysu.edu.cn
School of Computer Science and Engineering, Sun Yat-Sen University
Guangzhou, China

ABSTRACT

Music-driven dance synthesis is a task to generate high-quality dance according to the music given by the user, which has promising entertainment applications. However, most of the existing methods cannot provide an efficient and effective way for user intervention in dance generation, e.g., posture-controllable. In this work, we propose a powerful framework named **PC-Dance** to perform adaptive posture-controllable music-driven dance synthesis. Consisting of an music-to-dance alignment embedding network (M2D-Align) and a posture-controllable dance synthesis (PC-Syn), PC-Dance allows fine-grained control by input anchor poses efficiently without artist participation. Specifically, to relieve the cost of artist participation but ensure generating high-quality dance efficiently, a self-supervised rhythm alignment module is designed to further learn the music-to-dance alignment embedding. As for PC-Syn, we introduce an efficient scheme for adaptive motion graph construction (AMGC), which could improve the efficiency of graph-based optimization and preserve the diversity of motions. Since there is few related public dataset, we collect an *MMD-ARC* dataset for music-driven dance synthesis. The experimental results on *MMD-ARC* dataset demonstrate the effectiveness of our framework and the feasibility for dance synthesis with adaptive posture controlling.

CCS CONCEPTS

• **Computing methodologies → Procedural animation**.

KEYWORDS

Dance synthesis, Music-driven, Motion graph, Graph optimization, Datasets

*Corresponding author.

MM '22, October 10–14, 2022, Lisboa, Portugal

ACM ISBN 978-1-4503-9203-7/22/10…$15.00
https://doi.org/10.1145/3503161.3548099

ACM Reference Format:
Jibin Gao, Junfu Pu, Honglun Zhang, Ying Shan, and Wei-Shi Zheng. 2022. PC-Dance: Posture-controllable Music-driven Dance Synthesis. In *Proceedings of the 30th ACM International Conference on Multimedia (MM '22), October 10–14, 2022, Lisboa, Portugal.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3503161.3548099

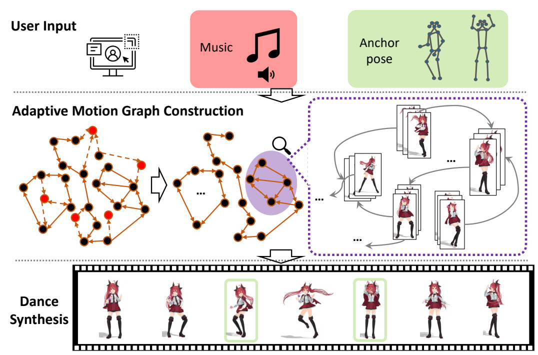

Figure 1: The dance synthesis by PC-Dance. With music and anchor poses input given by the user, the adaptive motion graph construction (AMGC) module will update an efficient motion graph for the given music. To provide fine-grained control over dance motions, the dance synthesis by PC-Dance is performed with the anchor poses.

1 INTRODUCTION

Dance is inseparable from our daily life of all time and it could be found everywhere, such as in film and games. Generally, dance is based on music, and dancers attempt to convey the content in music with limb movements according to their understanding of art. However, the conventional choreography of high-quality dance always requires the participation of artists for amazing presentation. It is common to cost lots of money for choreographing a high quality dance. Therefore, a music-driven dance synthesis model that can automatically generate corresponding high-quality dances from musical input would be desirable for its promising applications, such as daily entertainment and industrial production.

Recently, music-driven dance synthesis [1, 6, 18, 28, 31, 32, 35] has attracted more attention in decades. However, due to the hard interpretability of deep learning, most of existing generative model-based methods [10, 21, 22, 29] narrow their ability in short-term generation, and thus they are poor at generating high-quality dances of a long duration (tens of seconds), leading to either motion freezing or simple repetition. Therefore, many approaches [6, 16, 24, 25, 31] believe that the graph-based framework is the de facto standard solution to music-driven dance synthesis. However, existing works [6, 24, 25] have not yet exhaustively explored for the direct control of fine-grained dance movements (i.e., poses), leading to bad controllability for usage. Moreover, most of existing graph-based methods [16, 31] only consider the smoothing between motion nodes, ignoring the rules of choreography. Although some works [6, 24, 25] have considered this issue, they still failed to efficiently expand new motion data since the construction of the motion graph is heavily dependent on the participation of artists.

To address these problems and enable more flexible user intervention, we propose a new framework named PC-Dance to perform adaptive posture-controllable music-driven dance synthesis, which allows fine-grained control by input anchor poses efficiently without artist participation. Our system mainly consists of an music-to-dance alignment embedding network (M2D-Align) and the posture-controllable dance synthesis (PC-Syn). With the given music and anchor poses, PC-Syn could generate high-quality dance according to the user's expectations in a fine-grained manner with a posture constraint. Specifically, to ensure generate high-quality dance efficiently without artist participation, we construct a self-supervised rhythm alignment module by using the pseudo-label of the rhythm extracted from music to further learn the music-to-dance alignment embedding. Moreover, before the posture-controllable dance synthesis, an adaptive motion graph construction (AMGC) scheme is employed in our modelling with an indicator vector to improve the efficiency of graph-based optimization and preserve the diversity of motions.

Additionally, since there are few related datasets released, we collect a new dataset to evaluate dance synthesis, called the *MMD-ARC* dataset. The *MMD-ARC* dataset contains 213 entire dances with the length of 11.3 hours in total. For graph-based dance synthesis evaluation, there are 20,319 units of music-dance pairs in the *MMD-ARC* dataset. We have edited out the abnormal data from the original dance data. This dataset will be available.

In summary, our contributions are three-fold:

- An music-driven dance synthesis system (PC-Dance) is proposed, which allows fine-grained control by input anchor poses efficiently without artist participation;
- Self-supervised rhythm alignment is proposed for further learn the music-to-dance alignment embedding in a self supervision manner without artist participation;
- An efficient scheme for adaptive motion graph construction, which could improve the efficiency of graph-based optimization and preserve the diversity of motions.

2 RELATED WORK

2.1 Music-to-Dance Cross-Modal Mapping

Music-to-dance cross-modal mapping learning is a task to learn feature mappings for music and dance, which is a subtask of the music-driven dance synthesis. To better associate dance motions with music, existing works [6, 8, 9, 12, 14, 15, 20–22, 26, 30, 31] have mainly addressed this problem in two manners, namely the early traditional feature-based methods and the recent popular network learning-based methods. The early traditional feature-based methods [14, 20, 26, 30, 31] mainly compute the similarity between the music features (*e.g.*, onset, chroma) extracted from music signals and the dance motion features (*e.g.*, joint trajectories, hand and foot positions) extracted from dance motions, and find the mapping relationship by data matching. On the other hand, most of the network-based learning methods [9, 12, 22] perform mapping learning with score regression by learning deep features for music and dance motion. Specially, [6] constructs an embedding space with cross-modal mapping relationships between music and dance by considering the style and rhythm embeddings of music and dance separately, and then jointly learning the embedding spaces of them. By contrast, our method also adopts the modelling in embedding space, but different from [6],we emphasize learning the cross-modal mapping by means of directly embedding music and dance into the same space, and a self-supervised learning scheme is adopted for rhythm embedding to moderately learn the rhythm mapping relationship between music and dance. This is because there is no strong one-to-one correspondence between dance rhythm and music rhythm.

2.2 Graph-based Motion Synthesis

Graph-based motion synthesis [14, 23, 31, 33] aims to synthesize new 3D skeletal motions from an existing motion database. Inspired by [17], which generates new motions by cutting and pasting existing motion fragments from the database, early motion synthesis methods [2, 16, 19] introduce the concept of motion graphs, casting the motion synthesis into the formulation of finding a path in a pre-built motion graph. Some methods [14, 15, 31] optimize pathfinding by constructing music-to-dance mapping constraints. Several approaches [23, 27] model this problem using hidden Markov models (HMM), which could be solved by dynamic programming or beam search algorithms efficiently. Recently, many methods [3, 6] have applied motion synthesis to dance synthesis. However, these methods are still limited by the pre-built motion graphs, and the construction of different motion graphs will directly affect the efficacy of motion synthesis. Therefore, we introduce a strategy of adaptively updating the motion graph, by which the content of the motion graph can be adaptively optimized while ensuring the efficient construction of the motion graph.

3 METHODOLOGY

In this section, we will introduce our adaptive posture-controllable music-driven dance synthesis framework **PC-Dance** in details, as shown in Figure 2. In our framework, a music-to-dance alignment embedding network is construct to learn the cross-modal mapping from music to dance, which consists of a music-to-dance style alignment module and a self-supervised rhythm alignment module. Thereafter, given a piece of music and posture tokens, our PC-Dance generates a piece of dance with graph optimization on a motion graph constructed adaptively for the given music. To better understand the pipeline of the music-driven dance synthesis based

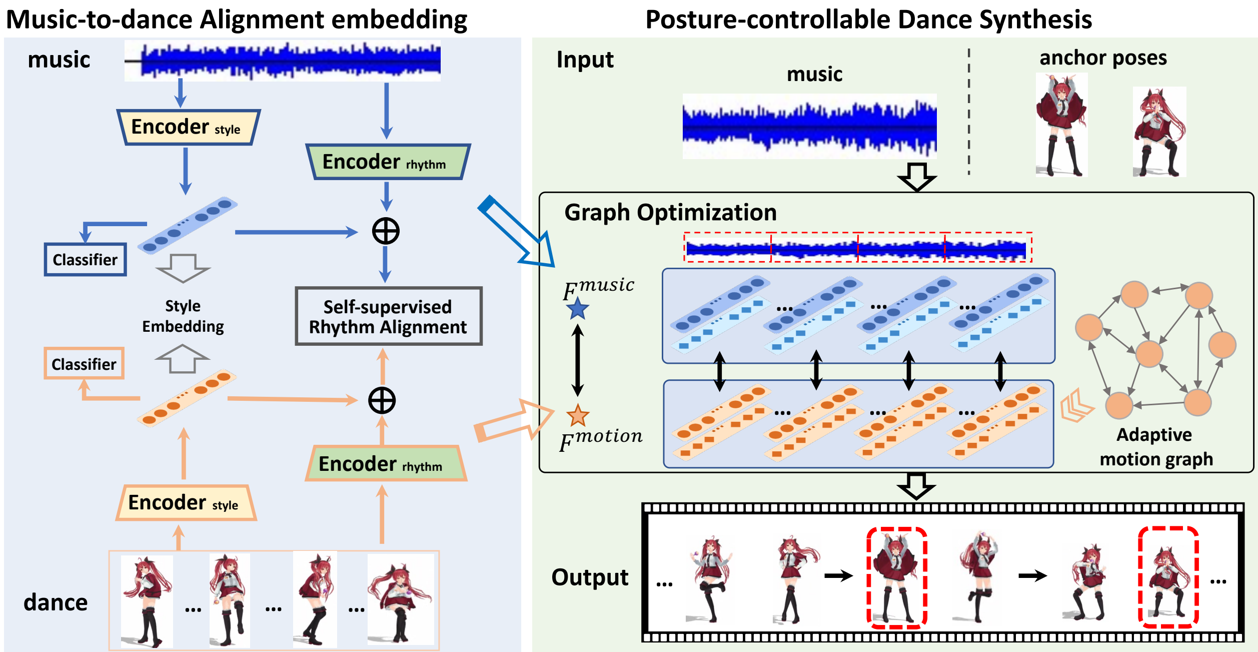

Figure 2: The overall of our proposed framework PC-Dance. In our system, firstly music-dance pairs are fed into corresponding encoders to learn music-to-dance alignment embedding. Then, with the learned embedding networks, music and anchor poses are input to perform graph optimization on an adaptive motion graph. Finally, according the optimal path obtained through the graph optimization, a controllable dance is generated, including the anchor poses.

on motion graph, we first make the problem formulation clear as below.

3.1 Problem formulation

Existing works for music-driven dance synthesis are mainly formulated in two steps, i.e., (1) learning the music-to-dance cross-modal mapping and (2) generating dance by inferring with given music. For our setting of graph-based dance synthesis, the key of dance generation is to perform graph optimization on a constructed motion graph to find an optimized path of motion nodes, and finally post-processing is utilized to accomplish the dance synthesis.

To learn the music-to-dance cross-modal mapping, high-quality pairs of music and dance are required for training an embedding network. We denote the raw data of a pair of music and dance as (m_0, d_0), and the embedded features could be represented as

$$(m_1, d_1) = (\mathcal{E}_m(m_0), \mathcal{E}_d(d_0)), \tag{1}$$

where $m_1, d_1 \in \mathbb{R}^k$ represent the embedded features of m_0 and d_0, respectively. $\mathcal{E}_m(\cdot)$ and $\mathcal{E}_d(\cdot)$ are the embedding networks to map music and dance into the same embedding space, respectively. For graph-based music-driven dance synthesis, a database (denoted as $\mathcal{B}$) containing lots of segments of dance is commanded for constructing a motion graph (denoted as $\mathcal{G}$ and $\mathcal{G} = (\mathbf{V}, \mathbf{E})$), where the nodes $\mathbf{V}$ represent the features of the segments of dance (mainly extracted by the learned $\mathcal{E}_d(\cdot)$). Since the motion graph is a directed graph, the edge $E_{i\to j}$, which represents the connection from node i to node j, could be written as

$$E_{i\to j} = \begin{cases} 1, & T_0(i,j) \le \delta; \\ 0, & \text{otherwise}; \end{cases} \tag{2}$$

where $T_0(i,j)$ denotes the transition gap from node i to node j and δ is a hyper-parameter threshold to limit the number of edges.

For graph optimization, a cost function $C(\cdot)$ is designed and employed to measure the cost of finding paths on the motion graph, which consists of many observational terms well-designed between the features of input music (extracted by the $\mathcal{E}_m(\cdot)$) and the features of motion node (mainly extracted by the learned $\mathcal{E}_d(\cdot)$). With a proper optimization method to minimize the cost function $C(\cdot)$, an optimized path p_o is determined. Finally, post-processing is used to generate the objective dance according to the optimized path p_o.

3.2 Music-to-Dance Alignment Embedding

As mentioned above, firstly we propose a music-to-dance alignment embedding (M2D-Align) network to learn the music-to-dance cross-modal mapping. As shown in Figure 2, our M2D-Align network is composed of two modules, a music-to-dance style alignment module to learn a embedding space for aligning the style latent feature of music and dance motion, and a self-supervised rhythm alignment module to guide the embedding space learning the motion rhythm with the supervision of music rhythm.

3.2.1 Music-to-Dance Style Alignment. With the difference of regional culture and the development for thousands of years, both music and dance have been differentiated into various sub-style genres. In general, the dance is choreographed according to the referred music and the style of dance naturally matches the that of music. Therefore, to generate a high-quality piece of dance, the style of dance should be consistent to that of the given music.

As discussed above, although there exist some differences between the music and dance genres, the style of music is dominant to the style of dance from the choreography perspective, and the style of dance usually could be described by the corresponding music. Thus, we develop a music-to-dance style alignment module to learn the unified style embedding of the pairs of music and dance. Music-dance unit pairs, which are with the length of t seconds, are utilized in our modelling. As in Equation (1), the style embedding could be expressed as

$$(m_s, d_s) = (\mathcal{E}_m^s(m_{s0}), \mathcal{E}_d^s(d_{s0})), \tag{3}$$

where $m_s \in \mathbb{R}^k$ and $d_s \in \mathbb{R}^k$ represent the style embedding of m_{s0} and d_{s0}, respectively. $\mathcal{E}_m^s(\cdot)$ and $\mathcal{E}_d^s(\cdot)$ are the style embedding encoders for music and dance, respectively.

To unify these two embedding spaces into the same space, the distance of music and dance embedding from the same unit pair should be close. Thus, we form a loss function to learn unified style embedding, which could be written as

$$\mathcal{L}_u = \sum_{i=1}^{N} \left\| m_s^i - d_s^i \right\|_2 + \mathcal{L}_{cls}, \tag{4}$$

where m_s^i and d_s^i denote the style embedding of the i-th music-dance pair. N is the number of sample pairs. $\mathcal{L}_{cls}$ is the classification loss of music and dance, and it is implemented by a cross entropy loss.

Moreover, to better learn discriminative embedding, a triplet loss is used to the style embedding learning. Therefore, we can write the final loss for M2D-Align as

$$\mathcal{L}_{sa} = \sum_{i=1}^{N'} \left(\left\| m_s^{a(i)} - d_s^{p(i)} \right\|_2^2 - \left\| m_s^{a(i)} - d_s^{n(i)} \right\|_2^2 + \epsilon \right) + \mathcal{L}_u, \tag{5}$$

where $\mathcal{L}_{sa}$ denotes the loss of our music-to-dance style alignment, and the features with superscripts $a(i), p(i), n(i)$ denote the anchor, positive and negative sample of the i-th triplet, respectively. N' is the number of triplets. ϵ is a hyper-parameter.

3.2.2 Self-Supervised Rhythm Alignment. It is common that the style consistency between music and dance requires the range and speed of dance motion fit the mood conveyed by the music. Moreover, with the development of dance, there are many conventional dance units for various kinds of music style. For example, given a piece of music whose style is "cute", the style consistency requires to use many cute dance units to generate a piece of cute dance (*i.e.*, $M^{cute} \rightarrow \{d^{cute(0)}, d^{cute(1)}, ...\} \rightarrow D^{cute}$). Thus, the style consistency aims to ensuring the mood between music and dance is matching, and it just constrains the content of dance (choosing dance units in the same style) according to the given music. However, an expressive dance performance could not be successful without performing the right movement at those key timings (*i.e.*, rhythm). Rhythm is important for a given music to choreograph a wonderful dance. Therefore, the rhythm consistency between music and dance is necessary. We attempt to learn the music-to-dance rhythm alignment, but the annotations of dance rhythm are difficult to acquire for its high labor-consumption, especially requiring artists' assistance. Therefore, to tackle this problem, we construct a self-supervised rhythm alignment module to learn the embedding space equipped with music-to-dance rhythm alignment in an self supervised manner.

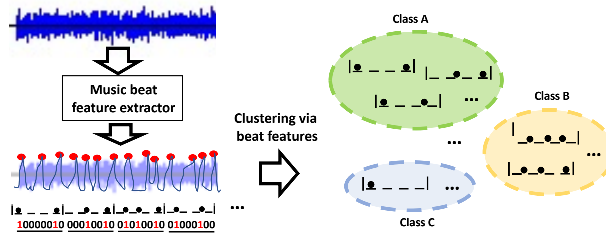

Figure 3: An example of getting the pseudo-labels of rhythm notes.

Specifically, as shown in Figure 3, we extract music beat feature through a beat detection model [4] as music rhythm feature for the pairs of dance and music unit, and it could be expressed as

$$R_m = \mathcal{A}(m_0). \tag{6}$$

Then, we use K-Means to group the rhythm features into P clusters, and the indices of the clusters Y_p are used as the rhythm annotations for corresponding music and dance units. In our self-supervised modelling, the Y_p is regarded as the pseudo-label of the rhythm. As in Equation (1), the rhythm embedding could be expressed as

$$(m_r, d_r) = (\mathcal{E}_m^r(m_{r0}), \mathcal{E}_d^r(d_{r0})), \tag{7}$$

where $m_r \in \mathbb{R}^k$ and $d_r \in \mathbb{R}^k$ represent the rhythm embedding of m_{r0} and d_{r0}, respectively. $\mathcal{E}_m^r(\cdot)$ and $\mathcal{E}_d^r(\cdot)$ are the rhythm embedding encoders for music and dance, respectively.

To jointly learn the rhythm embedding, we form the rhythm fusion features, written as

$$R_{md} = \mathcal{E}_{md}^r(m_{rs} \oplus d_{rs}), \tag{8}$$

where R_{md} represents the rhythm fusion features, and $\oplus$ is a concatenating operator. $\mathcal{E}_{md}^r(\cdot)$ is an encoder to fuse the rhythm embedding of music and dance. $m_{rs} = (m_r \oplus m_s)$ and $d_{rs} = (d_r \oplus d_s)$.

The loss function of our self-supervised rhythm alignment could be expressed as

$$\mathcal{L}_r = CrossEntropy(R_{md}, Y_p), \tag{9}$$

where Y_p is the pseudo-label of rhythm and $CrossEntropy(\cdot, \cdot)$ is a cross entropy loss function.

3.3 Posture-controllable Dance Synthesis

To make the dance synthesis more user-controllable, we construct a method to perform posture-controllable dance synthesis, where the input data does limit to music and users could feed posture anchors to somewhat control the content of generated dance. In our posture-controllable dance synthesis, apart from the input music, the posture anchors are optional to input as a control term. Then, to balance the variety of dance units and the searching space, we introduce an adaptive motion graph construction scheme to adaptively update the motion graph for the given music.

3.3.1 Inference Input. For the inference stage, the input music is required, and the posture anchor is optional. Given the inferring music M_{in}, it will be divided into a unit sequence $(m_{in}^0, m_{in}^1, ..., m_{in}^I)$ where every element represents a t-second music unit and I denotes the number of the elements. For posture anchor input, user could optionally give the pose sequence $(p_{in}^0, p_{in}^1, ..., p_{in}^J)$ where the element represents the pose data (*i.e.*, key-points format) and J is the number of poses.

3.3.2 Adaptive Motion Graph Construction. As mentioned in Section 3.1, a motion graph will be constructed for graph-based dance synthesis from the database $\mathcal{B}$. To improve the efficiency of graph-based optimization and decrease the space consumption, a adaptive motion graph construction scheme is employed in our modelling. Specifically, an indicator vector is constructed, and it is written as

$$\hat{\mathcal{I}} = \begin{cases} 1, & S_0(v^i, M_{in}) \geq \theta; \\ 0, & \text{otherwise}; \end{cases} \tag{10}$$

where $S_0(\cdot,\cdot)$ is a similarity function, v^i denotes the i-th node in the motion graph, and M_{in} is the given music for inference. θ is a hyper-parameter. Considering the stability of our motion graph, the final indicator vector $\mathcal{I}^a = \hat{\mathcal{I}} \| \mathcal{I}^0$ where $\mathcal{I}^0$ is the indicator of some pre-defined nodes which come from several pieces of general dance. $\|$ is the or operator.

With the $\mathcal{I}^a$, we can obtain the node set $\mathbf{V}^a$, and the formulation of E^a is similar to Equation 2. The transition gap function $T_0(i,j)$ is to calculate the distance from the last frame of node i to the first frame of node j.

3.3.3 Graph-based Optimization. To perform the graph-based optimization, we should design a cost function to guide our model to find an optimized path and generate a piece of high-quality dance. In the controllable setting, to ensure the input anchor poses to be synthesized into the generated dance sequence well, we constrain the posture anchor by calculating the similarity between poses and motion nodes, which could be expressed as

$$S_{ap}(i,j) = \max_l Cosine\_Sim(p_{in}^i, v_{p(l)}^j), \tag{11}$$

where $S_{ap}(i,j)$ denotes the cosine similarity between the anchor pose p_{in}^i and the node v^j, and $v_{p(l)}^j$ represents pose information of the l-th frame in the node j.

Moreover, for a piece of high-quality dance accompanied with given music, not only the consistency in style and rhythm between music and dance is important, but also should the smoothness and coordination of the motion be considered. To search a path of nodes on the motion graph with the same style and rhythm consistency as the given music, we close the distance of embedding features of music and dance motion with the learned style encoder $\mathcal{E}^s(\cdot)$ and rhythm encoder $\mathcal{E}^r(\cdot)$. It could be written as

$$C_{cons}(i) = \lambda_1 \mathcal{D}_s(m_s^i, d_s^i) + \lambda_2 \mathcal{D}_r(m_r^i, d_r^i), \tag{12}$$

where $\mathcal{D}_s(\cdot,\cdot)$ and $\mathcal{D}_r(\cdot,\cdot)$ are the functions to measure the distance of style embedding between the music and dance as well as that of rhythm embedding, respectively. In our experiments, $\mathcal{D}(\cdot,\cdot)$ is the sum of Euclidean distance and cosine distance.

To ensure the smoothness of the generated dance, we constrain that the adjacent nodes in the searched path should be with small gap between the node transition. It could be computed by

$$C_{tran}(i, i+1) = \mathcal{D}_t(v_o^i, v_o^{i+1}), \tag{13}$$

where v_o^i denotes the i-th node in the searched path and $\mathcal{D}_t(\cdot,\cdot)$ is the function to calculate the pose distance (Euclidean distance of joints in our setting) between the last frame of a node and the first frame of the next node.

The coordination of dance motions is relatively carefully designed according to the choreography rules. Typically, it takes into account the range of motion of the dance, the symmetry and repetition of the dance, and the stretch of the dancer's limbs. In our system, the range of motion of the dance is limited to a certain area, which could also encourage the system to search for symmetrical dance units to offset the overall range of motion. We write this constraint as

$$C_{rang}(i) = \left| \sum_{j=1}^{i} \mathcal{D}_p(v_o^j) \right|, \tag{14}$$

where v_o^j denotes the j-th node in the searched path and $\mathcal{D}_p(\cdot)$ represents the function to measure the range of motion of the node, *namely* the accumulation of position coordinate vectors.

In a short term of dance, little repetition of motion and great limb stretch are usually required to perform more amazing effect. For this purpose, we design a constraint condition to penalty the repeated nodes in a sliding window when searching the optimized path. Moreover, a liveness function of motion is utilized to guide our system to search a motion path with great limb stretch. Therefore, these two constraints could be expressed as

$$C_{repe}(i) = Count(\{v_o^{i-p+1}, v_o^{i-p+2}, ..., v_o^{i-1}\}, v_o^i) * \alpha, i \geq p, \tag{15}$$

$$C_{stre}(i) = \frac{\beta}{\mathcal{D}_h(v_o^i)}, \tag{16}$$

where v_o^i denotes the i-th node in the searched path and $Count(Q, v)$ denotes the function to count the number of occurrences of node v in sequence Q. $\mathcal{D}_h(\cdot)$ represents the liveness function to calculate the motion difference among joints of dancer's skeleton. α and β are hyper-parameters.

Finally, the overall cost function, which guides our model to find an optimized path, could be written as

$$\begin{aligned} C_{over}(i) = & \lambda_3 * C_{over}(i-1) + \lambda_4 * C_{tran}(i-1, i) \\ & + \lambda_5 * C_{cons}(i) + \lambda_6 * C_{rang}(i) + \lambda_7 * C_{repe}(i) \\ & + \lambda_8 * C_{stre}(i) + \lambda_9 * S_{ap}(P_{in}, i), \end{aligned} \tag{17}$$

where λ_i $(i = 3, 4, ..., 9)$ are hyper-parameters and P_{in} is the input anchor poses. When no anchor poses input, λ_9 is set to zero. The optimal dance motion path can be efficiently synthesized using a dynamic programming algorithm [11].

4 THE MMD-ARC DATASET

There is few dataset available for our model evaluation. Thus, we collect a new dataset, called the MikuMikuDance Archive (MMD-ARC) Dataset, for our following experimental study. The details of the *MMD-ARC* dataset could be found in Table 1.

- **Dataset Construction Details.** We downloaded 343 complete dance motion data with background music from the MikuMikuDance community. After manually filtering out the noise samples,

Table 1: Details of the *MMD-ARC Dataset*

	music-dance pair
#Samples	213
Total hour length	11.3
#Label	4
#Unit	20319
#Training set	16264
#Testing set	4055

Table 2: The results of our model compared with other methods on the *MMD-ARC* dataset.

Method	FID (↓)	Style Acc. (↑)	Diversity (↑)
Real Dance	1.7	-	81.5
DanceNet [35]	93.8	61.7%	57.7
Learning2dance [10]	25.2	65.2%	67.8
AI Choreographer* [6]	10.1	69.1%	74.2
Ours	**3.8**	**74.1%**	**76.3**

the remaining 213 complete dance motion samples were used as the original data of the *MMD-ARC* dataset, with a total time of 11.3 hours. The four different languages of music in the dataset are used as the labels. We randomly split the dataset into 170 complete training samples and 43 complete testing samples with a ratio of 4:1. To expand the data and facilitate the construction of motion graphs, we divided the complete dance samples into music-dance unit pairs at equal intervals with a duration of 2 seconds, and finally obtained 20,319 unit pairs, including 16,264 training unit pairs and 4,055 pair of test units. Samples of the *MMD-ARC* dataset can be found in our supplementary materials.

5 EXPERIMENT

In this section, we will introduce our experimental setting and the post-processing of the generated dance for better presentation. Then, quantitative and qualitative comparisons are conducted with several state-of-the-art methods. We also analyze the contributions of main components of our system through ablation study. Finally, the visualization of posture-controllable evaluation demonstrates the promising controllability of our proposal.

5.1 Experimental Setting

The music units and dance units are uniformly formatted to 2 seconds. We set δ as 20, θ as 0.95, and ϵ as 0.5. The hyper-parameters in the cost function is set to: $\alpha, \beta, \lambda_1, \lambda_2, ..., \lambda_9$ = 100, 100, 1, 10, 1, 2, 1, 10, 1, 10, 100. These hyper-parameters could be adjusted according to the system focus. Our model is implemented in PyTorch and the networks are trained on a P40 GPU server. The posture-controllable dance synthesis is tested on a desktop with a 3.80GHz i7-10700K CPU, 32GB RAM and a GeForce RTX 3080 GPU. The dimension k of embedding features is 32. The number of clusters grouping the rhythm features by K-means is 13. We adopt the music tagging network in [7] as our backbone for the style embedding of music and dance. The backbone of the rhythm embedding is adopted by the choreomusical rhythm embedding network in [6]. We train our music-to-dance alignment embedding network by using Adam optimizer with a batch size of 128 and a learning rate of 0.001 for 1000 epochs. Refer to *Supplementary Materials* for more details.

5.2 Data Pre-processing and Post-processing

As in [6], for style embedding alignment, we downsample music data to 16kHz and compute the log-amplitude mel spectrograms with 96 mel bins and 160 hop size to learn music style embedding. For dance motion input, the 6D representation [34] for the 3D rotations data of 22 joints is utilized to learn the dance style embedding. Thus, for 2-second unit pairs (60 key frames in 30 fps dance motions), the shapes of m_{s0} and d_{s0} are $[1, 96, 200]$ and $[6, 22, 60]$. For rhythm embedding alignment, we extract the spectral onset strength curve [5] and RMS energy curve for music input and thus the shape of m_{r0} is $[2, 200]$; for dance motions we compute the motion kinematic curve, two hand trajectory curvature curves and two foot contact curves, and thus the shape of d_{r0} is $[5, 60]$.

For the optimal node path of generated dance, there always exist gaps in the transfer between nodes since the nodes may come from different dance segments. Therefore, to generate smooth dance, data post-processing to smooth the gaps in the node transfer is necessary. Inspired by [2], we design a smoothing function to eliminate the discontinuities at node transfer. Refer to the supplementary materials for details.

5.3 Evaluation Metric

To compare our method with existing works [6, 10, 35], we utilize Fréchet inception distance (FID) score [13] to measure the distance between the distribution of generated dances and that of the real ones. Following [6, 18], a motion auto-encoder is trained on our *MMD-ARC* dataset and used for feature extraction. Similarly, to evaluate the diversity of generated dances, we calculate the average feature distance between generated dances for different music inputs, as used in [6, 18]. Moreover, style accuracy is computed to better present the performance of style embedding. Refer to our supplementary materials for more details of these metrics.

5.4 Comparisons

Since the *MMD-ARC* Dataset is brand new for existing methods, we have tried our best to evaluate many advanced methods whose codes are available on *MMD-ARC* and re-implement the state-of-the-art work, AI Choreographer [6] for comparisons. Finally, two deep generative models are considered, DanceNet [35] and Learning2Dance [10]. They are the most recently proposed models with the codes available. Besides, without the rhythm signature annotated by artists, the current state-of-the-art graph-based method, AI Choreographer [6], could not be completely implemented and evaluated on *MMD-ARC* dataset. To overcome this dilemma, we replaced the requirement of rhythm signature in AI Choreographer [6] with our self-supervised rhythm alignment, and called the modified version of AI Choreographer [6] as "AI Choreographer*".

As shown in Table 2, our method achieves the best performance on all three evaluation metrics, and outperforms three baselines (*i.e.*, DanceNet [35], Learning2Dance [10] and the modified version of

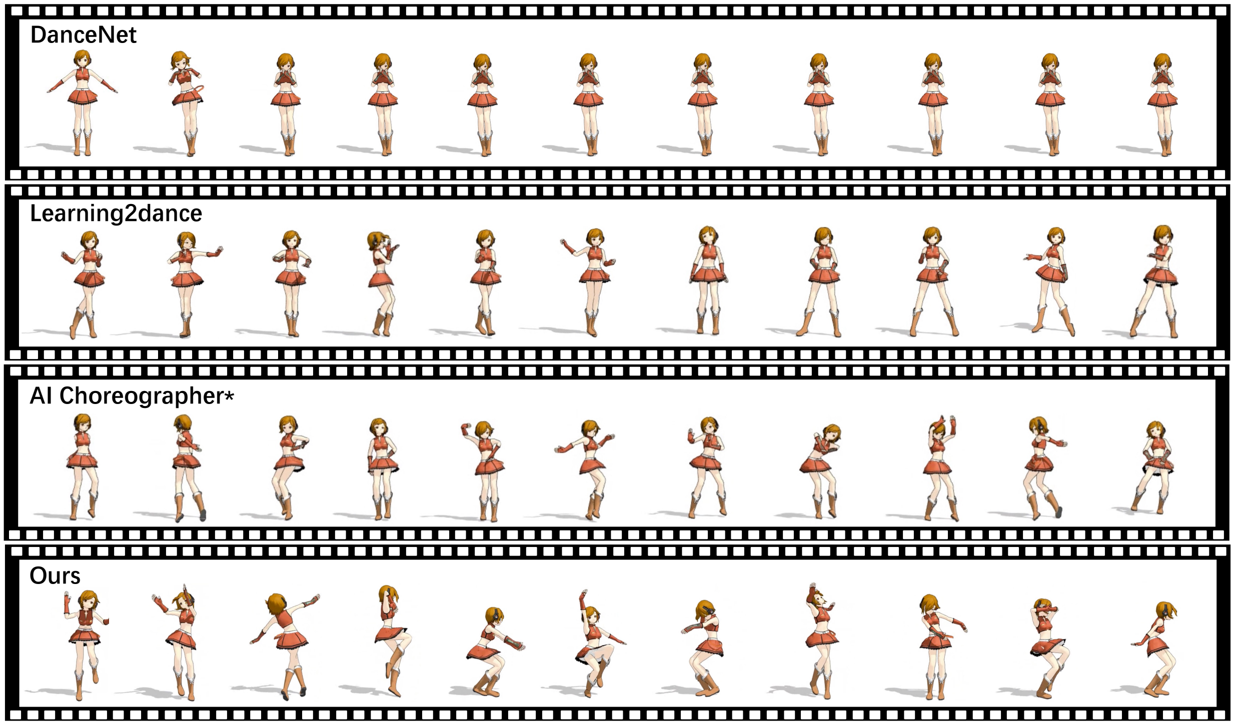

Figure 4: Dance motions generated by four different methods for an English song. 'DanceNet' [35] and 'Learning2dance' [10] are two competitive deep generative models, and 'AI Choreographer*' is the modified version of the state-of-the-art graph-based method, AI Choreographer [6], without rhythm signature.

Table 3: The results of our ablation study on the *MMD-ARC* dataset.

Method	FID (↓)	Style Accuracy (↑)	Diversity (↑)
Full model	3.8	74.1%	76.3
w/o Triplet Loss	4.2	69.8%	74.6
w/o SSRA	4.8	68.7%	74.2

Table 4: Dance synthesis time required for different sizes of motion graph on the *MMD-ARC* dataset.

Edge Count	Music Duration	Synthesis time
100k	60 s	1.54 s
100k	120 s	3.20 s
200k	60 s	3.31 s
200k	120 s	6.47 s

AI Choreographer [6] as "AI Choreographer*") with improvements of more than 5% by comparison on style accuracy. The smallest FID score of our proposal reveals that the generated dances by our system are in a more similar distribution to the real dances, while the greatest score of 'Diversity' implies that our system could generate various dances for difference music inputs better. Moreover, the results indicate that the performance of our method is closest to the real dance on each evaluation metric, which demonstrates the effectiveness of our method.

5.5 Model Analysis

In this section, we first make the qualitative comparison by visualization the generated dance motions synthesized by the baselines and our method, and analysis the difference among them. Then, an ablation study is conducted to explore the contributions of each main component in our framework. To better evaluate the performance of our proposal, a user study has been designed for the generated dances synthesized by the baselines and our method. Moreover, to evaluate the efficiency of our system, we also test the dance synthesis time-consuming of our model on motion graphs with different sizes and music input with different lengths. Finally, posture-controllable testing is performed to check the ability of our system to directly control the dance generation with posture input.

5.5.1 Generated Dance Visualization. To view the quality of the generated dances, we visualized the dance motions generated by four different methods for an English song, as shown in Figure 4. From Figure 4, we find that dances generated by the deep generative models (*i.e.*, DanceNet [35] and Learning2dance [10]) only has left-and-right swings at the elbow and knee joints, with little movement of the entire body, and the dance movements are relatively stiff. By contrast, dance motions generated by the graph-based methods (*i.e.*, the modified AI Choreographer and our proposal) are more natural, which can express the wild of that English song. This partly benefits from the graph-based scheme, where the motions are synthesized from real dance motions in the database. Compared with the "AI choreographer*", our method could find the optimal path where the

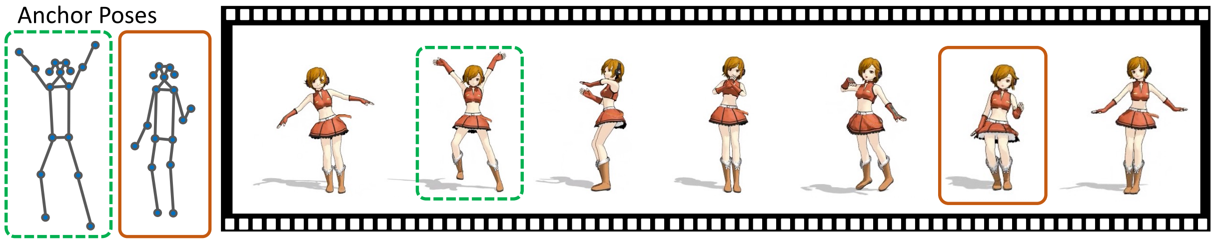

Figure 5: Controllable dance motions generated with anchor poses input.

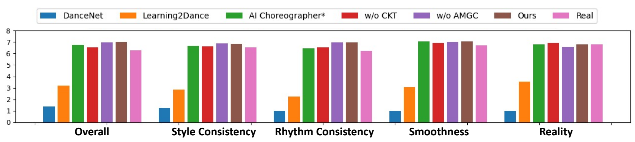

Figure 6: User study results.

node motions are more stretched to generate more amazing dance performance. Watch more demos in our supplementary materials.

5.5.2 Ablation Study. To investigate the contributions of each main component in our framework, we conducted an ablation study on our model by removing one of the components from our full model. As shown in Table 3, when removing the triplet loss in our music-to-dance alignment embedding, the model performance decreased more than 5%, which demonstrates the contrastive learning scheme is useful to better align the embedding spaces of music and dance motion. Moreover, when we removed the self-supervised rhythm alignment module (SSRA), a drop of 7% was obtained, which implies that the rhythm information could help the alignment between music and dance. Since the impact of each constraint in our cost function is difficult to present by figures, we evaluate them through video presentation in our supplementary materials.

5.5.3 Posture-Controllable Testing. To verify the controllability of our method, we input anchor poses to controllably generate dance motions, as shown in Figure 5. It can be observed that our proposed method can smoothly generate dance motions according to the postures input by the user, which includes preset anchor poses. The results indicate that the controllable setting can improve the dance generation diversity and user interactivity of our system.

5.5.4 Dance Synthesis Time. To ensure that our system can be employed in real production, the efficiency of dance synthesis is an important consideration. Therefore, we test the dance synthesis time-consuming of our model on motion graphs with different sizes and music input with different lengths. As shown in Table 4, for the motion graph with 100k edges on the *MMD-ARC* dataset, it took 1.54 seconds for our system to perform dance synthesis for a 60-second music input. Roughly, the time spent grows linearly with the number of edges and the duration of the music on the database from *MMD-ARC* dataset. The results demonstrate the good efficiency of our system. Moreover, we tested the impact of the hyper-parameter θ in our AMGC on a 60-second song, and the results is presented in our supplementary materials, which demonstrate the effectiveness of our AMGC. Refer to *Supplementary Materials* for more details.

5.5.5 User Study. To eliminate the evaluation bias caused by subjective differences as much as possible, we designed a user study about generated dances from seven sources, including the real dance and those generated by DanceNet [35], Learning2dance [10], the modified AI Choreographer [6], our proposal, our model without the range constraint ('w/o CKT', Coordination-Keeping Tools) and that without adaptive motion graph construction ('w/o AMGC'). In this study, we asked the surveyed users to score each of given dances in $[1, 10]$ on five evaluation items, including the overall, style consistency, rhythm consistency, smoothness and reality. 16 participants were invited in our study, and 64 score data for each evaluation item were obtained to be averaged as the final score of each method. As shown in Figure 6, we find that the dance motions generated by our dance synthesis system got the comparable scores at all evaluation items compared with the real dances, which demonstrates the great capability of our system to high-quality dance synthesis.

6 CONCLUSION

In this paper, a new framework, **PC-Dance**, is designed for adaptive posture-controllable music-driven dance synthesis, which allows fine-grained control by input anchor poses efficiently without artist participation. With the given music and anchor poses, our system could generate high-quality dance according to the user's expectations. Specifically, to ensure generate high-quality dance efficiently without artist participation, we construct a self-supervised rhythm alignment module to further learn the music-to-dance alignment embedding, and introduce an efficient scheme for adaptive motion graph construction, which could improve the efficiency of graph-based optimization and preserve the diversity of dance motions. The experimental results on the new collected dataset *MMD-ARC* demonstrate the effectiveness of our framework and the feasibility for dance synthesis with adaptive posture controlling.

7 ACKNOWLEDGEMENT

We thank Jia-Chang Feng for his constructive advice. This work was supported partially by the NSFC (U21A20471, U1911401, U1811461), Guangdong NSF Project (No. 2020B1515120085, 2018B030312002), Guangzhou Research Project (201902010037), and the Key-Area Research and Development Program of Guangzhou (202007030004).

REFERENCES

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

[2] Okan Arikan and David A Forsyth. 2002. Interactive motion generation from examples. *ACM Transactions on Graphics (TOG)* 21, 3 (2002), 483–490.

[3] Alexander Berman and Valencia James. 2015. Kinetic imaginations: Exploring the possibilities of combining AI and dance. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*.

[4] Sebastian Böck, Filip Korzeniowski, Jan Schlüter, Florian Krebs, and Gerhard Widmer. 2016. madmom: a new Python Audio and Music Signal Processing Library. In *Proceedings of the 24th ACM International Conference on Multimedia*. Amsterdam, The Netherlands, 1174–1178. https://doi.org/10.1145/2964284.2973795

[5] Sebastian Böck and Gerhard Widmer. 2013. Maximum filter vibrato suppression for onset detection. In *Proc. of the 16th Int. Conf. on Digital Audio Effects (DAFx). Maynooth, Ireland (Sept 2013)*, Vol. 7.

[6] Kang Chen, Zhipeng Tan, Jin Lei, Song-Hai Zhang, Yuan-Chen Guo, Weidong Zhang, and Shi-Min Hu. 2021. Choreomaster: choreography-oriented music-driven dance synthesis. *ACM Transactions on Graphics (TOG)* 40, 4 (2021), 1–13.

[7] Keunwoo Choi, György Fazekas, Mark Sandler, and Kyunghyun Cho. 2017. Convolutional recurrent neural networks for music classification. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2392–2396.

[8] Abe Davis and Maneesh Agrawala. 2018. Visual rhythm and beat. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 2532–2535.

[9] Rukun Fan, Songhua Xu, and Weidong Geng. 2011. Example-based automatic music-driven conventional dance motion synthesis. *IEEE transactions on visualization and computer graphics* 18, 3 (2011), 501–515.

[10] J. P. Ferreira, T. M. Coutinho, T. L. Gomes, J. F. Neto, R. Azevedo, R. Martins, and E. R. Nascimento. 2021. Learning to dance: A graph convolutional adversarial network to generate realistic dance motions from audio. *Computers & Graphics* 94 (2021), 11 – 21. https://doi.org/10.1016/j.cag.2020.09.009

[11] G David Forney. 1973. The viterbi algorithm. *Proc. IEEE* 61, 3 (1973), 268–278.

[12] Satoru Fukayama and Masataka Goto. 2015. Music content driven automated choreography with beat-wise motion connectivity constraints. *Proceedings of SMC* (2015), 177–183.

[13] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. 2017. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems* 30 (2017).

[14] Jae Woo Kim, Hesham Fouad, and James K Hahn. 2006. Making Them Dance.. In *AAAI Fall Symposium: Aurally Informed Performance*, Vol. 2.

[15] Tae-hoon Kim, Sang Il Park, and Sung Yong Shin. 2003. Rhythmic-motion synthesis based on motion-beat analysis. *ACM Transactions on Graphics (TOG)* 22, 3 (2003), 392–401.

[16] Lucas Kovar, Michael Gleicher, and Frédéric Pighin. 2008. Motion graphs. In *ACM SIGGRAPH 2008 classes*. 1–10.

[17] Alexis Lamouret and Michiel van de Panne. 1996. Motion synthesis by example. In *Computer Animation and Simulation'96*. Springer, 199–212.

[18] Hsin-Ying Lee, Xiaodong Yang, Ming-Yu Liu, Ting-Chun Wang, Yu-Ding Lu, Ming-Hsuan Yang, and Jan Kautz. 2019. Dancing to music. *arXiv preprint arXiv:1911.02001* (2019).

[19] Jehee Lee, Jinxiang Chai, Paul SA Reitsma, Jessica K Hodgins, and Nancy S Pollard. 2002. Interactive control of avatars animated with human motion data. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*. 491–500.

[20] Minho Lee, Kyogu Lee, and Jaeheung Park. 2013. Music similarity-based approach to generating dance motion sequence. *Multimedia tools and applications* 62, 3 (2013), 895–912.

[21] Buyu Li, Yongchi Zhao, Zhelun Shi, and Lu Sheng. 2021. DanceFormer: Music Conditioned 3D Dance Generation with Parametric Motion Transformer. *arXiv preprint arXiv:2103.10206* (2021).

[22] Ruilong Li, Shan Yang, David A Ross, and Angjoo Kanazawa. 2021. Learn to Dance with AIST++: Music Conditioned 3D Dance Generation. *arXiv preprint arXiv:2101.08779* (2021).

[23] Adriano Manfrè, Ignazio Infantino, Filippo Vella, and Salvatore Gaglio. 2016. An automatic system for humanoid dance creation. *Biologically Inspired Cognitive Architectures* 15 (2016), 1–9.

[24] Paul H Mason. 2012. Music, dance and the total art work: choreomusicology in theory and practice. *Research in dance education* 13, 1 (2012), 5–24.

[25] Mohd Anis Md Nor and Kendra Stepputat. 2016. Sounding the Dance, Moving the Music. *Choreomusicological Perspectives on Maritime Southeast Asian Performing Arts. Routledge* (2016).

[26] Ferda Ofli, Yasemin Demir, Yücel Yemez, Engin Erzin, A Murat Tekalp, Koray Balcı, İdil Kızoğlu, Lale Akarun, Cristian Canton-Ferrer, Joëlle Tilmanne, et al. 2008. An audio-driven dancing avatar. *Journal on Multimodal User Interfaces* 2, 2 (2008), 93–103.

[27] Ferda Ofli, Engin Erzin, Yücel Yemez, and A Murat Tekalp. 2011. Learn2dance: Learning statistical music-to-dance mappings for choreography synthesis. *IEEE Transactions on Multimedia* 14, 3 (2011), 747–759.

[28] Mathis Petrovich, Michael J Black, and Gül Varol. 2021. Action-Conditioned 3D Human Motion Synthesis with Transformer VAE. *arXiv preprint arXiv:2104.05670* (2021).

[29] Xuanchi Ren, Haoran Li, Zijian Huang, and Qifeng Chen. 2020. Self-supervised dance video synthesis conditioned on music. In *Proceedings of the 28th ACM International Conference on Multimedia*. 46–54.

[30] Takaaki Shiratori and Katsushi Ikeuchi. 2008. Synthesis of dance performance based on analyses of human motion and music. *Information and Media Technologies* 3, 4 (2008), 834–847.

[31] Takaaki Shiratori, Atsushi Nakazawa, and Katsushi Ikeuchi. 2006. Dancing-to-music character animation. In *Computer Graphics Forum*, Vol. 25. Wiley Online Library, 449–458.

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

[33] Yanzhe Yang, Jimei Yang, and Jessica Hodgins. 2020. Statistics-based Motion Synthesis for Social Conversations. In *Computer Graphics Forum*, Vol. 39. Wiley Online Library, 201–212.

[34] Yi Zhou, Connelly Barnes, Jingwan Lu, Jimei Yang, and Hao Li. 2019. On the continuity of rotation representations in neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 5745–5753.

[35] Wenlin Zhuang, Congyi Wang, Siyu Xia, Jinxiang Chai, and Yangang Wang. 2020. Music2dance: Dancenet for music-driven dance generation. *arXiv preprint arXiv:2002.03761* (2020).