ChoreoMaster: Choreography-Oriented Music-Driven Dance Synthesis

KANG CHEN, NetEase Games AI LAB, China
ZHIPENG TAN, NetEase Games AI LAB, China
JIN LEI, NetEase Games AI LAB, China
SONG-HAI ZHANG*, Tsinghua University, China
YUAN-CHEN GUO, Tsinghua University, China
WEIDONG ZHANG, NetEase Games AI LAB, China
SHI-MIN HU, Tsinghua University, China

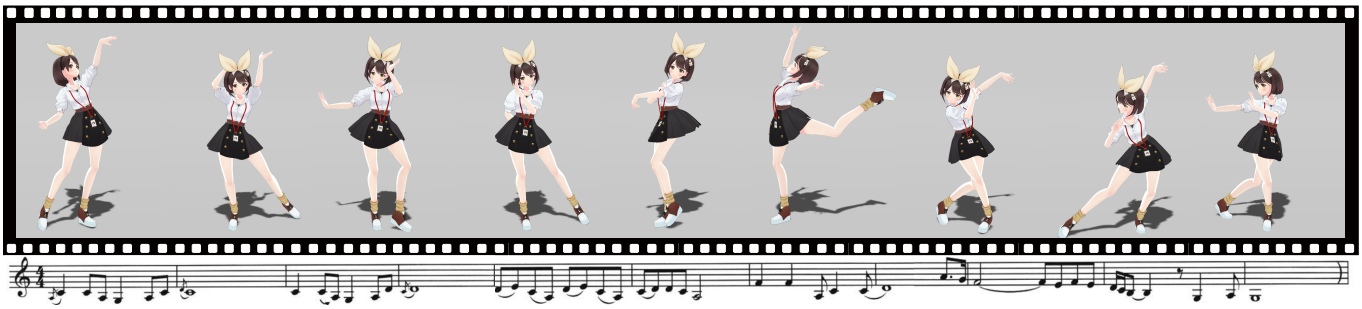

Fig. 1. Dance results automatically synthesised by ChoreoMaster for a traditional Chinese song.

Despite strong demand in the game and film industry, automatically synthesizing high-quality dance motions remains a challenging task. In this paper, we present ChoreoMaster, a production-ready music-driven dance motion synthesis system. Given a piece of music, ChoreoMaster can automatically generate a high-quality dance motion sequence to accompany the input music in terms of style, rhythm and structure. To achieve this goal, we introduce a novel choreography-oriented choreomusical embedding framework, which successfully constructs a unified choreomusical embedding space for both style and rhythm relationships between music and dance phrases. The learned choreomusical embedding is then incorporated into a novel choreography-oriented graph-based motion synthesis framework, which can robustly and efficiently generate high-quality dance motions following various choreographic rules. Moreover, as a production-ready system, ChoreoMaster is sufficiently controllable and comprehensive for users to produce desired results. Experimental results demonstrate that dance motions generated by ChoreoMaster are accepted by professional artists.

*corresponding author

Authors' addresses: Kang Chen, NetEase Games AI LAB, Hangzhou, China, chenkangnobel@gmail.com; Zhipeng Tan, NetEase Games AI LAB, Hangzhou, China, tanzhipeng@corp.netease.com; Jin Lei, NetEase Games AI LAB, Hangzhou, China, leijin@corp.netease.com; Song-Hai Zhang, Tsinghua University, Beijing, China, shz@tsinghua.edu.cn; Yuan-Chen Guo, Tsinghua University, Beijing, China, guoyc19@mails.tsinghua.edu.cn; Weidong Zhang, NetEase Games AI LAB, Hangzhou, China, zhangweidong02@corp.netease.com; Shi-Min Hu, Tsinghua University, Beijing, China, shimin@tsinghua.edu.cn.

0730-0301/2021/8-ART1 $15.00
https://doi.org/10.1145/3450626.3459932

CCS Concepts: • **Computing methodologies** → **Procedural animation**; **Learning latent representations**.

Additional Key Words and Phrases: Choreography System, Dance Motion Synthesis, Cross-Modality Learning

ACM Reference Format:
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 Trans. Graph.* 40, 4, Article 1 (August 2021), 13 pages. https://doi.org/10.1145/3450626.3459932

1 INTRODUCTION

Dance is ages old. Dance movements are common elements in films and video games, and so there are rich demands for high-quality 3D dance animation assets in these industries. However, the production of dance animations is extremely costly and inefficient, and typically involves an artist to choreograph a dance (normally based on given music), an experienced dancer to perform the dance, and a motion capture team to record the dancer's movements. The whole task requires skill and expertise in choreography and dancing, as well as a tedious process of cleaning and repairing the captured motions. One minute of original dance animation can easily cost thousands of (US) dollars. Therefore, it would be of great benefit if this task could be carried out automatically by a production-ready tool.

In the context of music-driven dance motion synthesis, for a tool to be considered production-ready, it should have two basic characteristics. Firstly, it should robustly and stably produce high-quality dance motions satisfying basic choreographic rules, and preferably provide multiple suitable alternatives for each input. Secondly, users should have complete control over the synthesis process, being able to specify desirable and undesirable actions, to adjust the diversity

and novelty of generated motions, and so on. Unfortunately, none of the existing research methods for dance motion synthesis have been successfully transformed into practical production tools, as currently, no single solution meets these requirements.

Synthesizing new motions based on existing ones is the classic topic in computer graphics of *motion synthesis*. After years of study, graph-based frameworks have become the de facto standard solution to this problem. In such a framework, the task of synthesizing a new motion is typically cast as finding an optimal path in a pre-constructed *motion graph* [Arikan and Forsyth 2002; Kovar et al. 2002]. Each node denotes a motion segment in the database while each edge holds the desirability of transition between its associated nodes. Using such a graph-based scheme, Kim et al. [2003] made the first attempt to synthesize rhythmic motions by adding constraints linking motion beats and rhythmic patterns. Shiratori et al. [2006] and Kim et al. [2006] formally posed the problem of music-driven dance motion synthesis, and further developed more sophisticated rules to associate dance motion segments with input music clips. However, the fundamental drawbacks of existing graph-based methods are the underestimation and oversimplification of dance choreography, leading to clearly visible artifacts when tested on a larger database. On the one hand, hand-crafted basic features like beat and rhythm, though intuitively seeming reasonable, are actually incapable of modeling the deep intrinsic contextual connections between music and dance, such as style consistency, structural plausibility, and so on. On the other hand, music aside, existing methods fail to address widely used choreographic rules, leading experienced dance artists to comment that the synthesized motions, while appearing to be a set of dance movements fluidly pieced together, overall do not look like a well-composed artform.

With the booming of artificial intelligence technologies, deep generative techniques have successfully been applied to synthesizing various types of data, including images, text, etc. In this context, the problem of music-driven dance motion synthesis has also been considered [Alemi et al. 2017; Tang et al. 2018]. When properly set up, these methods do appear to grasp some deeper relationships between music and dance than traditional techniques, yet still fail to meet the standard required for real use. Their most obvious shortcoming is poor controllability, since synthesis performed by neural networks works as an inexplicable black box, which is a huge drawback for a practical production tool. Moreover, from a machine learning perspective, neural networks characterize data by projecting it into a low-dimensional latent space, during which process, high-frequency motion details are considered to be noises and intentionally ignored. This inevitably lowers the quality of synthesized dance motions, causing them to be 'dull' and 'blurred'. Furthermore, existing methods typically lack explicit attention to professional choreographic rules. As a consequence, the generalizability of the trained model is limited. As Alemi et al [2017] note, their model may generate strange and unaesthetic movements given music outside the training set.

Fortunately, after multiple rounds of iteration with professional artists and systematic study of the theory of choreography (see e.g., [Mason 2012; Nor and Stepputat 2016]), we have found some widely used choreographic rules that can be utilized to facilitate the problem of music-driven dance motion synthesis. In particular:

- the style of music and body movements should be consistent, conveying similar mood and tone,
- each synchronized dance and music segment should present the same rhythmic pattern, while rhythmic patterns in dance phrases appear with great regularity,
- the organization of a dance should be coordinated with the structure of the corresponding music, e.g., repeated musical phrases (verse and chorus) are typically associated with repeated movements, while identical meters in a phrase often correspond to symmetrical movements.

Following these rules allows the synthesized dances to reach a suitable aesthetic standard as expected by professional artists. Based on these rules, we have developed ChoreoMaster, a choreography-oriented music-driven dance motion synthesis system (see Fig. 1). As illustrated in Fig. 2, starting from an annotated database containing both paired and unpaired music and dance motion sequences, we first capture the music-dance connections through a choreography-oriented choreomusical embedding module (see Section 3). Specifically, we find a choreomusical style embedding by mapping music and dance phrases into a unified space where phrases of similar style are closely clustered, and a choreomusical rhythm embedding by identifying rhythm patterns for each meter of music or dance movement. The learned choreomusical embedding is then incorporated within a novel choreography-oriented graph-based motion synthesis framework (see Section 4), which can robustly and efficiently generate high-quality dance motions following various choreographic rules, while simultaneously offering great controllability to users. Experimental results demonstrate that ChoreoMaster can robustly and efficiently generate diverse high-quality dance motions widely recognised by professional artists. ChoreoMaster has successfully produced hours of dance assets for several projects in Netease Games and to our best knowledge, is industry's first production-ready tool for this purpose.

The contributions of this paper are thus:

- we introduce three rules from choreography theory, which greatly facilitate music-driven dance motion synthesis;
- we develop a cross-domain embedding framework, incorporating the introduced rules, to correctly and effectively characterize complex choreomusical relationships from limited available high-quality music/motion data, which successfully casts qualitative choreographic knowledge into computable metrics;
- we present the first production-ready dance motion synthesis system, which can robustly generate high-quality dance motions in a highly controllable way;
- we demonstrate that deep features, a graph-based framework and traditional optimization methods can be effectively combined to provide semantically correct, robust and controllable productional audio-based animation synthesis tools.

2 RELATED WORK

In this section, we discuss previous work in related areas.

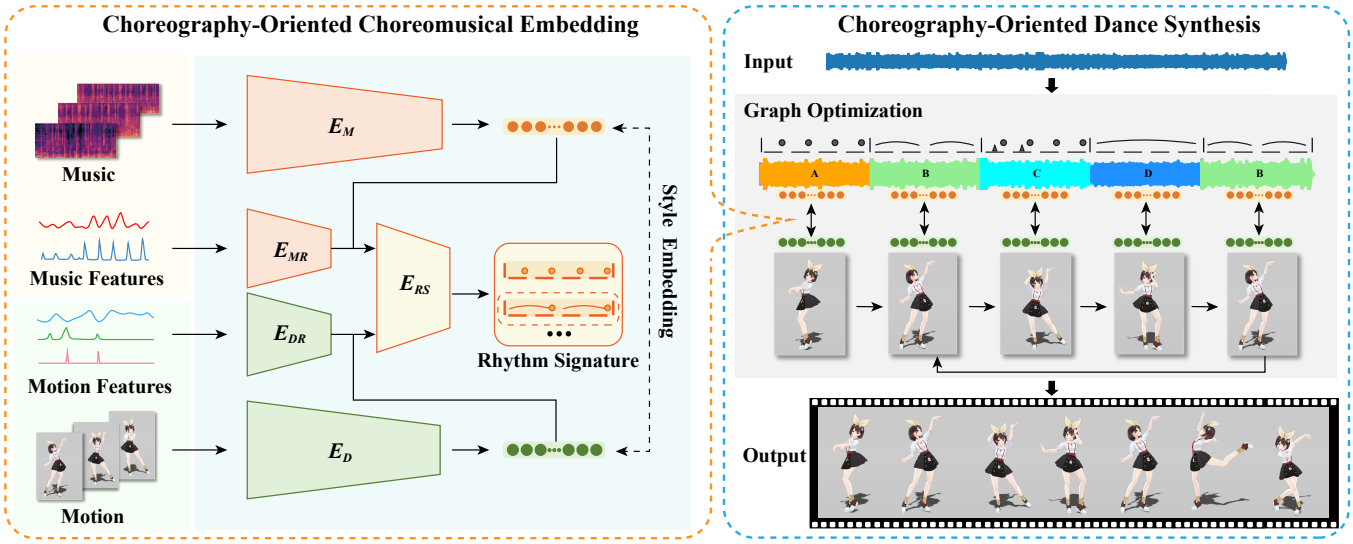

Fig. 2. ChoreoMaster comprises two choreography-oriented modules: a choreomusical embedding module to capture music-dance connections, and a graph-based motion synthesis module to generate high-quality dance motions, following various choreographic rules, from input music.

2.1 Graph-based Motion Synthesis

Motion synthesis has long been an important topic in computer graphics; its main goal is to synthesize new 3D skeletal motions from an existing motion database. Lamouret et al. [1996] raised the idea and presented the first prototype system to create new motions by cutting-and-pasting together existing motion segments from the database. Arikan et al. [2002], Kovar et al. [2002] and Lee et al. [2002] formally introduced the concept of graph-based motion synthesis, casting the problem as finding paths in a pre-constructed *motion graph*. Kim et al. [2003] extended this framework to cope with rhythmic motions by introducing constraints involving motion beats and rhythmic patterns. Shiratori et al. [2006] and Kim et al. [2006] developed more sophisticated music-to-dance matching constraints when path finding in the *motion graph*. Ofli et al. [2011] and Manfrè et al. [2016] formulated this problem using a hidden Markov model (HMM), allowing it to be efficiently solved using dynamic programming or beam search algorithms. Berman et al. [2015] discussed the possibilities of generating new kinds of dance movements using a *motion graph*, to assist the creative processes of dancers and choreographers.

Over time, graph-based frameworks have become the de facto standard solution to the motion synthesis problem, because of their numerous advantages. For instance, Yang et al. [2020] recently utilized a graph-based framework to successfully synthesize body motions for social conversations. With a properly constructed *motion graph*, transitions between adjacent segments of the synthesized motion are guaranteed to be smooth. However, from the perspective of professional artists, a dance is more than just a sequence of dance elements smoothly stitched together. Technically, our system can be viewed as a choreography-oriented extension to the traditional graph-based motion synthesis framework, from *motion graph* building to graph-based optimization, in which choreographic rules are respected throughout.

2.2 Music-to-Dance Cross-Modal Mapping

Properly formulating the connections between music and dance is of vital importance to music-driven dance motion synthesis. Various attempts have been made to better associate dance movements with music. Early approaches [Kim et al. 2006; Lee et al. 2013; Ofli et al. 2008; Shiratori and Ikeuchi 2008; Shiratori et al. 2006] typically perform similarity-search based on features detected in music signals (e.g., onset, chroma, MFCC) and dance motions (e.g., movement speed, joint trajectory). Image features have also been explored to extract visual beats from dancing videos [Davis and Agrawala 2018]. Learning-based methods have also been explored. For instance, Fan et al. [2011] use a boost-based learning algorithm to regress music to a motion mapping score and Fukayama et al. [2015] adopt a probabilistic model to measure the likelihood of assigning a dance motion to a given piece of music.

Although these methods can exploit some superficial connections between music and dance, they typically fail to model the deep intrinsic choreomusical relationships. Furthermore, the requirement of paired data also prevents these algorithms from benefiting from more readily available unpaired music and motion data.

Now, choreography is an empirical discipline and as mentioned before, there are some undeniable general rules which should be followed. Following these principles, we develop a novel choreography-oriented choreomusical embedding framework with two key elements: in a cross-modal style embedding process, we learn to map music and dance segments into a unified latent space where segments conveying similar mood and tone are closely clustered, then in a choreomusical rhythm embedding process, we learn to assign possible rhythm signatures for each meter of music or dance movement. Unpaired music and dance motion sequences can also be used by our framework. Quantitative and qualitative evaluation of our

approach demonstrate that the choreomusical relationships characterized by our framework can greatly promote the soundness and naturalness of synthesized dance motions.

2.3 Deep Generative Dance Motion Synthesis

Recently, deep generative methods have also been applied to the problem of dance motion synthesis. Based on the dimensionality of the motion data, existing methods can be divided into 2D and 3D solutions. Lee et al. [2019] presented the first 2D music-to-dance generation framework, which models dance units with a VAE (variational autoencoder) and recurrently generates dance sequences using a GAN (generative adversarial network). Since the human skeleton naturally forms a graph, Ren et al. [2020] and Ferreira et al. [2020] employ GCNs (graph convolutional networks) to improve the naturalness of generated 2D dance motions. Notwithstanding these advances, 2D dance motion synthesis has a different goal from our system. Instead of producing dance animation assets, the 2D pose sequences produced act as intermediate guidance for generation of dance videos. The absence of the third dimension severely limits application of 2D frameworks to 3D scenarios.

The idea of synthesizing 3D human motion with neural networks was first suggested by Grzeszczuk et al. [1998]. Then, Lee et al. [2006] employed neural networks to synthesize human neck movements. Since the rise of deep learning techniques, various time-series data generation frameworks have been adapted for music-driven 3D dance motion synthesis purpose, for instance, temporal convolutional autoencoder [Holden et al. 2016], FCRBM (factored conditional restricted Boltzmann machine) [Alemi et al. 2017], LSTM-autoencoder [Tang et al. 2018], CSGN (convolutional sequence generation network) [Yan et al. 2019], GAN [Sun et al. 2020], Bi-LSTM combined with temporal convolution [Zhuang et al. 2020], transformers [Li et al. 2021], etc. The key idea behind all these methods is to translate music into motion encoded in a low-dimensional latent space, and then recover corresponding dance motions by decoding from the latent space. As noted, the poor controllability and unstable performances resulting mean that such methods are unsuited to practical production environments. To alleviate these issues, some recent approaches [Duan et al. 2020; Ye et al. 2020] translate music into sequences constructed from a set of predefined dance action units, instead of skeletal dance motions. However, even smooth transitions between dance phrases cannot be always guaranteed in such methods, as noted by Duan [2020]. It is clear that graph-based frameworks are more powerful and flexible in arranging action units, while providing much more controllability and interpretability.

3 CHOREOGRAPHIC-ORIENTED CHOREOMUSICAL EMBEDDING

3.1 Background

Music and dance have been inextricably interwoven since time immemorial and their relationship has evolved with human civilization. The study of choreomusical relationships has formed a discipline called choreomusicology [Mason 2012], which summarizes the theory and practice of dance choreography. Obviously, generating dance motions which respect choreographic rules is essential to our system. However, as in every other art form, the evaluation of dance aesthetics and choreomusical style and rhythm relationships is a rather complicated matter, and difficult to formalize. Specifically, style and rhythm are two interrelated factors in both music and dance, relatively independently, yet closely correlated. On the one hand, music and dance following the same rhythmic patterns might present very different auditory and visual styles, and vice versa. On the other hand, the distribution of rhythmic patterns is strongly related to the musical and dance style: for instance, soothing music tends to have a soft rhythm while rock music has a strong beat.

Properly formulating the choreomusical distance between music and dance addressing the complex style and rhythm relationships is of vital importance to music-driven dance motion synthesis. Moreover, such choreomusical relationships are preferably presented in a disentangled way as this would significantly improve the system's interpretability and controllability. To this end, we use a choreography-oriented choreomusical embedding framework—see Fig. 3, which we explain next.

3.2 Choreomusical Style Embedding

Style consistency is a basic requirement for a dance composition. Associating vigorous movements with soothing music would make a peculiar dance. A straightforward way to keep style consistency is to manually divide music and dance into various categories according to style, and force the synthesis algorithm to pick dance movements from the same category as the input music. However, such a solution is actually insufficient to achieve a satisfactory result as well as being difficult to put into practice. Firstly, the boundaries between different music or dance styles are not always so clear. Assigning style labels to music or dance requires tremendous expertise. Secondly, the classification criteria for music and dance styles are different, since they evoke auditory feelings and visual feelings respectively. For instance, both music and dance can be classified by genre, but most music or dance genres do not have an equivalent tag in its counterpart. Thirdly, each main style of music or dance contains numerous sub-styles (e.g., hip-hop can be further classed as popping, locking, breaking, urban, etc.), making it even harder to achieve style consistency by explicitly classifying music and dance data.

To tackle this problem, we adopt a choreomusical embedding network to implicitly model the connections between music and dance styles. Our key idea is to map music and dance segments into a unified embedding space where segments conveying similar mood and tone are closely clustered. Specifically, we first use unpaired music and dance data to independently train two classification networks, then paired data is utilized to transform the two feature spaces into a unified embedding space, where items of music and dance remain classifiable, while paired music and dance items stay as close as possible.

The architecture is illustrated in Fig. 3 (left). We mainly adopt the state-of-the-art music tagging network in [Choi et al. 2017] as our backbone for the music encoding branch E_M. It is composed of four convolutional block layers and two GRU layers. Symmetrically to E_M, we build a dance encoding branch E_D, except that the convolutional blocks are replaced by graph convolutional blocks. The general purpose of E_M and E_D is to compress music and dance

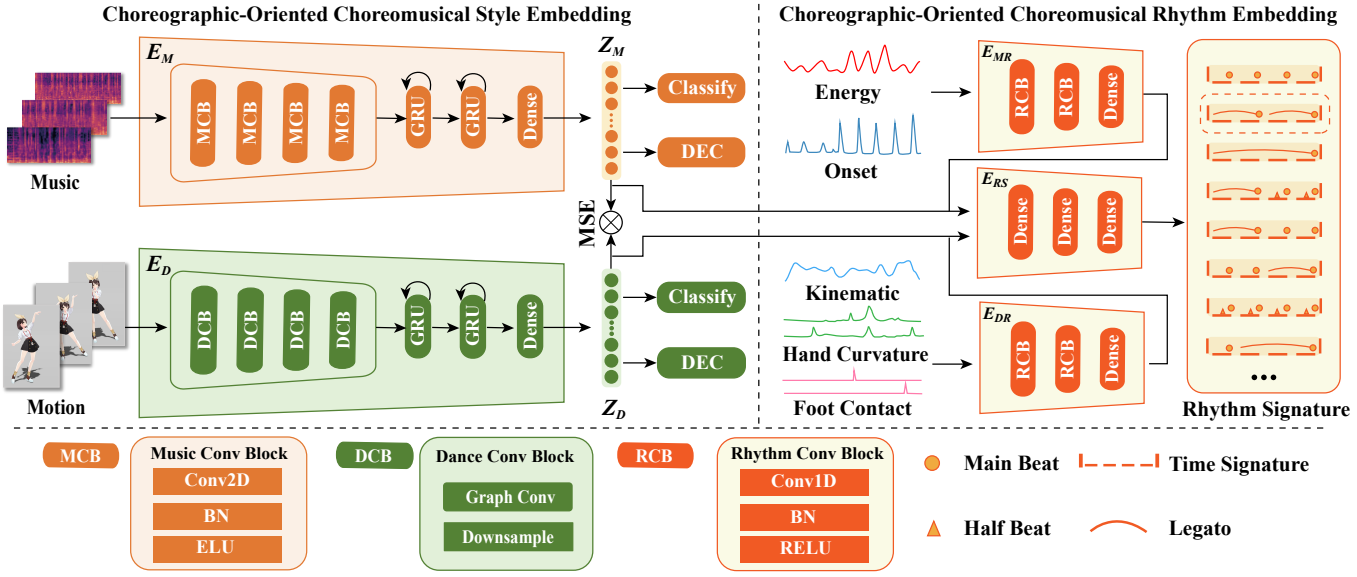

Fig. 3. Our choreographic-oriented choreomusical embedding network contains a choreomusical style embedding network and a choreomusical rhythm embedding network.

sequences into two 32-dimensional embedding vectors, i.e., Z_M and Z_D respectively. In our implementation, music data is downsampled to 16kHz and represented as log-amplitude mel spectrograms (computed with 96 mel bins and 160 hop size), while dance motion data is represented by global joint positions (18 main body joints are used, not including fingers and toes). A musical/dance phrase is considered to be the minimum unit expressing a style, so we set the input length to the length of a typical music or dance phrase, 8 seconds (i.e., 240 key frames in 30 fps dance motions). Therefore, the input shape for E_M is [1, 96, 800] and E_D is [3, 18, 240].

To benefit from the vast amount of unpaired music and motion data, we employed a two-stage training procedure. In the first phase, the music and dance branches are trained independently using all labeled unpaired music and dance data. To better reflect latent substyles in the learned embedding space, we also incorporate the unsupervised deep embedding clustering (DEC) strategy proposed by Xie et al. [2016], which encourages data in the feature space of a classification network to be better clustered.

The training losses for music and dance embedding are:

$$\begin{aligned} \mathcal{L}_m &= \lambda_1 L_m + \lambda_2 L_{\text{dec}} \\ \mathcal{L}_d &= \lambda_1 L_d + \lambda_2 L_{\text{dec}} \end{aligned} \tag{1}$$

where L_m, L_d are the classification losses (i.e., cross-entropy loss) of music and dance respectively, L_{dec} is the the KL divergence loss defined in Equation (2) of [Xie et al. 2016], and λ_1 and λ_2 are balancing weights. Then, in the second phase, two branches are jointly trained using synchronized music and motion pairs, where the training loss is defined as:

$$\mathcal{L}_{\text{style}} = \lambda_3 L_d + \lambda_4 L_m + \lambda_5 L_z \tag{2}$$

where L_m and L_d are the classification losses, L_z is the MSE loss between Z_M and Z_D, and $\lambda_3, \ldots, \lambda_5$ are weights. Through these two phases of training, we can map any music and dance segments into a unified choreomusical embedding space, where style consistency between music and dance can be measured by the Euclidean distance between the corresponding embedding vectors.

3.3 Choreomusical Rhythm Embedding

Body movements should be coordinated with musical rhythms in a well composed dance. In music theory, the term rhythm is often expressed in terms of the musical meter. Meter refers to the organizational patterns of beats, while a beat is the basic temporal unit of music. Accordingly, we can use beat and meter for dance motions. Typically, musical beat corresponds to pulses of sound in music, while dance beat corresponds to pausing or sharp turning of body movements. Formally, meter is indicated by time signature (i.e., 2/4, 3/4, 4/4 and etc.), where the upper number depicts the number of beats in a bar, and the lower denotes the tempo duration of a beat. For instance, a 4/4 time signature contains four quarter note beats in each bar.

Unlike style, rhythm can be clearly represented using musical notation. However, matching dance movements to musical rhythm is still a difficult task. Music usually includes multiple instrumental tracks and vocals, while dance movements often involve many simultaneously moving joints. Each musical track or body joint thus has its own beat pattern, and locating the true beat pattern which should be coordinated to is a challenging task. For example, choreographers may prefer to follow drum beat, piano melody or human vocals in different parts of the music when composing a dance to it. To better understand the rhythmic relationships between music and dance, we asked professional artists to manually specify the beat patterns of dances in our database. We could thereby retrieve the beat patterns of a music from its synchronized dance. By analyzing their labelling results, we found that beat patterns in each meter can be mathematically represented as a binary vector, which we refer

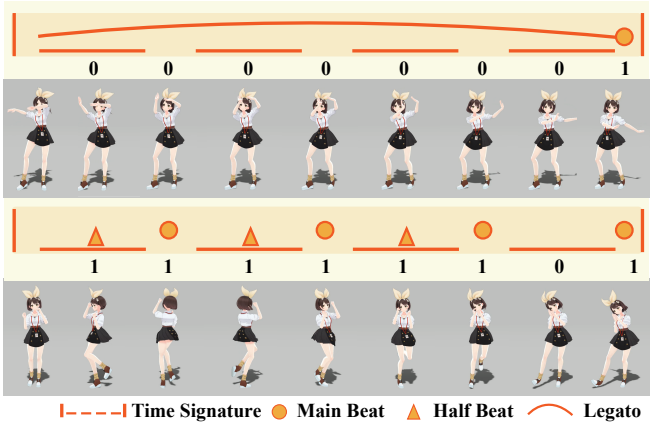

Fig. 4. Rhythm signature examples. Even bits denote the presence of *regular beats*, while odd bits depict *half beats*. Consecutive zeros indicate a *legato*.

to as rhythm signature, a natural comprehensive form of unified choreomusical rhythm embedding.

Since all dance motions in our dataset are structured in four-beat meters, the proposed rhythm signature consists of 8 bits in our system (see Fig. 4). In each rhythm signature, even bits denote the presence of *regular beats* (1 : present, 0 : not present) which correspond to the evenly-spaced standard beats indicated by the time signature, while odd bits depict *half beats* (1 : present, 0 : not present) which account for the rhythmic points in-between two regular beats (usually caused by ties, rests, or dots, or simply because the adjacent beat is constructed from multiple smaller beats). Consecutive zeros in a rhythm signature indicate a *legato*, or smooth period in music and dance motions. The distance between two rhythm signatures can be defined using Hamming distance, the number of bit positions in which the two bit patterns differ. To put more emphasis on *regular beats*, we allocate different weights to different bits when calculating Hamming distance: 1.0 for *regular beats* and 0.5 for *half beats*.

Theoretically, there are $2^8 = 256$ types of rhythm signature. However, statistical analysis of the labeled data showed that the number of common rhythm signatures is only about 13. Another interesting discovery was that the distribution of rhythm signatures varies greatly with dance type. For instance, Chinese traditional dances tend to have more *legatos* while hip-hop dances tend to have more *half beats*. This illustrates the interweaving relationships between style and rhythm. Based on the above ideas, we designed a rhythm signature classification network to effectively acquire choreomusical rhythm embeddings for music and dance.

The architecture of this network is illustrated in Fig. 3(right); it comprises three blocks. There are two separate feature extraction blocks for music E_{MR} and dance E_{DR} respectively, each constructed with two convolutional layers and one dense layer. Finally there is a shared block, E_{RS} for rhythm signature classification, constructed with three dense layers. Style embedding vectors of the corresponding music segment Z_M and dance segment Z_D are concatenated with the feature vectors extracted by E_{MR} and E_{DR} respectively, since statistical correlations between style and rhythm have already been observed. It is noteworthy that there are some differences

Fig. 5. Without considering style compatibility, improper edges may appear, resulting in a lovely motion switching to a sexy or cool motion as illustrated in the figure.

between the style and rhythm embedding branch. Firstly, the goal of rhythm embedding is to determine explicit rhythm signatures rather than latent vectors, because rhythm signatures have a clear definition and thus provide more interpretability and controllability to our system. Secondly, the tempo unit used for rhythm embedding is set to one bar (i.e., 2 seconds in our system) rather than phrases, as rhythm signatures are defined in terms of bar. Here, the style embedding of a bar refers to the style embedding of the phrase it belongs to. Thirdly, as beats are reflected by pulses changes in music and speed/direction changes in joint movements, it is not necessary to train the network from low-level original signals. Instead, we feed the network with certain extracted features. Specifically, the spectral onset strength curve [Böck and Widmer 2013] and RMS energy curve for music (dimension: [2, 200]); the motion kinematic curve, two hand trajectory curvature curves and two foot contact curves for dance (dimension: [5, 60]). The motion kinematic curve is computed using the weighted angular velocity function proposed by Shiratori et al. [2006]. The hand trajectory curvature curve records the curvatures of the trajectories of the two wrist joints, and the foot contact curve records contact information between both feet and the floor. All blocks in this network are jointly trained using labelled paired music and dance data with the following loss function:

$$\mathcal{L}_{\text{rhythm}} = \lambda_6 L_{dr} + \lambda_7 L_{mr} \tag{3}$$

where L_{dr} and L_{mr} are the classification loss for dance and music respectively, and λ_6 and λ_7 are weights. To penalize large prediction errors, as well as conventional cross-entropy loss, we add the weighted Hamming distance between the predicted rhythm signature and the ground truth rhythm signature when calculating L_{dr} and L_{mr}.

4 CHOREOGRAPHY-ORIENTED DANCE SYNTHESIS

As noted, comprehensive controllability is of great importance in a production-ready tool. Therefore, we adopt a graph-based motion synthesis framework. A typical graph-based motion synthesis framework contains two key steps: *motion graph* construction and graph-based optimization. In this section, we explain how the learned choreomusical embedding and other choreographic rules are incorporated into our graph-based motion synthesis framework.

4.1 Motion Graph Construction

A *motion graph* is a directed graph where each node denotes a motion segment in the database while each edge depicts the cost of transition between two adjacent nodes. Conventional graph-based dance motion synthesis systems typically segment dance

Fig. 6. Dance motions in our database are augmented with mirroring, blending and reshuffling.

motions into motion beats, and calculate the transition cost based on the distances between joint positions and movement speeds. However, this would pose two problems: first, correlations between motions within a dance meter are ignored; second, such metrics fail to account for style compatibility between motions (see Fig. 5).

In our system, these issues are addressed by introducing choreographic rules. Instead of dance motion beat, each node in our *motion graph* corresponds to a dance motion meter. The learned style embedding vector and labeled rhythm signature are also attached to each graph node. To make better use of existing data and to encourage more diverse synthesised results, dance motion meters in our database are augmented in three ways (see Fig. 6):

- *mirroring*: dance movements are mirrored left to right;
- *blending*: the upper and lower body movements in two different meters are blended to create a new motion meter;
- *shuffling*: motion beats in two different meters are shuffled to create two new motion meters, e.g., '1234' and 'abcd' may produce '12cd' and 'ab34' if they can be stitched smoothly.

The *mirroring* operation is applied to all motion meters in the database, while *blending* and *reshuffling* are very conservatively performed. i.e., only between meters with the same rhythm signature and very close style encodings. All motions augmented through *blending* and *reshuffling* were manually checked. New nodes are created in the *motion graph* for all valid augmentations.

Style compatibility is also respected in our *motion graph*. Specifically, the edge transition cost between two nodes D_p and D_q is defined as:

$$T(D_p, D_q) = \lambda_8 T_d + \lambda_9 T_z \tag{4}$$

Where T_d is a regular motion transition cost, computed as the summed distance of positions (meters), rotations (radians) and speeds (meters per second) between main joints (18 joints here) in transitional frames of two adjacent nodes, and T_z is the Euclidean distance between two style embedding vectors. λ_8 and λ_9 are weights. An edge is created in the graph if the transition cost between adjacent nodes is below a threshold δ_T. A higher δ_T results in more edges in the graph, which increases the diversity of the results, but may also cause artifacts as bad transition edges may also be included in the graph.

4.2 Graph-based Optimization

In the graph-based framework, each synthesized motion corresponds to a path in the *motion graph*. Therefore, in our system, synthesizing dance motions for the input music can be viewed as finding optimal paths, satisfying various choreographic rules, in the graph (see Fig. 7). Given an input piece of music, we first divide it into bars using the automatic musical bar detection algorithm suggested in [Gainza 2009]. Then we retrieve all musically meaningful phrases using the music segmentation and similarity labeling method proposed by Serra et al [2012; 2014]. Similar bars within a phrase are further detected (having spectrogram difference within a small threshold) and given an identity ID. Overall, each bar M_i in the music is given a structural tag (A_1^1, A_2^1, B_1^1 and B_2^2 in Fig. 7), where A depicts the phrase identity ID, the subscript and the superscript denote its index and meter identity ID in the phrase respectively. Then, for each meter M_i in the music sequence $M = \{M_i | i = 1, \ldots, n\}$ we obtain its style embedding Z_{M_i} and the top K possible rhythm signatures $\{R_{M_i}^1, \ldots, R_{M_i}^K\}$. The goal of our system is to assign a dance motion node D_i in the *motion graph* to each musical meter M_i so that the following cost is minimized:

$$C = \lambda_{10} \sum_{i=1}^{n} C_d(i) + \lambda_{11} \sum_{i=1}^{n-1} C_t(i, i+1) + \zeta \sum_{i<j}^{n} C_s(i, j) \tag{5}$$

where C_d, C_t and C_s are the data term, transition term and structure constraint term respectively, $\lambda_{10}, \lambda_{11}$ are weights, and ζ is a large penalty coefficient.

Data term. $C_d(i)$ accounts for the style and rhythm matching cost between music meter M_i and dance motion meter D_i, and is defined as:

$$C_d(i) = \lambda_{12} G_z(Z_{M_i}, Z_{D_i}) + \lambda_{13} \min_{k=1}^{K} G_r(R_{M_i}^k, R_{D_i}) \tag{6}$$

where G_z and G_r are style embedding distance and rhythm signature distance between music/dance meters respectively, and λ_{12} and λ_{13} are two weights.

Transition term. $C_t(i)$ ensures a smooth transition between adjacent motion segments in the synthesized motion, and equals the transition cost stored on graph edges: $C_t(i, i+1) = T(D_i, D_{i+1})$.

Structure term. C_s addresses structural consistency between music and dance. Choreographers often use motion repetition to echo the repetitive structure in music. For instance, repeated musical phrases (e.g., verse and chorus) most likely correspond to repeated movements, while identical bars in a phrase often correspond to symmetrical movements. Derived from these choreographic rules, we include two structural constraints. For *repeat constraint*, D_i and D_j should be the same motion if M_i and M_j belong to different phrases while the phrase identity ID and index ID are the same. And for *mirror constraint*, D_i and D_j should be two mirrored motions, if M_i and M_j belong to the same phrase and their meter identity ID are the same. For each pair of D_i and D_j, $C_s(i, j)$ is set to 1 if any of the constraints is violated:

$$C_s(i, j) = \begin{cases} 0, & \text{if } D_i \text{ and } D_j \text{ satisfy the constraints;} \\ 1, & \text{otherwise.} \end{cases} \tag{7}$$

Other practical requirements can also be turned into constraints and easily incorporated into this framework. The optimal dance motion sequence can be efficiently synthesized using a dynamic programming algorithm [Forney 1973]. By running it multiple times and skipping used movements, different dance motions for the same input music can be easily produced.

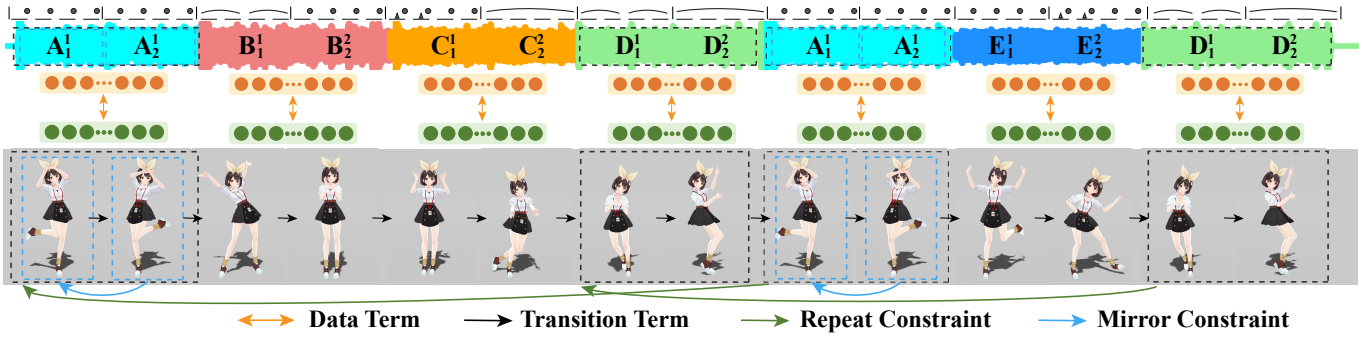

Fig. 7. Choreography-oriented dance synthesis process. We use several choreography-oriented constraints including a data term, a transition term, and repeat/mirror constraint to synthesize high-quality dance motions meeting professional aesthetic requirements. Each differently colored music segment represents a musical phrase with 4-8 bars. For simplicity, we only show two signatures and one style embedding for each musical phrase.

5 EVALUATION

We now evaluate our method. We first demonstrate the setup of our system, followed by a quantitative and qualitative comparison to several state-of-the-art methods. Then, we make an overall analysis of the performance and efficiency of our system, and finally, demonstrate the promising controllability of ChoreoMaster.

5.1 System Setup

In this section we introduce our dataset and system environment.

Table 1. Style distribution of our music and dances. Each piece is labeled with a two-dimensional style attribute.

	Style 1	Duration	Style 2	Duration
Music	Chinese	20.7 h	Mild	21.4 h
	Japanese	40.8 h	Exciting	32.9 h
	English	18.7 h	Neutral	48.2 h
	Korean	22.3 h	–	–
Dance	Anime	7.5 h	Sexy	1.6 h
	Hip-Hop	5.4 h	Lovely	5.9 h
	K-pop	3.9 h	Cool	5.4 h
	Tradition	3.1 h	Gentle	2.7 h
	–	–	Other	4.3 h

Table 2. Rhythm signature distribution in labeled results.

R_m	Ratio	R_m	Ratio	R_m	Ratio
00000001	20.1%	00000101	3.9%	00010001	4.2%
00010101	4.3%	01000001	3.8%	01000101	4.5%
01010001	3.8%	01010101	30.1%	01010111	6.4%
01110111	5.2%	01111111	4.6%	11111101	3.8%
11111111	5.3%	-	-	-	-

Database. Our motion resources consist of high-quality dance mocap resources and MikuMikuDance (MMD) resources collected from the anime community. In total, we have 19.91 hours of dance motions, of which 9.91 hours have paired music. By utilizing the motion augmentation techniques described in Section 4.1, our database dance motion was expanded to 2.56 times. We additionally collected a large music dataset containing 1,954 songs with a total duration of 102.5 hours. The dance and music are semi-automatically segmented into bars and phrases. All music and dances are labeled with two-dimensional styles, as shown in Table 1. The labels in Table 1 came from a large collection of candidate tags. We kept them because the annotations from different people using these labels were relatively consistent, so more convincing. ChoreoMaster does not rely on the necessity or sufficiency of a specific tagging system, because these tags are not used as hard matching constraints in the synthesis process. All dances were manually labeled with rhythm signatures by professional artists; rhythm signatures of music that has synchronized dances could thereby be obtained; rhythm signatures whose frequency is less than 0.5% were considered to be noises and were manually reclassified as another common signature. In total we kept 13 rhythm signatures, distributed as shown in Table 2. Certain rhythm signatures are more frequent in certain styles of dance: for example, rhythm signature 00000001 occurs 44.8% of the time in traditional dances, but only 11.9% of the time in hip-hop dances. Besides, we randomly divided the music-dance pairs into three parts, i.e, the training set (80%), the validation set (10%) and the test set (10%).

System Environment. We set the hyperparameters to: $\lambda_1, \ldots, \lambda_{13} = 0.7, 0.3, 0.5, 0.5, 5, 0.4, 0.6, 1, 1.5, 1, 2, 1, 1.5, \delta_T = 20, K = 3, \zeta = 1000$. $\lambda_1, \ldots, \lambda_7$ are involved in the embedding framework and were qualitatively optimized using grid search; details are given in the supplemental document. $\lambda_8, ..., \lambda_{13}$ and δ_T control the synthesis process and may be adjusted by the user according to desire for smoothness, novelty and importance of style and rhythm consistency. Default values of $\lambda_8, ..., \lambda_{13}$ and δ_T were determined by consulting professional artists. Our experiments show that ChoreoMaster works well for parameters within a wide range. All networks were trained using PyTorch on a P40 GPU server. Our dance synthesis system was tested on a desktop with a 3.20GHz i7-8700 CPU, 16GB RAM and a GTX 1080Ti GPU. E_M and E_D were trained using the Adam and SGD optimizers respectively, and the number of clusters in DEC was

set to 12 for music and 20 for dance. E_{DR}, E_{DM}, E_{RS} were trained using the SGD optimizer. We trained with a batch size of 64 and a learning rate of 0.001, for 500 epochs. In total, 14 hours were taken to train the embedding networks, 13 hours for style embedding and 1 hour for rhythm embedding.

5.2 Comparisons

In this section, we compare our method to several alternative dance generation methods to demonstrate the advances made by our system. We start with a brief introduction to these methods, followed by a quantitative comparison, and a qualitative evaluation via a comprehensive user study. Further details are given in our supplementary material.

We chose two traditional dance synthesis methods and three recent state-of-the-art 3D dance synthesis methods based on generative models for comparison. Lee et al. [2013] proposed a traditional dance synthesis method that retrieves candidate motions by evaluating similarity between input music clips and existing music clips. Fukayama et al. [2015] built a probabilistic model to optimize the dance sequence for the input music. Yan et al. [2019] proposed CSGN, which constructs skeleton sequences from latent variables using graph convolutions, and used it to generate 3D dance motions. Sun et al. [2020] proposed the DeepDance method, a GAN-based cross-modal association framework for 3D dance generation. Li et al. [2021] utilized a cross-modal transformer-based model for music-conditioned 3D dance generation.

Since currently we do not have access to the code or data of [Fukayama and Goto 2015] and [Li et al. 2021], we synthesized dances using the same music and compared with the results showed in their released demo videos, which can be found in our supplementary video. Additionally, we also carried out an ablation study by removing the style embedding, rhythm embedding and structural constraints in turn to generate results. Since these methods can only deal with music-dance pairs, to make the comparison fair, all methods were given our music-dance pairs with a duration of 9.0 hours for training, and the remaining 0.91 hours were left for testing. Besides, rhythm annotations were also used for data segmentation (in Yan et al. [2019] and Sun et al. [2020]) and beat detection (in Lee et al. [2013]). The comparisons were made upon the automatically generated dance results for 30 music clips with a duration of 30-90s, among which 20 clips were from the testing data. Fig. 8 illustrates one of the comparison results and more results are showed in our supplementary video.

Quantitative Evaluation. We adopt several evaluation metrics to quantitatively compare these methods, as shown in Table 3. These metrics are:

1. *FID score.* Fréchet inception distance (FID) [Heusel et al. 2017] was used to measure how close the distribution of generated dances is to that of the real ones. Following [Lee et al. 2019], we trained a motion auto-encoder on our dance dataset as the feature extractor. The FID in Table 3 shows our generated dances are much closer to the real ones than other methods, as our framework addresses much more choreographic disciplines.

2. *Beat accuracy.* This measures how accurately the motion beats are aligned to the music beats, represented by the ratio of aligned beats to all music beats. We used the labeled music rhythm signature as the ground-truth, and used the motion rhythm detection method proposed in [Shiratori and Ikeuchi 2008] to detect the motion kinematic beats. Table 3 shows that with the help of learned choreomusical rhythm embedding, ChoreoMaster achieved the highest beat accuracy of all methods. Results not using rhythm signature are much lower, showing the effectiveness of the rhythm signature.

3. *Diversity.* We follow [Lee et al. 2019] to evaluate the average feature distance between generated dances for different music inputs. The same feature extractor used in measuring FID was again used. Our method achieves the highest diversity score, as shown in Table 3. Furthermore, the results without the structural constraint are slightly higher, because the structural constraints impose some motion repetition.

Table 3. Comparison of our method to Lee et al. [2013], Yan et al. [2019] and Sun et al. [2020]. We also compare results of our method without style embedding (w/o EC), rhythm signature (w/o RC) or structural constraints (w/o SC).

Method	FID	Beat Accuracy	Diversity
Real Dance	2.7	92.6%	83.5
Lee et al. [2013]	24.5	38.4%	75.1
Yan et al. [2019]	94.6	8.2%	56.2
Sun et al. [2020]	87.4	12.7%	64.1
Ours (w/o EC)	20.5	85.2%	72.4
Ours (w/o RC)	17.9	58.3%	76.5
Ours (w/o SC)	18.5	83.8%	78.3
Ours	16.8	88.4%	77.9

User Study. Since evaluating dance quality is very subjective, we also performed a user study to help us qualitatively evaluate our method. We used 30 test music clips, and invited 35 participants, 10 of whom being choreographers or artists, to rate the following factors from 0 to 10: (1) dance realism (ignoring the music), (2) music-to-dance style consistency, (3) music-to-dance rhythm consistency and (4) music-to-dance structural consistency.

Results are shown in Fig. 9. Our method achieves higher scores than other methods, and is close to real dances. We can see that the generative models (Yan et al. [2019] and Sun et al. [2020]) achieve low scores throughout, since most of their dance results appear to be 'dull' and 'blurred', lacking aesthetic appeal, and having many artifacts. The results generated by Lee et al. [2013] are much better, but their dances show poor correlation with the music.

T-test results (see the supplementary materials for details) show that significant improvement is achieved by using the style embedding, rhythm embedding and structural constraints. Our method outperform the one without EC by more than 43% and 28% respectively for artists and normal users, outperforms the one without RC by more than 40% and 32% respectively for artists and normal users. and outperforms the one without SC by more than 34% and 28% respectively for artists and normal users. In comparison, the ground-truth only outperforms our results by no more than 6% and 2% respectively for artists and normal users.

Fig. 8. Dance motions generated by four different methods for a traditional Chinese song.

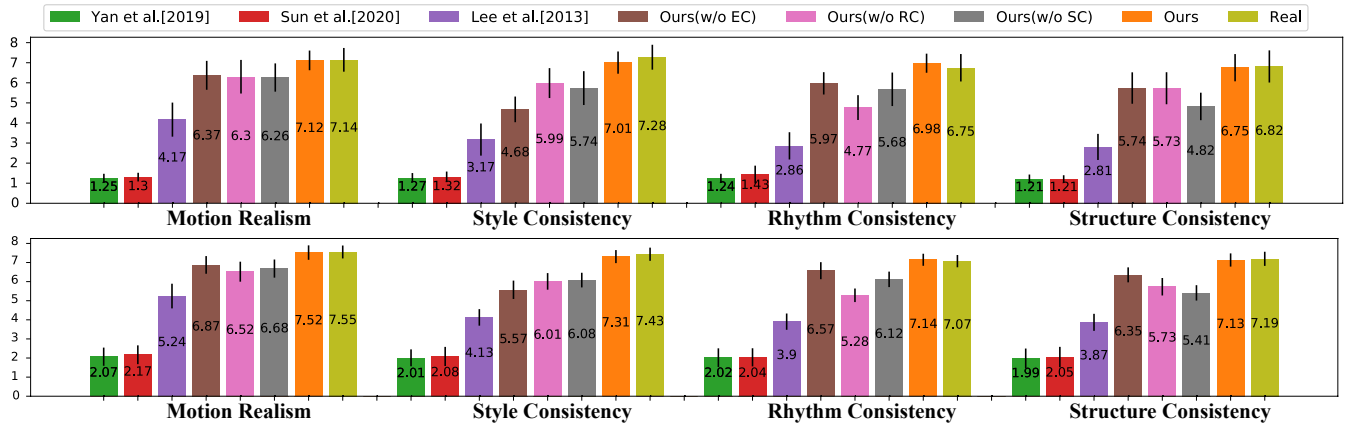

Fig. 9. User study results for ordinary users (above) and for choreographers and artists (below). Our method achieves higher scores than Lee et al. [2013], Lee et al.[2019], Yan et al. [2019] and Sun et al. [2020].

Since some music-driven 2D dance generative models have recently been proposed by Lee et al. [2019] and Ren et al. [2020], we also made a comparison with these methods, by projecting our 3D dance into 2D. Results are shown in the supplementary video.

5.3 System Analysis

In this section, we make an over all performance analysis of the learned choreomusical embeddings, and then consider the efficiency of our system.

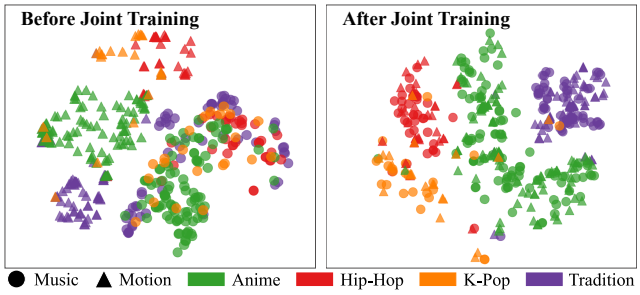

Fig. 10. T-SNE visualization of the choreomusical style embedding of musics and dances from the test set. Left, right: results before and after joint training. The embedding of music-dance pairs becomes much closer after joint training.

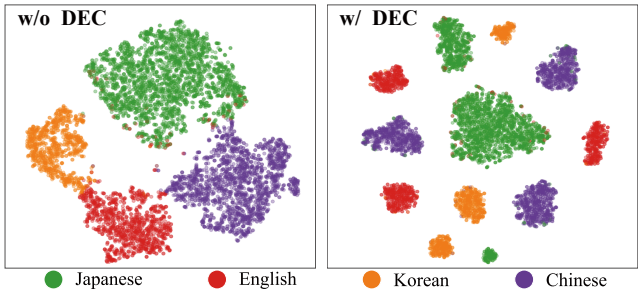

Fig. 11. T-SNE visualization of music distributions in the feature space of the classification network trained without DEC loss (left) and with DEC loss (right).

Table 4. Style classification accuracy for the first and second style dimensions (Accuracy 1, 2) and the Euclidean embedding distance of music-motion pairs.

Training Method	Type	Accuracy 1	Accuracy 2	Distance
Separate Training	Motion	72.4%	67.4%	1.464
	Music	95.1%	78.3%	
Joint Training	Motion	70.5%	65.7%	0.316
	Music	90.8%	77.1%	

Choreomusical Style Embedding. The purpose of our choreomusical style embedding network is to map the music and dance phrases into a unified latent space where segments of similar style are as close as possible. Here we evaluate its performance by showing the classification accuracy in Table 4 and a T-SNE visualisation of the choreomusical style embedding in Fig. 10. After the separate training, the style embeddings of music and dance are already classifiable, but the embeddings of paired data are far from each other. After the joint training step, we can see from Fig. 10 (right) that the paired music and dance have become much closer in the embeddings space, while the classification accuracy has only dropped slightly, showing that our choreomusical style embedding network have successfully reached it purpose.

To evaluate the impact of DEC [Xie et al. 2016], we additionally trained a style embedding network with the DEC loss removed from Equation 1. In our experiments, no obvious difference was found regarding the style classification accuracy of music/dance between the networks trained with and without a DEC loss. However, clusters of latent sub-styles were clearly better reflected in the feature space of the network trained with a DEC loss. Fig. 11 illustrates such distribution difference on music data.

Choreomusical Rhythm Embedding. We next evaluate the performance of our choreomusical rhythm embedding network. Table 5 shows the top-1 and top-3 classification accuracy of the network with or without the style embedding. The classification accuracy increases after joint training and the style embedding also contributes to the results, justifying our design of the shared network of E_{RS} and using the style embedding as input. We also note that the top-1 accuracy is not so high, while the top-3 accuracy is much higher. We calculate the minimum weighted Hamming distance to the ground truth in top-3 rhythm signatures for music and dance motions in the test set. The distributions are showed in Fig. 12, from which we can see that most of the values are less than 1.0. Therefore, we use the top-3 results during our graph optimization process, i.e., $K = 3$ in Equation 6.

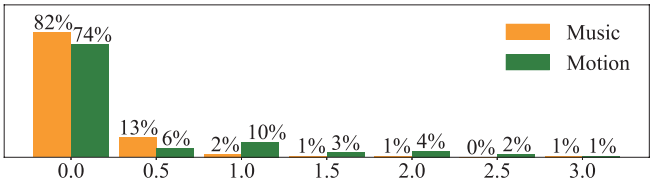

Fig. 12. Weighted Hamming distance of top-3 rhythm signatures after joint training with style embedding. Most distances are below 1.0.

Table 5. Top-1 and top-3 classification accuracy of our choreomusical rhythm embedding network with or without style embedding.

Training Method	Type	w/o Style		w/ Style	
		Top-3	Top-1	Top-3	Top-1
Separate Training	Motion	60.6%	31.8%	67.8%	40.9%
	Music	68.6%	44.9%	76.7%	54.3%
Joint Training	Motion	66.6%	36.9%	73.8%	47.1%
	Music	75.4%	51.3%	82.3%	59.1%

Speed. Our system is highly efficient and only needs a few seconds to synthesize a high-quality dance. The synthesis time is nearly linear to the motion duration, music duration and number of edges (determined by the transition threshold δ_T). We tested the speed using paired dance motion data (9.91 hours) and all dance motion data (19.91 hours), and two different settings of δ_T (i.e., 20 and 12); results are shown in Table 6. The time required to infer the choreomusical style and rhythm embeddings for input musics are less than 0.05s, thus omitted from the table. We also illustrate the distributions of node in-degrees and out-degrees of the constructed 19.91 hours' motion graph (see Fig. 13) to show their sparsity.

Table 6. Dance synthesis time required for different sizes of motion graph, different edge density, and different music durations.

Motion Duration	Edge Count	Music Duration	Synthesis Time
9.91 h	103k ($\delta_T = 20$)	61 s	0.9 s
		148 s	1.88 s
	248k ($\delta_T = 12$)	61 s	2.2 s
		148 s	4.5 s
19.91 h	328k ($\delta_T = 20$)	61 s	3.3 s
		148 s	6.4 s
	728k ($\delta_T = 12$)	61 s	5.7 s
		148 s	11.2 s

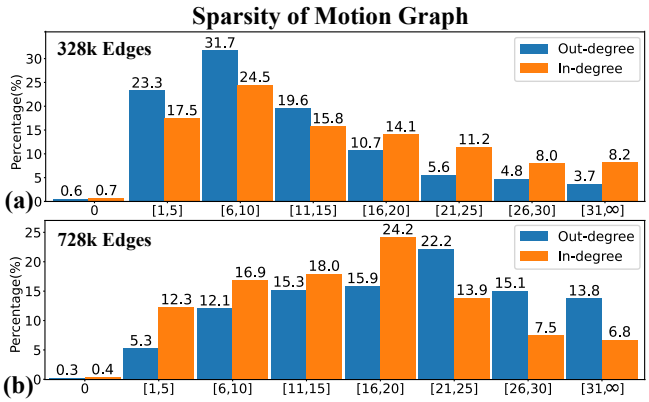

Fig. 13. Sparsity of the motion graph built on 19.91 hours of dance motions. (a): 328k edges; (b): 728k edges.

5.4 System Controllability

The controllability is of vital importance to a production-ready tool, and artists may have various individual requirements using our system. Here we briefly show how ChoreoMaster can adapt to some common requirements. We provide examples in the supplementary video.

Precise Control. The most common demand for artists is to have precise control over the synthesized results, for instance, forcing some specific dance motions to appear at specific locations in the synthesized results (motions having specific semantic connections with the music, for example), replacing unwanted motions without affecting other movements in the synthesized dance sequence, or specifying a custom trajectory to move along. ChoreoMaster can easily support such control by adding extra user constraints to the data term (i.e., Equation 6), specifically, forcing some user-specified dance motion node to appear or not appear at some user-specified temporal interval in the synthesized dance, or encouraging the character's root position to stay close to user-specified locations at user-specified time points.

Range Control. Another common requirement is to restrict the range of motion of the synthesized dances, to prevent dancers from moving off the stage—the stage is always limited in the game or film industry. This can be achieved by adding an extra range constraint to the cost function, penalizing motions violating the specified movement ranges.

Novelty Control. As noted by [Berman and James 2015], dance *motion graph* can also be used to assist the creative process of dancers and choreographers. In our system, this can be achieved by lowering the edge transition threshold δ_T (Section 4.1) and increasing the transition weight λ_{11} (Equation 5). Specifically, lower δ_T and higher λ_{11} tend to encourage the system to use more original dance motions, while setting a larger δ_T or smaller λ_{11} encourages more novel transitions (i.e., transitions not presented in the dance database) in the results. We received numerous positive feedbacks from professional artists about deriving inspirations from the novel transitions generated by ChoreoMaster.

6 LIMITATIONS AND FUTURE WORK

Although ChoreoMaster has successfully been put into practical use, limitations remain. Currently, it cannot synthesize dance styles that are absent from the database, like ballet, waltzes, etc. This can be addressed by further expanding our dance database. Furthermore, as noted in Section 3.3, all dance motions in our dataset are structured in four-beat bars, as typically required by our artists. Therefore, when dealing with three-beats-to-the-bar music, our system will still match it to four-beat meter dance motions. Again, further data acquisition can overcome this problem. Lastly, our system still cannot handle the semantic relationships between dance motions and music lyrics: human interaction is required to ensure semantic consistency. It would be interesting to incorporate further techniques into our system, like natural language processing modules.

7 CONCLUSION

In this paper, we have presented ChoreoMaster, a production-ready music-driven dance motion synthesis system. It includes a novel choreography-oriented choreomusical embedding framework to explore the connections between music and dance in terms of style and rhythm, using an annotated high-quality database. The learned choreomusical embedding is then incorporated into a novel choreography-oriented graph-based motion synthesis framework, which can robustly and efficiently generate high-quality dance motions respecting various choreographic rules while also offering great controllability to users. Experimental results demonstrate that ChoreoMaster can robustly and efficiently generate diverse high-quality dance motions widely recognised by professional artists. ChoreoMaster has successfully produced many hours of dance assets for several projects in Netease Games and to our best knowledge, is the first production-ready tool for this purpose in the industry.

ACKNOWLEDGEMENTS

We thank the reviewers for their constructive comments, and the artists for giving their time and great expertise to help improve our system. This work was supported by the National Key Technology RD Program (Project Number 2017YFB1002604), the National Natural Science Foundation of China (Project Numbers 61772298, 61521002), and the Research Grant of Beijing Higher Institution Engineering Research Center.

REFERENCES

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.

Okan Arikan and David A Forsyth. 2002. Interactive motion generation from examples. *ACM Trans. Graph.* 21, 3 (2002), 483–490.

Alexander Berman and Valencia James. 2015. Kinetic imaginations: exploring the possibilities of combining AI and dance. In *Proc. of IJCAI*.

Sebastian Böck and Gerhard Widmer. 2013. Maximum filter vibrato suppression for onset detection. In *In Proc. of the Int. Conf. on DAFx. (2013)*, Vol. 7.

Keunwoo Choi, György Fazekas, Mark Sandler, and Kyunghyun Cho. 2017. Convolutional recurrent neural networks for music classification. In *In Proc. of ICASSP*. IEEE, 2392–2396.

Abe Davis and Maneesh Agrawala. 2018. Visual Rhythm and Beat. *ACM Trans. Graph.* 37, 4, Article 122 (July 2018), 11 pages.

Yinglin Duan, Tianyang Shi, Zhengxia Zou, Jia Qin, Yifei Zhao, Yi Yuan, Jie Hou, Xiang Wen, and Changjie Fan. 2020. Semi-Supervised Learning for In-Game Expert-Level Music-to-Dance Translation. *arXiv preprint arXiv:2009.12763* (2020).

Rukun Fan, Songhua Xu, and Weidong Geng. 2011. Example-based automatic music-driven conventional dance motion synthesis. *IEEE TVCG* 18, 3 (2011), 501–515.

João P Ferreira, Thiago M Coutinho, Thiago L Gomes, José F Neto, Rafael Azevedo, Renato Martins, and Erickson R Nascimento. 2020. Learning to dance: A graph convolutional adversarial network to generate realistic dance motions from audio. *Computers & Graphics* 94 (2020), 11–21.

G David Forney. 1973. The Viterbi algorithm. *Proc. of the IEEE* 61, 3 (1973), 268–278.

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

Mikel Gainza. 2009. Automatic musical meter detection. In *2009 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 329–332.

Radek Grzeszczuk, Demetri Terzopoulos, and Geoffrey Hinton. 1998. Neuroanimator: Fast neural network emulation and control of physics-based models. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*. 9–20.

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), 6626–6637.

Daniel Holden, Jun Saito, and Taku Komura. 2016. A Deep Learning Framework for Character Motion Synthesis and Editing. *ACM Trans. Graph.* 35, 4, Article 138 (July 2016), 11 pages.

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

Tae-hoon Kim, Sang Il Park, and Sung Yong Shin. 2003. Rhythmic-Motion Synthesis Based on Motion-Beat Analysis. *ACM Trans. Graph.* 22, 3 (July 2003), 392–401.

Lucas Kovar, Michael Gleicher, and Frédéric Pighin. 2002. Motion Graphs. *ACM Trans. Graph.* 21, 3 (July 2002), 473–482.

Alexis Lamouret and Michiel van de Panne. 1996. Motion Synthesis By Example. In *Computer Animation and Simulation '96*, Ronan Boulic and Gerard Hégron (Eds.). Springer Vienna, Vienna, 199–212.

Hsin-Ying Lee, Xiaodong Yang, Ming-Yu Liu, Ting-Chun Wang, Yu-Ding Lu, Ming-Hsuan Yang, and Jan Kautz. 2019. Dancing to Music. In *Advances in NIPS*. 3581–3591.

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.

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.

Sung-Hee Lee and Demetri Terzopoulos. 2006. Heads up! Biomechanical Modeling and Neuromuscular Control of the Neck. *ACM Trans. Graph.* 25, 3, 1188–1198.

Ruilong Li, Shan Yang, David A. Ross, and Angjoo Kanazawa. 2021. Learn to Dance with AIST++: Music Conditioned 3D Dance Generation. arXiv:cs.CV/2101.08779

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.

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.

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

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.

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

Xuanchi Ren, Haoran Li, Zijian Huang, and Qifeng Chen. 2020. Self-supervised Dance Video Synthesis Conditioned on Music. In *In Proc. of the 28th ACM MM*. 46–54.

Joan Serra, Meinard Müller, Peter Grosche, and Josep Lluis Arcos. 2012. Unsupervised detection of music boundaries by time series structure features. In *AAAI*.

Joan Serra, Meinard Müller, Peter Grosche, and Josep Ll Arcos. 2014. Unsupervised music structure annotation by time series structure features and segment similarity. *IEEE TMM* 16, 5 (2014), 1229–1240.

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.

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

Guofei Sun, Yongkang Wong, Zhiyong Cheng, Mohan S Kankanhalli, Weidong Geng, and Xiangdong Li. 2020. DeepDance: Music-to-Dance Motion Choreography with Adversarial Learning. *IEEE TMM* (2020).

Taoran Tang, Jia Jia, and Hanyang Mao. 2018. Dance with melody: An lstm-autoencoder approach to music-oriented dance synthesis. In *Proc. of the ACM MM*. 1598–1606.

Junyuan Xie, Ross Girshick, and Ali Farhadi. 2016. Unsupervised deep embedding for clustering analysis. In *International conference on machine learning*. 478–487.

Sijie Yan, Zhizhong Li, Yuanjun Xiong, Huahan Yan, and Dahua Lin. 2019. Convolutional sequence generation for skeleton-based action synthesis. In *Proc. of the IEEE ICCV*. 4394–4402.

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.

Zijie Ye, Haozhe Wu, Jia Jia, Yaohua Bu, Wei Chen, Fanbo Meng, and Yanfeng Wang. 2020. *ChoreoNet: Towards Music to Dance Synthesis with Choreographic Action Unit*. Association for Computing Machinery, New York, NY, USA, 744–752.

Wenlin Zhuang, Congyi Wang, Siyu Xia, Jinxiang Chai, and Yangang Wang. 2020. Music2Dance: DanceNet for Music-driven Dance Generation. arXiv:cs.CV/2002.03761

A NETWORK ARCHITECTURES

In this section, we describe the details of our network architectures. All tensor sizes for each layer in our networks are shown in Tables 7 and 8.

Table 7. Network architecture of E_M and E_D

E_M		E_D	
Layer	Output Dim	Layer	Output Dim
Input	(1,96,800)	Input	(3,18,240)
MCB1	(64,48,400)	DCB1	(64,18,120)
MCB2	(128,16,200)	DCB2	(128,12,100)
MCB3	(128,4,50)	DCB3	(128,6,50)
MCB4	(128,1,25)	DCB4	(128,1,25)
GRU1	(256,25)	GRU1	(256,25)
GRU2	(128,25)	GRU2	(128,25)
Dense	(32)	Dense	(32)
Output	(32)	Output	(32)

Table 8. Network architecture of E_{MR} , E_{DR} and E_{RS}

	E_{MR}	E_{DR}		E_{RS}
Layer	Output Dim	Output Dim	Layer	Output Dim
Input	(2,200)	(5,60)	Input	(64)
RCB1	(64,50)	(64,30)	Dense	(128)
RCB2	(128,10)	(128,10)	Dense	(128)
Dense	(32)		Dense	(13)
Output	(32)		Output	(13)