# Music-Driven Choreography Based on Music Feature Clusters and Dynamic Programming

Shuhong Lin, Moshe Zukerman, *Life Fellow, IEEE*, Hong Yan, *Fellow, IEEE*

***Abstract*—Generating choreography from music poses a significant challenge. Conventional dance generation methods are limited by only being able to match specific dance movements to music with corresponding rhythms, restricting the utilization of existing dance sequences. To address this limitation, we propose a method that generates a label, based on a probability distribution function derived from music features, that can be applied to music segments of varying lengths. By using the Kullback-Leibler divergence, we assess the similarity between music segments based on these labels. To ensure adaptability to different musical rhythms, we employ a cubic spline method to represent dance movements. This approach allows us to control the speed of a dance sequence by resampling it, enabling adaptation to varying rhythms based on the tempo of newly input music. To evaluate the effectiveness of our method, we compared the dances generated by our approach with those generated by other neural network-based and conventional methods. Quantitative evaluations demonstrated that our method outperforms these alternatives in terms of dance quality and fidelity.**

***Index Terms*—Choreography, Music-driven Dance, Dynamic Programming, Cubic Spline**

## I. Introduction

For centuries dancing has played a significant role in human entertainment and culture as an artistic expression. Dancers on a stage may present a harmonious visual beauty by swaying their bodies according to the rhythm of the background music, so that participants, both dancers and spectators, enjoy the performance. Because of this feature, dancing is still popular even when people enter a virtual world. However, it is costly and challenging to create a virtual dance that looks natural based on a given piece of music.

Producers of virtual reality and game applications that involve virtual dancing often incur significant costs in employing professional choreographers and hiring motion-capture equipment. To achieve cost savings researchers have considered the development of computer-based choreography by using existing motion data as important [1]–[3]. The main challenge arises from the relationship between music and human motion. There have been several approaches to address this.

Neural network-based approaches are currently popular. The encoder-decoder framework [4], which is used successfully in cross-model learning, such as natural language processing and time series prediction, has also been applied to choreography generation [3]. In this method, the encoder projects the human pose and music parameters into a latent vector space. The decoder restores the vectors in the latent vector space to the human pose parameters. Existing publications, e.g., [5]–[7], have focused on finding a better network structure to improve the time consistency of the generated dance music and movement and the generation of human movement that appears more real and natural. The former requires that the generated dance and corresponding music can be synchronized on the beat, and the latter requires that the generated human posture can meet human dynamic constraints.

These approaches require a large amount of dance movement data. Public dance datasets obtained from professional dancers using motion capture equipment are highly accurate but rare [8] because of the significant cost.

Currently, most data sets are based on posture estimation of dance videos [9], [10], giving dance movement data that are often noisy. Results obtained by estimating real 3D positions from the 2D positions in the video will inevitably contain larger errors than those obtained by motion capture equipment [11]. These datasets often adopt different human skeletal models, so researchers can choose only one dataset as the training set of neural networks.

Unlike neural network-based approaches which suffer from freezing the action, approaches based on artificially designed features have better robustness. These approaches [1], [2] establish a relationship between dance movement and music segmentation. After building a music-motion database, a new dance can be produced according to the similarity of the features extracted from input music. Methods based on dynamic programming [12] and motion graph [1], [13] can be applied to organize candidate motions to avoid an exponential search space.

In this paper, we propose a new dance generation method based on dynamic programming and the similarity of musical features. In the pre-processing phase, we use the dance's background music as the label of the dance movements to build a database of dance segments. In the generation phase, the same process is used to generate labels for dance background music input by the user. Then candidate dance segments are selected according to the similarity of the labels of the input dance and those in the dance database. Finally, we organize the candidate dance motions by dynamic programming and generate transition motions between adjacent segments.

The key contributions of this paper are:

1) We generate a new music label based on the probability distribution function using music features. This label can

The work described in this paper was supported by the Hong Kong Innovation and Technology Commission (InnoHK CIMDA) and Research Grants Council of Hong Kong (CityU 11204821) *(Corresponding author: Shuhong Lin.)*

Shuhong Lin, Moshe Zukerman, and Hong Yan are with the Department of Electrical Engineering, and Center for Intelligent Multidimensional Data Analysis, City University of Hong Kong, Kowloon, Hong Kong SAR, PRC (e-mail: shuhonlin2-c@my.cityu.edu.hk; m.zu@cityu.edu.hk; h.yan@cityu.edu.hk).

be applied to music segments of different lengths, and by using the Kullback-Leibler divergence (KL divergence), we can determine the similarity of music segments based on this label. The music similarity measurement we have proposed can be applied to dance segments of varying lengths. This distinguishes our approach from existing methods for dance generation, as we are not constrained by the duration of the music segments. Consequently, it significantly broadens the range of dance segments that can be selected, offering more possibilities to create new dances.

2) To make the motion sequences adaptable to different music rhythms, we employ cubic splines [14] to represent dance movements. This allows us to control the speed of a dance sequence by resampling it, thereby adapting to different rhythms based on the tempo of newly input music.
3) We conducted a comparative analysis between our method and other neural network-based and conventional dance generation methods [2], [9], [10], [15]. This quantitative evaluation demonstrated the superiority of our method over these existing methods.

The remainder of this paper is organized as follows. Section II reviews existing methods from related publications. In Section III, we present the details of the new method to generate dance based on music. Section IV contains extensive quantitative results of its comparison against existing methods and a discussion.

## II. Related work

Existing methods for choreography generation can be divided into two categories: (1) conventional methods based on feature similarity, and (2) methods based on neural networks. We provide a new method for the first category and compare it with existing methods. Below, we review the relevant publications under these two categories.

### A. Conventional Methods Based on Feature Similarity

These follow a decomposition-combination process: (1) decompose the complete dance in the dance database into small segments, each of which consists of several frames; (2) extract musical and motion features from each segment; (3) select candidate segments, and combine them according to the similarity of their features. After Kim *et al.* [16] proposed a temporal correspondence between musical and kinematic beats, the segmentation of music data based on musical rhythm has been widely utilized in later music-driven motion generation studies [5], [6], [12] because rhythm is a key factor in how people perceive the speed of movement. Shiratori *et al.* [1], [13] applied synchronization of music and motion intensity to select candidate motion segments and utilize a motion graph [17] to generate a new dance. Chu *et al.* [18] proposed a new algorithm to extract the rhythm of music and motion and applied it to background music replacement. Ofli *et al.* [19] suggested a many-to-many relationship between dance movements and musical data and treated musical measures as the smallest compositional unit of music to divide the entire dance into segments. To address the many-to-many relationship between music and dance, two statistical models were proposed. One learns a many-to-one mapping from music to dance, while the other focuses on a one-to-many mapping from music to dance. Fan *et al.* [12] utilized correlation coefficients between musical and motion features as indicators of the music-motion relationship, and introduced a two-way dynamic programming procedure for simultaneous forward and backward searching processes. Such an approach can be used if both the initial pose and ending pose are known. Lee *et al.* [2] directly used music features to detect the boundary of the dance segments and measured the similarity of dance segments to aid artificial selection. Their main pipeline for dance generation is similar to our pipeline. They first establish a dance database and extract a music feature as the label of dance movement. The key difference between the two methods is that our method can generate vibrant dances automatically without any human intervention, while their method requires human selection to organize dance segments, as an aid for manual choreography. Specific methodological differences between the two approaches are their use of the average of music feature vectors within corresponding time intervals to generate motion data labels and our incorporation of the distribution function of music feature vectors within the corresponding time intervals. This allows our algorithm to consider finer details of dance features during the dance generation process. For segment labeling, Lee *et al.* [2] directly used the average over the frames within each segment so that each music segment is represented by a single feature vector.

The method we propose first performs clustering on all frames of the music in the database. Then, the distribution function of the clusters in each segment is used as the label for that segment. Based on our method, the data labels obtained can preserve more temporal information.

Conventional methods based on feature similarity decompose the entire dance into several segments and then reorganize these segments to generate a new dance according to the input music. Because dance movements directly come from ground-truth data, the generated dance movements are realistic.

However, the entire dance is divided into segments according to the rhythm which decides the length of the segment, and these methods can not adjust the length of segments. As a result, specific motion segments can only be synchronized with music that has specific rhythms. This may adversely affect efficiency when using the already limited dance motion data.

### B. Methods Based on Neural Networks

A Recurrent Neural Network (RNN) predicts the next frame according to its last output and its current hidden states, which makes it popular for motion generation [10], [20], [21].

In the conventional RNN training phase, the RNN is recursively given a sequence of ground truth motion data rather than its own training output. This method increases the speed of training but leads to error accumulation in the test phase because the RNN is unaware of the ground-truth data in this phase [21]. Zhou *et al.* [21] proposed an auto-conditioned Recurrent Neural Network (acRNN) to utilize its own output

periodically as the following input and gradually increase the proportion of its output to overcome the freezing of motion caused by error accumulation. Huang *et al.* [10] followed Zhou *et al.* [21] and used a dynamic auto-condition learning approach to alleviate error accumulation by the recurrent neural network so that their neural network was able to generate a long-term dance movement without freezing motion.

After Vaswani *et al.* [22] proposed a new neural network model, named transformer, and achieved success in sequence-to-sequence generation tasks in Natural Language Processing (NLP), the transformer became a popular method in time series prediction. It may also be applied to human motion prediction and generation. Li *et al.* [9] proposed the largest public dance-music paired dataset, AIST++. In existing studies on dance generation [10], [23], the specific dance motion is usually paired with unique music. It is also possible to match several different dance motions with the same music in the dance dataset, AIST++. Li *et al.* [9] proposed the Full Attention Cross-Modal Transformer (FACT) which uses a short motion seed and musical data to synthesize future dance motions so that the network can learn the many-to-many relationships between musical data and dance. FACT uses two transformers to encode music features and human pose features separately to two embedding vectors and then concatenates them as the input of the cross-model transformer. To address motion freezing, the cross-model transformer predicts several frames in the future rather than just single frames. Zhang *et al.* [24] first extract the musical rhythm and melody and use a musical style classifier. Their network uses musical style, melody and rhythm as control signals. They [25] proposed a new concept, the dance melody line, as a control signal. The dance melody line is extracted from dance motion directly and shared by different dance motions. Li *et al.* [15] designed a 3D-pose Vector Quantised-Variational AutoEncoder (VQ-VAE) to simulate a codebook, using elements in this codebook to present any dance motion pieces. To present a wide range of dance movements based on a limited dance data set, they trained two 3D-pose VQ-VAEs, for the upper and the lower body, separately. To address the one-to-many relationship between music and dance, Sun *et al.* [26] proposed an adversarial learning framework based on a generative adversarial network (GAN).

The above studies usually generate the dance motion frame by frame. The dance is regarded as a sequence of poses so that the neural networks are trained to learn the relationships between poses. However, a short dance combines hundreds of poses. So this strategy makes it easier for errors to accumulate and makes the training process of neural networks more time-consuming. Ye *et al.* [7] considered the work process of human choreographers who treat a dance as a combination of choreographic action units (CAU) which are indivisible clips of dance movements. The unchanged CAU allows neural networks to ignore all the minor details of the whole dance and use fewer steps to generate the dance. This strategy reduces error accumulation and computational time. However, the CAU is a more sophisticated concept and determining a CAU is the most crucial step in such a method. Current methods often collect CAUs designed by professional choreographers. Another challenge of this method is to generate the transitions between CAUs. Ye *et al.* [7] proposed a U-shape neural network inspired by image inpainting [27] to inpaint the transition gap between adjacent CAUs. Lee *et al.* [23] proposed a decomposition-to-composition framework consisting of a Variational Autoencoder (VAE) and a Generative Adversarial Network (GAN) to model CAUs and to learn how to organize them. Li *et al.* [8] proposed a two-stage approach. The first stage selects the keyframes and generates poses in between them, and the second stage generates the motion curve with a similar effect between two adjacent keyframes. Chen *et al.* [6] designed an embedding module to capture music-dance connections. Aristidou *et al.* [5] used the dance units and considered the distribution of the motion motif to keep the dance style consistent.

## III. Proposed method

In this section, we describe in detail the process of our dance synthesis method. The entire process consists of two main stages. The first stage is to establish a dance motion database consisting of a sufficiently large number of dance segments with their corresponding musical labels, and the second stage is to synthesize a new dance according to the particular audio signal input digital data provided by the user.

### A. Dance Dataset

We used a dance dataset called AIST++ [9]. To the best of our knowledge, AIST++ is the largest human dance motion dataset. It contains 1,408 sequences of 3D human dance motions paired with corresponding music data. Each frame of the sequence is represented by a human skeleton which is a popular representation of the human pose. It consists of bones and joints. Each joint represents a key point of the human body, and each bone represents a link between two different joints (Fig. 2). There are two kinds of human skeletons in AIST++: (1) Common Objects In Context (COCO) format with 17 joints [28], and (2) Skinned Multi-Person Linear Model (SMPL) format with 24 joints [29].

Li *et al.* [9] divided AIST++ into several non-overlapping parts for cross-modal analysis between human motion and music data. They chose 998 sequences from the dataset and divided them into a training and a testing dataset. We only used the AIST++ training dataset to establish the dance motion database.

A dance motion sequence in AIST++ is a time series of human poses and denoted as $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_T\}$. Each human pose is represented by a vector $\mathbf{x}_t = (\mathbf{p}_{t,0}, \mathbf{q}_{t,0}, \mathbf{q}_{t,1}, \cdots, \mathbf{q}_{t,23})$ which describes an SMPL skeleton containing a root node and 23 joint nodes as shown in Fig. 2. The root node of the SMPL skeleton is represented by a 3-dimensional coordinate $\mathbf{p}_{t,0} \in \mathbb{R}^3$ and a unit quaternion $\mathbf{q}_{t,0} \in \mathbb{R}^4$. The 3-dimensional coordinate and the unit quaternion represent the position of the skeleton and the global rotation of the skeleton, respectively. The structure of the SMPL skeleton and the correspondence between SMPL joint names and indices are shown in Fig. 2. For each joint node, we used a unit quaternion $\mathbf{q}_{t,j} \in \mathbb{R}^4$ to indicate its relative

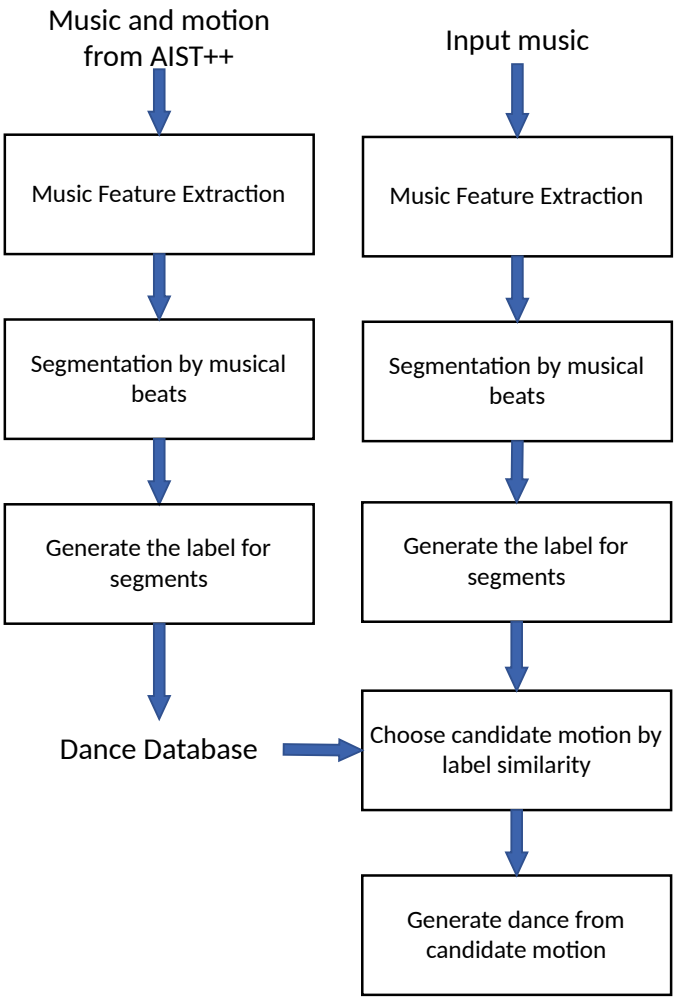



Fig. 1: The overview of our method.

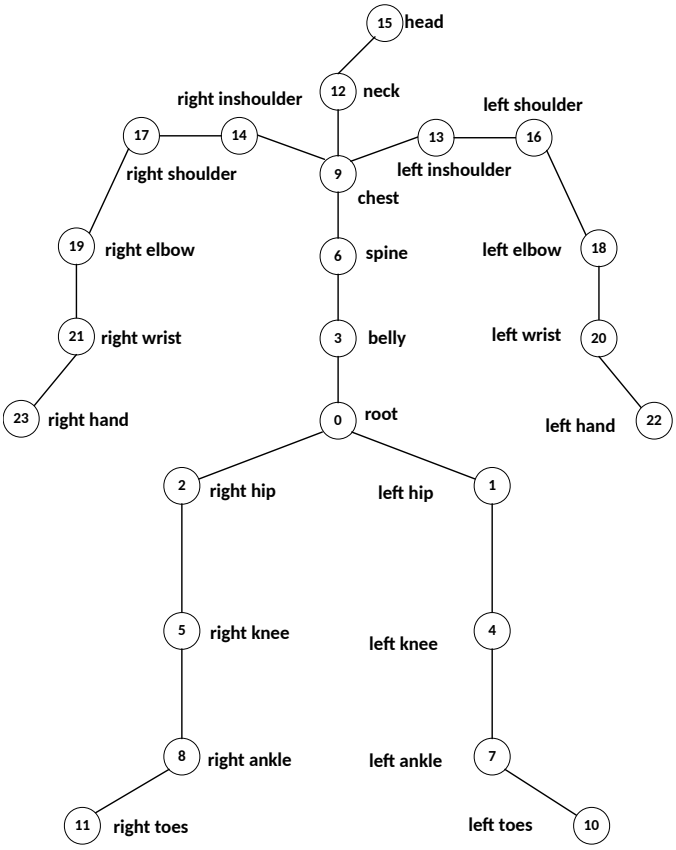



Fig. 2: The structure of the SMPL skeleton and the correspondence between SMPL joint names and indices.

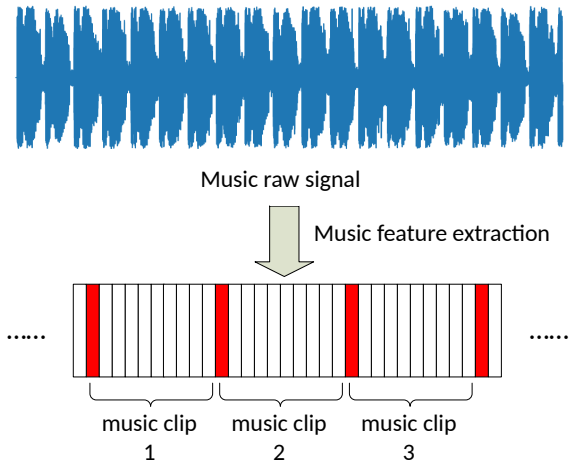



Fig. 3: The feature vectors are extracted from audio signals and synchronized with the dance motion. The red blocks indicate the beats.

rotation with respect to its parent joint. To represent each human pose in our motion sequences, we created a vector by concatenating a 3-dimensional position vector and 24 4-dimensional quaternions. This results in a vector with a total dimensionality of $3 + 24 \times 4 = 99$ for each human pose.

### *B. Dance Segmentation*

Here we describe how we divide the entire dance sequence into segments.

According to [16], there is a strong correlation between musical and kinematic beats along the time axis. We assume that the original dance in the database is well synchronized with the corresponding music, and dancers will start or terminate actions at the time of musical beats. Thus, musical beats are clues to detect the boundaries of a dance segment. As shown in Fig. 3, we extracted the raw musical features from the raw music signal and generated the same number of music feature vectors as the number of poses. In Fig. 3, each rectangle represents a music feature vector and the red blocks represent music feature vectors corresponding to the musical beats. We used these musical beats as boundaries to divide the entire dance into several short segments. These dance segments have an advantage over poses as units for choreography because the kinematic beats can always synchronize with the musical beats within the same dance segment.

### *C. Music Features*

Previous studies have used many musical features to analyze musical structure, music generation, and choreography generation. We selected 20-dimensional Mel-frequency cepstral

| Music feature | Dimension | Feature function |
|---|---|---|
| Onset Strength Envelope | 1 | The amplitude variations over time in an audio signal. |
| Mel-Frequency Cepstral Coefficients | 20 | Simulate the auditory characteristics of the human ear. |
| Chroma Energy Normalized | 12 | Quantify the distribution of musical pitch classes. |
| Peak of Onset Strength Envelope | 1 | The presence of peaks or prominent local maxima in a given spectrogram. |

TABLE I: The music features we used.

coefficients (MFCC), 12-dimensional chroma, a 1-dimensional envelope and a 1-dimensional one-hot peak as the raw music features. The details are shown in Table I.

The selection of these features is based on their ability to capture different aspects of music, including spectral content, pitch information, dynamics, and specific frequency components. By combining these features, a comprehensive representation of each audio signal can be obtained. Algorithms for tracking beats in audio signals have been well-studied. In our implementation, we used the realization of Librosa [30] which is a widely used library in audio analysis and processing.

### *D. Segment Label Synthesis*

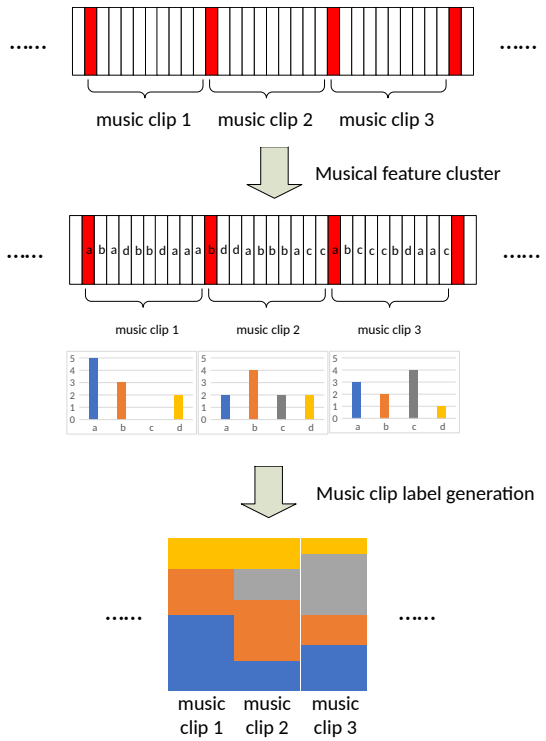



Fig. 4: The music labels of dance segments based on musical beats and the music feature clustering.

Existing publications usually use the musical and kinematic features together as the label of the dance segments, e.g., [12], [13], [19]. There are two representations for pose similarity measures: the positions and rotations of joints. The problem with the position of joints is how to keep spatial invariance when Euclidean distance is used to measure the similarity of postures. For example, the Euclidean distance between two similar poses facing different directions may be significant. The local rotation of joints is a better presentation than the 3D position if we want to keep the spatial invariance because similar poses always have similar local rotations of joints, even if they face different directions. However, Pavllo *et al.* [31] pointed out that it is crucial to consider prediction errors (caused by differences between the ground-truth poses and the predicted poses) in different joints, and it is difficult to determine the weight of each joint in the similarity measure. For example, the same rotational difference has a more significant effect in spinal joints than in limb joints. To address this issue, we used only music features to measure the similarity between segments because the audio signal has less noise and is more robust. In the AIST++ dataset [9], each dance genre has several background music tracks and there exists a one-to-many relationship between background music and dance motion in the AIST++ dataset. [9]. Thus, the music features are enough for the dance genre and motion selections.

Dance segments obtained after segmentation have different lengths because the rhythms of the background music in the AIST++ are different. Drawing on [32], we proposed a music-feature-based label that can be used to measure the similarity between music segments of varying lengths. As shown in Fig. 4, we first extracted the raw music features from the audio signal, and then principal component analysis (PCA) was applied for dimensionality reduction. Next, we used the K-means clustering algorithm to divide the music feature vectors into clusters. In Fig. 4, we used different letters as labels to represent the different clusters. To measure the musical similarity of dance segments, we calculated the discrete probability distribution of each segment and used the average Kullback–Leibler (KL) divergence to measure the similarity of the two music pieces. For two discrete probability distributions $Q$ and $P$, the distance between them is obtained by

$$Dis(P,Q) = \frac{1}{2}\left(D_{KL}\left(P\|Q\right) + D_{KL}\left(Q\|P\right)\right) \tag{1}$$

where $D_{KL}\left(P\|Q\right)$ is the KL divergence from $Q$ to $P$:

$$D_{KL}\left(P\|Q\right) = \sum_{x=1}^{K} P\left(x\right) \log\left(\frac{P\left(x\right)}{Q\left(x\right)}\right) \tag{2}$$

and $P\left(x\right)$ and $Q\left(x\right)$ are the discrete probability distribution functions (PDFs) of the labels of the input music and the music from the database (ground truth), respectively. $K$ is the number of clusters and is set by the user. This music-feature-based label was used to analyze the structure of music. As shown in Fig. 5, we used a piece of music in AIST++ to generate the self-distance matrix. Both vertical and horizontal axes are time indexes. Dark colors represent high similarity and light colors represent low similarity. The pattern of diagonal lines in the self-distance matrix represents repetitive parts of the music data.

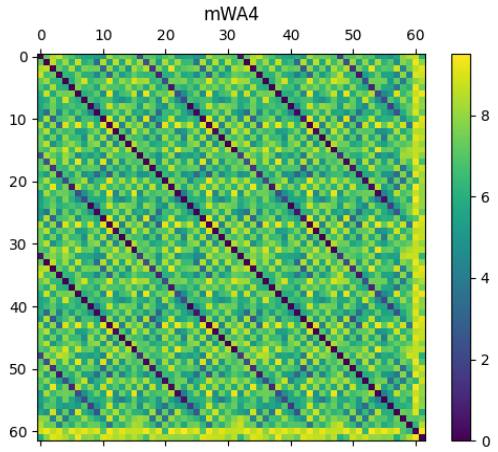



Fig. 5: A self-distance matrix corresponding to background music called "mWA4" in the AIST++.

As shown in Fig. 5, the self-distance matrix of the sample music can describe its structure.

### E. Motion Features

We utilized the 3-dimensional position for segment connection and transition motion generation.

Since we chose the SMPL skeleton with 24 joints as our human skeleton and the SMPL skeleton is described by joint rotation, we need to obtain the 3D positions of all joints by forward kinematics [33]. Because the length of the bones that connect adjacent joints is assumed to be constant, the pose of the human body is determined by the global position of the root joint and the orientation of all joints.

The motion capture data in AIST++ comes from different dancers with different body shapes. To facilitate the motion analysis, we need to adjust the lengths of the skeletons for different dancers so that the same bones on different dancers have the same length. we calculated the average lengths of the bones from different dancers and then adjusted the bone lengths to their average lengths. After we obtained the information on the lengths of all of the bones and the position of the root joint determining the global position of the skeleton, we calculated the positions of the remaining 23 joints using forward kinematics, which employs the rotational data of the joints relative to each other to determine their respective positions.

Because a person may have similar poses in different directions, we used the joints' relative position with respect to the root joint and applied the method of [34] to connect the adjacent motion segments to obtain the total 3D-position distance, denoted $D_{pose}$, between two human poses, $\mathbf{p}_a$ and $\mathbf{p}_b$, as:

$$D_{pose}\left(\mathbf{p}_a, \mathbf{p}_b\right) = \left\|\mathbf{p}_a - \mathbf{T}_{\theta,x_o,z_o}\mathbf{p}_b\right\| \tag{3}$$

where

$$\theta = \arctan\left(\frac{\sum_j w_j\left(x_{a,j}z_{b,j} - x_{b,j}z_{a,j}\right) - \left(\bar{x_a}\bar{z_b} - \bar{x_b}\bar{z_a}\right)}{\sum_j w_j\left(x_{a,i}x_{b,j} + z_{b,j}z_{a,j}\right) - \left(\bar{x}\bar{x_b} + \bar{z_b}\bar{z_a}\right)}\right) \tag{4}$$

$$x_o = \bar{x_a} - \bar{x_b}\cos\theta - \bar{z_b}\sin\theta \tag{5}$$

$$z_o = \bar{z_a} + \bar{x_b}\sin\theta - \bar{z_b}\cos\theta \tag{6}$$

where $\bar{x}_a = \sum_j w_j x_{a,j}$ and $\sum_j w_j = 1$. The linear transformation $\mathbf{T}_{\theta,x_o,z_o}$ rotates a human pose $\mathbf{p}_b$ about the vertical axis by $\theta$ and translates it by $(x_o, 0, z_o)$ so that the initial pose of the next motion segment can be adjusted to a position and orientation as similar as possible to the current pose.

The 3D positions of all joints are not enough for character animation generation. For instance, the 3D position alone cannot fully capture the twisting motion of a bone. Thus, the final result of the generated dance by our method is described by the global position of the root joint and the orientation of all joints.

### F. Cubic Spline

To make the same dance segment fit music with different rhythms, we transferred the discrete pose sequence to a continuous motion curve. Then we can modify the length of the motion segments and the movement speed by resampling to address the insufficiency of the limited data. The motion features contain the local rotation of 24 joints and their global position. Accordingly, we obtained 25 cubic splines as motion curves for each segment. For the trajectory of the root joint, we used a cubic Hermite spline. On a given time interval, the position of the root joint $p(t)$ can be expressed by a polynomial, as follows.

$$\begin{aligned} p(t) =& h_{00}(t)p_0 + h_{10}(t)(x_1 - x_0)m_0 + h_{10}(t)p_1 \\ &+ h_{11}(t)(x_1 - x_0)m_1 \end{aligned} \tag{7}$$

where $t = \frac{x - x_0}{x_1 - x_0}$. The $m_0$ and $m_1$ are the tangents of $p(t)$ at its endpoints $p(x_0)$ and $p(x_1)$, respectively.

$$h_{00}(t) = 2t^3 - 3t^2 + 1 \tag{8}$$

$$h_{01}(t) = t^3 - 2t^2 + t \tag{9}$$

$$h_{10}(t) = -2t^3 + 3t^2 \tag{10}$$

$$h_{11}(t) = t^3 - t^2. \tag{11}$$

One advantage of the cubic Hermite spline is that it can keep the velocity and the position in the keyframes and produce smooth curves. For the rotation, we used the quaternion cubic spline which can produce a smooth curve between two poses and keep the angular velocity of the endpoints. The main idea of the quaternion Cubic Spline is based on the Cubic Spline, and both are widely used in animation.

The cubic spline was also applied in motion transition. After we determined two adjacent dance segments, we used the last five frames from the last segment and the first five frames from the next segment to generate a three-frame transition motion between the two segments.

### G. New Dance Generation

When the user inputs new music data, we followed the same steps to divide it into segments and used the same parameters for the PCA and K-means clustering to generate labels for these segments, as shown in Fig. 4. To speed up the generation, we compared the probability distribution function of the complete input music data with that of the music data (ground truth) in the training dataset and chose the $n$-closest music data, where $n$ is set by the user. We only selected the segments from these music data as the candidate segments so that we did not need to search for candidate segments among the entire training dataset.

Having selected the $n$-closest music data, we applied dynamic programming to select candidate dance segments by minimizing a cost function that contains two terms: the music distance and the pose distance. The optimization problem is formulated as follows.

$$\min\left(\sum_{i=2}^{N}\left[D_{pose}\left(\mathbf{p}_{i-1}^{\text{end}},\mathbf{p}_{i}^{\text{start}}\right)+\lambda\cdot Dis\left(P_{0,i},Q_{0,i}\right)\right]+D_{pose}\left(\mathbf{p}^{\text{initial}},\mathbf{p}_{1}^{\text{start}}\right)\right), \tag{12}$$

where $\mathbf{p}^{\text{initial}}$ is the initial pose, and $\mathbf{p}_i^{\text{strart}}$ and $\mathbf{p}_i^{\text{end}}$ are the first and last poses of $i$th segment, respectively. The variable $P_{0,i}$ is the music label's PDF of the first $i$ selected segments, and $Q_{0,i}$ is the PDF of the first $i$ music segments of the input music. When we determined the $i$-th candidate segment, we did not use the pairwise KL divergence between the PDFs of the $i$-th input music clips and the $i$-th candidate segment from the database. Instead, we used the PDFs on the time interval from the first beat to the $i$-th beat as shown in Fig. 6.

The latter approach is more appropriate in our case because the former approach will make the dynamic programming process focus on matching the music features of the current segment selection, while the latter considers matching the music features of the entire dance sequence.

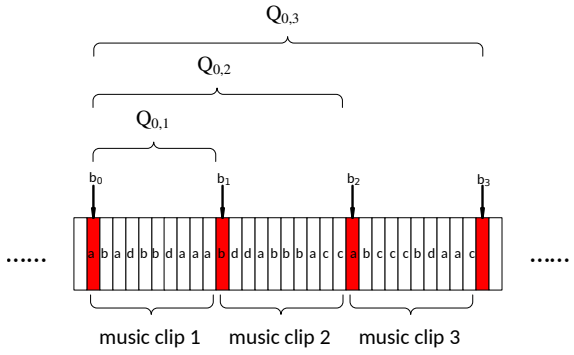



Fig. 6: Derivation of the probability distribution functions.

In Fig. 6, we illustrate how we derived the PDFs for the Dynamic Programming procedures. The symbol $Q_{0,i}$ is the PDF of the first $i$ music segments of the input music. The symbols $b_0$, $b_1$, etc. represent the locations of the music beats. The clusters corresponding to the music features of each frame are represented by different lowercase letters. Because we use the music beats as the boundaries of the segment, the locations of these beats are important to calculate the PDF as it is based on the music features from the start of the dance up to the current beats, rather than just the music features between the current and previous beats.

We applied Equation (3) to obtain the difference, $D_{pose}$, between the final pose of the last segment and the first pose of the next segment, instead of the Euclidean distance of the coordinates of the corresponding joints, because the Euclidean distance may include a difference in a position that is not related to the change of pose. For example, if there is no change in the pose, i.e., $D_{pose} = 0$, there can be a change in position, so the Euclidean distance will not be zero.

## IV. Experiments

We compared our method to three neural network-based state-of-the-art methods: DanceRevolution [10], FACT [9] and Bailando [15]. DanceRevolution [10] uses its own dance dataset. Thus, we retrained DanceRevolution [35] on the AIST++ dataset to achieve a fair comparison with our method. We used the same parameters for DanceRevolution as used by the authors [10]. Li *et al.* [9] provided the AIST++ dataset and proposed FACT which is a transformer-based method. Bailando [15] proposed by Li *et al.* is the latest work on music-driven dance generation based on the AIST++ dataset. For FACT [9] and Bailando [15], we downloaded the pre-trained model from the authors' GitHub pages [36], [37].

### A. Implementation Details

The AIST++ dataset provides 992 pieces of dance paired with music. There are 952 pieces of dance motion for training, 20 pieces for testing and 20 pieces for evaluation. Including the data in the test group, Li *et al.* [9] also used randomly paired music and motion seeds. Because users may use randomly paired music and dance motions as input in practice, the test dataset we used also contains the randomly paired music and dance motions. We used the same dance movements and music combinations that Li *et al.* used. FACT uses a 10-second motion seed from the ground-truth dance motion so that they can generate different dances based on the same music and different motion seeds. Similarly, we used the first frame from the ground-truth dance motion as the initial pose of our method. The method described in Lee *et al.*'s paper [2] aims to assist mutual choreography. In their study, they employed two methods: random generation (selecting dance movements recommended according to music features by their algorithm) and manual choreography based on the algorithm's recommendations. In our implementation of the method of [2], we replicated exactly the first method based on random selection as described in [2], but for the implementation of the second method of [2], we conducted a manual selection of dance movements selection as in [2].

DanceRevolution proposed by Huang *et al.* [10] uses a skeleton with 25 joints, and the human poses are represented by the position of the joints. For a fair comparison, we adjusted the number of output action parameters and retrained the

model using the AIST++ training dataset rather than using the pre-trained model provided by the authors. The dances in AIST++ have varying lengths, while all the dance clips in the DanceRevolution dataset are one minute long. Thus, we take 10-second overlapping clips every two seconds to make all the training dance clips have equal lengths to facilitate network training. DanceRevolution always begins each dance with the same initial pose, regardless of the specific dance performed. Another difference between our method and DanceRevolution is that the music from the DanceRevolution dataset contains lyrics while ours does not. The hyperparameters used in our neural network training were the same as those in the original paper [10].

### B. Evaluation Metrics

We measured the performance of our method from three perspectives by five metrics.

*1) Motion quality:* The Fréchet Inception Distances (FID) [38] have been popular for measuring the similarity between ground-truth and synthetic images using generative models (e.g., a GAN). In this paper, as in [8], [9], we used FID to assess the similarity between features of the generated dance and all ground-truth dances in AIST++. The lower the FID, the more similar the synthetic dance is to the ground-truth dance.

Below, we provide the details of the FID calculation to measure the distance between two features extracted from two datasets. These details are similar to an equivalent FID calculation presented in [38], where the FID was introduced as an evaluation metric for synthetic data, and it is included here for completeness.

$$\begin{aligned} FID\left(S_{\text{gt}}, S_{\text{gen}}\right) =& \left\|\boldsymbol{\mu}_{\text{gt}} - \boldsymbol{\mu}_{\text{gen}}\right\|_2^2 \\ &+ \text{tr}\left(\boldsymbol{\Sigma}_{\text{gt}} + \boldsymbol{\Sigma}_{\text{gen}} - 2\left(\boldsymbol{\Sigma}_{\text{gt}}\boldsymbol{\Sigma}_{\text{gen}}\right)^{\frac{1}{2}}\right) \end{aligned} \tag{13}$$

where $S_{\text{gt}}$, $S_{\text{gen}}$ are the sets of features extracted from the ground-truth and generated datasets and have mean vectors $\boldsymbol{\mu}_{\text{gt}}$, $\boldsymbol{\mu}_{\text{gen}}$ and covariance matrices $\boldsymbol{\Sigma}_{\text{gt}}$, $\boldsymbol{\Sigma}_{\text{gen}}$, respectively. The trace of a square matrix is denoted by $\text{tr}(\cdot)$. The first term of Equation (13) is the squared Euclidean distance between the two mean vectors, $\boldsymbol{\mu}_{\text{gt}}$, $\boldsymbol{\mu}_{\text{gen}}$, and the second term is the trace of the following square matrix.

$$\left[\boldsymbol{\Sigma}_{\text{gt}} + \boldsymbol{\Sigma}_{\text{gen}} - 2\left(\boldsymbol{\Sigma}_{\text{gt}}\boldsymbol{\Sigma}_{\text{gen}}\right)^{\frac{1}{2}}\right].$$

Accordingly, the FID will be different depending on the features used. In this paper, we calculated two kinds of FID, denoted $FID_k$ and $FID_g$, that were obtained based on the distributions of the kinetic [39] and geometric features [40], respectively. The kinetic features represent the velocity of the motion, and the geometric features represent the geometric relative position between different joints. Both FIDs, $FID_k$ and $FID_g$, are widely used in research publications on human motion generation [8], [9], [15].

*2) Motion diversity:* To measure the diversity of the generated motion, we calculated the average Euclidean distances for kinetic and geometric features as $Dist_k$ and $Dist_g$, respectively. For each feature type, kinetic and geometric, we calculated the Euclidean distances between any two features for the generated dance motions and then averaged them to obtain our evaluation of motion diversity.

*3) Music-dance alignment:* As in previous studies, we calculated the average temporal distance between each music beat and its closest kinematic beat as follows.

$$\frac{1}{|B^m|}\sum_{t^m \in B^m} exp\left(-\frac{min_{t^d \in B^d}\left\|t^d - t^m\right\|^2}{2\sigma^2}\right) \tag{14}$$

where $B^m$ and $B^d$ are the sets of the times of music and kinematic beats, respectively. The parameter $\sigma$ was set to be 3.

### C. Results

The quantitative results on the AIST++ test set are shown in Table II. We compared our method with fourstate-of-the-art methods: the method proposed by Lee *et al.* [2], DanceRevolution [10], FACT [9] and Bailando [15]. Bailando, proposed by Li *et al.* [15], is the latest work on music-driven dance generation to our knowledge.

According to the comparison, our proposed approach consistently outperformed all other existing approaches in the vast majority of evaluations.

Our method performed significantly better than Lee *et al.*'s method [2] in terms of the similarity of kinetic and geometric features between the ground-truth and generated dances. The random selection shows a weak ability in dance generation because their method simply replicates the dance clips from the dance dataset. The human manual selection significantly improves the scores in all evaluation metrics, but Lee *et al.*'s method still suffers the same problems as manual selection. The lower score in music-dance alignment indicates our segmentation algorithm performs better than segmentation based on the novelty function used by Lee *et al.* Specifically, our method improved by 20.33 (58.93%) over the best-compared baseline model, FACT [9], on $FID_k$ that evaluates the similarity of the kinetic features between a ground-truth and generated dance by using kinetic features related to the velocity of all joints to calculate the similarity. Directly using the dance movements from the dataset does not yield a lower $FID_k$ score. The method proposed by Lee *et al.* also uses dance clips from the dataset to organize a new dance. Through experiments, We still find a high $FID_k$ score when manually selecting appropriate dance segments. Hence, replicating dance segments from the dataset alone does not yield satisfactory results. Previous studies [1], [9], [23] have shown a significant relationship between musical rhythm and joint velocity in dance. In our approach, we utilize cubic splines to regulate the speed of joint movements based on the background music's rhythm rather than just a replication from the dataset. For $FID_g$, we achieved a 1.09 (9.71%) improvement over Bailando which achieved the best performance among the three baseline methods. Lower $FID_k$ and $FID_g$ means the difference in the statistics of the kinetic and geometric features between the generated motion and the ground-truth motion is smaller. Thus, the generated motion

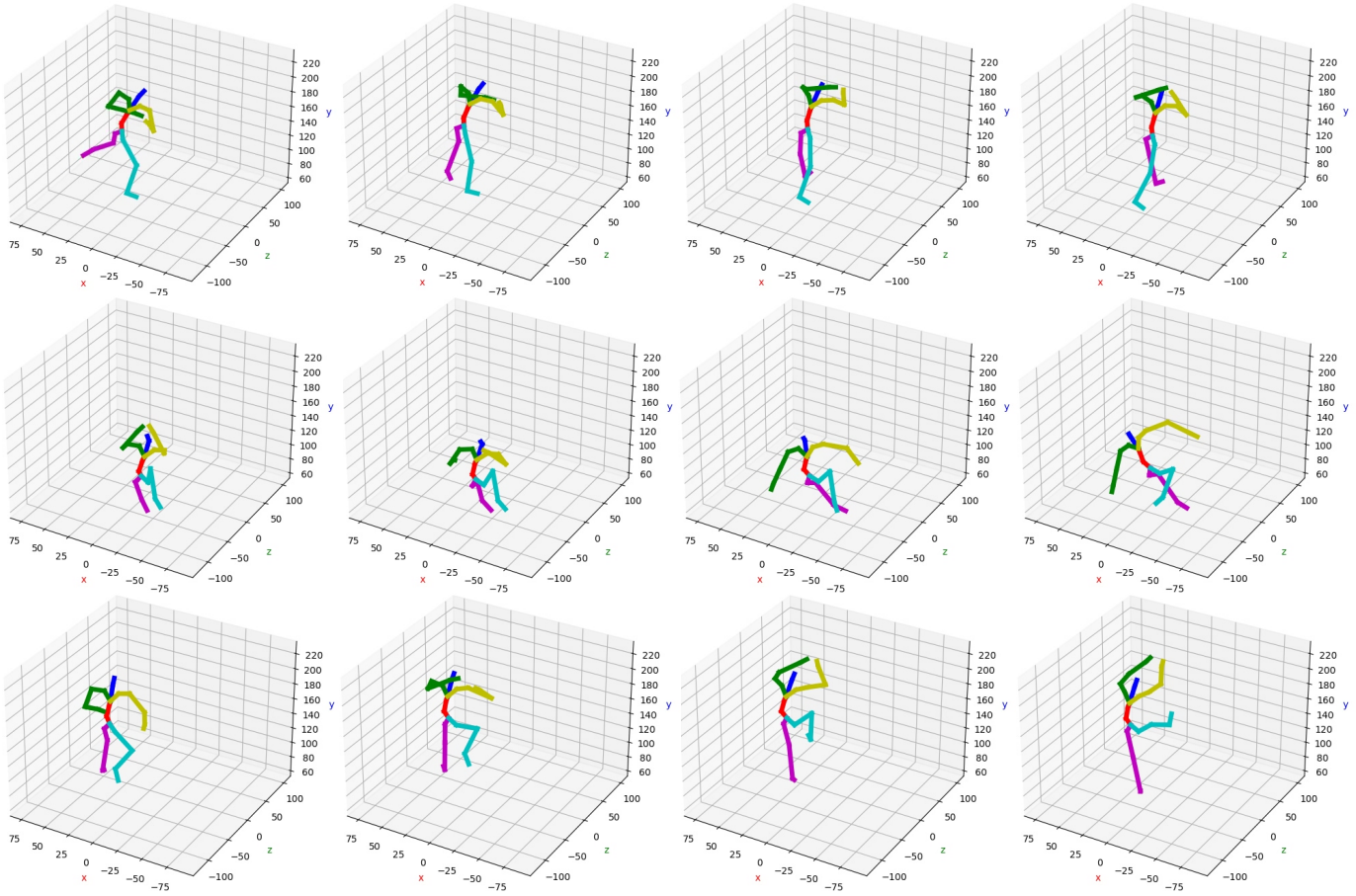


Fig. 7: Examples of dance motions generated by our method. Each row represents a different dance motion.

| | $FID_k \downarrow$ | $FID_g \downarrow$ | $Dist_k \uparrow$ | $Dist_g \uparrow$ | $BeatAlign \uparrow$ |
|---|---|---|---|---|---|
| AIST++ (Ground Truth) | | | 10.0212 | 7.3223 | 0.283 |
| Lee *et al.* (random) [2] | 497.3034 | 32.2086 | 22.8797 | 3.6424 | 0.202 |
| Lee *et al.* (manual) [2] | 68.1100 | 22.7618 | 11.9332 | 5.1348 | 0.215 |
| DanceRevolution [10] | 85.1973 | 44.9465 | 1.7390 | 4.2113 | 0.203 |
| FACT [9] | 34.4971 | 16.4088 | 7.9224 | 6.1689 | 0.217 |
| Bailando [15] | 62.3286 | 11.2768 | 9.5037 | **6.5164** | 0.194 |
| Ours | **14.1669** | **10.1839** | **9.8634** | 5.6391 | **0.263** |

TABLE II: Quantitative results on the AIST++ test set with random pairs of music and initial pose; ↑ means the larger score is better while ↓ means the lower score is better; bold values represent the best results for each column.

sequences of our method have a more similar distribution to the ground-truth motion sequences in AIST++.

In terms of motion diversity, we obtained a better result for $Dist_k$ (9.86 versus 9.50), while Bailando and FACT have better performance than our method in $Dist_g$ (5.64 versus 6.52). However, Bailando has a relatively high value for $FID_k$ and shows a lower score for music-dance alignment. The diversity of motion we generate is better than the baseline in the kinetic feature space. In the geometric feature space, the neural network-based approaches show better performance. The beat-alignment score in Table II shows better performance by our method than the baseline methods in synchronization between musical and kinematic beats. The third row is a kick action from the dance.

In Fig. 7, we present twelve frames to demonstrate our generated dance motion. Each row of four frames represents a motion from a generated dance. The animation comprises 60 frames per second and the presented frames were extracted every 10 frames and are separated by $\frac{1}{6}$ second time-periods. Different bones are distinguished by different colors. The first and second rows show two dance motions in a standing and a sitting state, respectively, while the motion in the third row is a kick, demonstrating our method can generate complex dance motions.

In Fig. 8, we compared a dance achieved by our method against dances generated by the following state-of-the-art methods: the method of Lee *et al.* [2], DanceRevolution [10], FACT [9] and Bailando [15]. The first row displays the ground-truth dance movement from the test dataset. Then, each row displays a short sequence of dance generated by a specific method, with each column aligned in time. The time interval between adjacent frames remains constant. Based on Fig. 8,

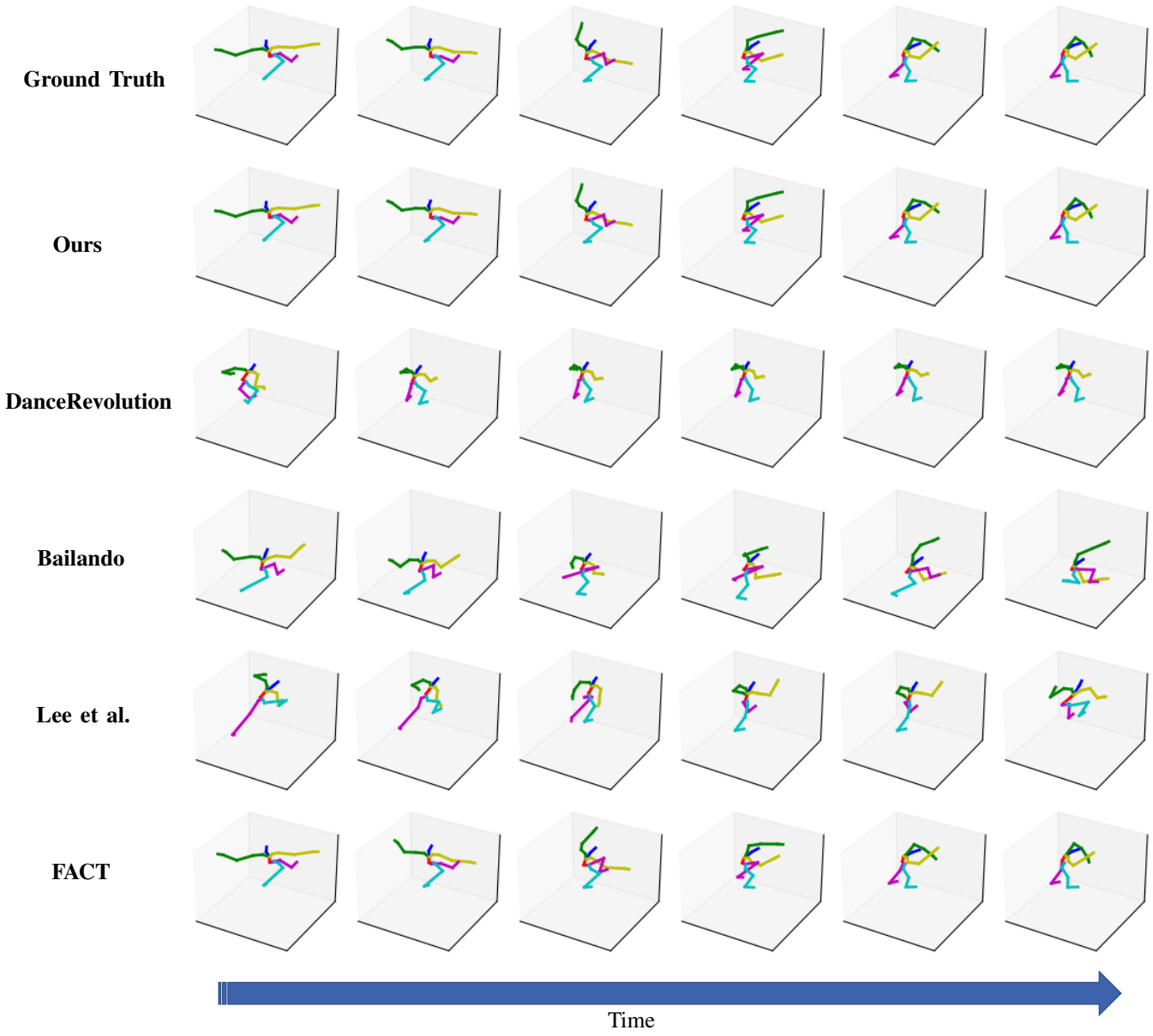



Fig. 8: Examples of dance motions generated by our method and other state-of-the-art methods.

we observe that we have achieved a significantly better performance relative to the conventional method proposed by Lee *et al.* and that FACT has achieved the best performance among the three neural network-based methods. Our method performs better than FACT in some of the metrics. For instance, in the third column, the generated pose of the left forearm using our method exhibits a closer resemblance to the real data than the pose generated by FACT.

To further compare to Bailando, we present the results of two other experiments. The first experiment was to generate a 20-second output, while the other one aimed to generate an entire dance. The results of the generated dance based on the first 20-second output are shown in Table III, and those on the entire dance are shown in Table IV. In Tables III and IV, we observe that our method performed significantly better than Bailando when the test dataset contains random pairs of music and initial poses. When the test dataset does not contain the random pairs of background music and initial poses, Bailando

| | $FID_k \downarrow$ | $FID_g \downarrow$ | $Dist_k \uparrow$ | $Dist_g \uparrow$ |
|---|---|---|---|---|
| Bailando* | 28.1616 | **9.6258** | 7.8373 | **6.3446** |
| Ours* | **12.1742** | 11.0592 | **8.7406** | 5.8964 |
| Bailando | 27.7354 | 9.5659 | 7.5485 | **6.1969** |
| Ours | **11.7145** | **8.3338** | **8.3129** | 5.8381 |

TABLE III: Comparison based on the first 20-seconds of generated dance. The symbol * indicates that the test dataset does not contain random pairs of music and initial poses, and ↑ indicates that larger is better while ↓ indicates that lower is better.

| | $FID_k \downarrow$ | $FID_g \downarrow$ | $Dist_k \uparrow$ | $Dist_g \uparrow$ |
|---|---|---|---|---|
| Bailando* | 64.8358 | **11.2058** | **9.8364** | **6.6346** |
| Ours* | **18.5848** | 12.6156 | 9.7680 | 5.4900 |
| Bailando | 62.3286 | 11.2768 | 9.5037 | **6.5164** |
| Ours | **14.1669** | **10.1839** | **9.8634** | 5.6391 |

TABLE IV: Comparison based on the complete generated dance. The symbols *, ↑ and ↓ have the same meaning as in Table III.

has a slightly better performance in about half of the metrics such as $FID_g$. The clear advantage of our method is in the case where the test dataset contains random pairs of music and initial poses. This demonstrates that our method is more robust, which is especially important because users may use their background music and initial poses as input, and we cannot assume that the pair of background music and initial poses they use is the same as in AIST++.

There is a one-to-many relationship between background music and dance movement in AIST++. Bailando may not have considered the impact of this relationship, which may be the reason for their lower score observed in Tables III and IV for the test dataset with random pairs of music and initial pose.

The value of $FID_k$ becomes relatively high in long-term dance generation as shown in Table IV, suggesting difficulty in capturing the rhythm of a longer dance. Bailando has achieved a better result for the short-term dance generation than the long-term dance generation. Our method can perform better when the user inputs a random pair of music and initial pose because our method considers the one-to-many relationship between background music and dance movement in AIST++.

| | $FID_k \downarrow$ | $FID_g \downarrow$ | $Dist_k \uparrow$ | $Dist_g \uparrow$ |
|---|---|---|---|---|
| Our method | **14.1669** | **10.1839** | **9.8634** | **5.6391** |
| Without chroma | 17.0012 | 12.1171 | 6.8134 | 5.1132 |
| Without MFCCs | 15.9165 | 14.6387 | 8.2509 | 5.3442 |

TABLE V: Ablation study of our method on musical features. The ↑ and ↓ have the same meaning as in Table III.

We conducted an ablation study to explore the influence of musical feature selection. In addition to using the musical features given in Table I, a combination without chroma, and a combination without MFCCs were tested as shown in Table V. The performance is best if both MFCCs and chroma are used as input musical features. All four evaluation metrics indicate a significant performance decline without either MFCCs or chroma. The MFCCs have a greater impact on $FID_k$, $Dis_k$, and $Dis_g$, while chroma has a greater impact on $FID_g$.

## V. Conclusions

We have proposed a new dance generation method based on dynamic programming and the similarity of music features. The new method generates labels of corresponding dance segments according to the clustering of music features. To compare the similarity between dance segments with varying time lengths, we use KL divergence between the discrete probability distribution functions of music features that can indicate the structure of the background music.

We have used a cubic spline to adjust the length of the dance segments to accommodate different musical rhythms so that the same clip of background music can be matched with more dance moves. Because there is a one-to-many relationship between background music and dance movements, we used the background music and initial pose of the dance as the input of our method, allowing it to generate different dance motions based on the same background music.

We have compared the dances generated by our method with those of neural network-based methods, and the quantitative evaluation shows superior performance by our method for long-term dance generation.

## References


[1] T. Shiratori, A. Nakazawa, and K. Ikeuchi, "Dancing-to-music character animation," *Computer Graphics Forum*, vol. 25, no. 3, pp. 449–458, Dec. 2006.

[2] M. Lee, K. Lee, and J. Park, "Music similarity-based approach to generating dance motion sequence," *Multimedia Tools Appl.*, vol. 62, no. 3, pp. 895–912, Feb. 2013.

[3] J. Lee, S. Kim, and K. Lee, "Listen to dance: Music-driven choreography generation using autoregressive encoder-decoder network," arXiv:1811.00818, 2018.

[4] L. Bryan and Z. Stefan, "Time-series forecasting with deep learning: a survey," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 379, no. 2194, Feb. 2021, Article no. 20200209.

[5] A. Aristidou, A. Yiannakidis, K. Aberman, D. Cohen-Or, A. Shamir, and Y. Chrysanthou, "Rhythm is a dancer: Music-driven motion synthesis with global structure," *IEEE Transactions on Visualization and Computer Graphics*, vol. 29, no. 8, pp. 3519–3534, Aug. 2023.

[6] K. Chen, Z. Tan, J. Lei, S.-H. Zhang, Y.-C. Guo, W. Zhang, and S.-M. Hu, "Choreomaster: Choreography-oriented music-driven dance synthesis," *ACM Trans. Graph.*, vol. 40, no. 4, pp. 1–13, Jul. 2021.

[7] Z. Ye, H. Wu, J. Jia, Y. Bu, W. Chen, F. Meng, and Y. Wang, "Choreonet: Towards music to dance synthesis with choreographic action unit," in *Proceedings of the 28th ACM International Conference on Multimedia*, ser. MM '20. New York, NY, USA: Association for Computing Machinery, Oct. 2020, pp. 744–752.

[8] B. Li, Y. Zhao, S. Zhelun, and L. Sheng, "Danceformer: Music conditioned 3D dance generation with parametric motion transformer," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, no. 2, Jun. 2022, pp. 1272–1279.

[9] R. Li, S. Yang, D. A. Ross, and A. Kanazawa, "AI choreographer: Music conditioned 3D dance generation with AIST++," in *Proceedings of the 2021 IEEE/CVF International Conference on Computer Vision (ICCV)*. Los Alamitos, CA, USA: IEEE Computer Society, Oct. 2021, pp. 13 381–13 392.

[10] R. Huang, H. Hu, W. Wu, K. Sawada, M. Zhang, and D. Jiang, "Dance revolution: Long-term dance generation with music via curriculum learning," in *Proceedings of the International Conference on Learning Representations*, Vienna, Austria, May 2021.

[11] L. Needham, M. Evans, D. P. Cosker, L. Wade, P. M. McGuigan, J. L. Bilzon, and S. L. Colyer, "The accuracy of several pose estimation methods for 3D joint centre localisation," *Scientific Reports*, vol. 11, pp. 20 673–20 683, Dec. 2021, Article no. 20673.

[12] R. Fan, S. Xu, and W. Geng, "Example-based automatic music-driven conventional dance motion synthesis," *IEEE Transactions on Visualization and Computer Graphics*, vol. 18, no. 3, pp. 501–515, Mar. 2012.

[13] T. Shiratori, A. Nakazawa, and K. Ikeuchi, "Synthesizing dance performance using musical and motion features," in *Proceedings of the 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, Orlando, FL, USA, 2006, pp. 3654–3659.

[14] D. H. U. Kochanek and R. H. Bartels, "Interpolating splines with local tension, continuity, and bias control," in *Proceedings of the 11th Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH '84. New York, NY, USA: Association for Computing Machinery, 1984, pp. 33–41. [Online]. Available: https://doi.org/10.1145/800031.808575

[15] S. Li, W. Yu, T. Gu, C. Lin, Q. Wang, C. Qian, C. C. Loy, and Z. Liu, "Bailando: 3D dance generation via actor-critic GPT with choreographic memory," in *Proceedings of the 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, New Orleans, LA, USA, Jun. 2022, pp. 11 040–11 049.

[16] T.-h. Kim, S. I. Park, and S. Y. Shin, "Rhythmic-motion synthesis based on motion-beat analysis," *ACM Trans. Graph.*, vol. 22, no. 3, pp. 392–401, Jul. 2003.

[17] L. Kovar, M. Gleicher, and F. Pighin, "Motion graphs," in *Proceedings of the ACM SIGGRAPH 2008 Classes*, ser. SIGGRAPH '08. New York, NY, USA: Association for Computing Machinery, Aug. 2008, pp. 1–10.

[18] W.-T. Chu and S.-Y. Tsai, "Rhythm of motion extraction and rhythm-based cross-media alignment for dance videos," *IEEE Transactions on Multimedia*, vol. 14, no. 1, pp. 129–141, Oct 2012.

[19] F. Ofli, E. Erzin, Y. Yemez, and A. M. Tekalp, "Learn2dance: Learning statistical music-to-dance mappings for choreography synthesis," *IEEE Transactions on Multimedia*, vol. 14, no. 3, pp. 747–759, Jun. 2012.

[20] Y. Yan, B. Ni, W. Zhang, J. Xu, and X. Yang, "Structure-constrained motion sequence generation," *IEEE Transactions on Multimedia*, vol. 21, no. 7, pp. 1799–1812, Jul. 2019.

[21] Y. Zhou, Z. Li, S. Xiao, C. He, Z. Huang, and H. Li, "Auto-conditioned recurrent networks for extended complex human motion synthesis," in *Proceedings of the International Conference on Learning Representations*, Vancouver, BC, Canada, Feb. 2018.

[22] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, ser. NIPS'17. Red Hook, NY, USA: Curran Associates Inc., 2017, pp. 6000–6010.

[23] H.-Y. Lee, X. Yang, M.-Y. Liu, T.-C. Wang, Y.-D. Lu, M.-H. Yang, and J. Kautz, "Dancing to music," in *Proceedings of the Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds., vol. 32. Vancouver, USA: Curran Associates, Inc., 2019, pp. 3581–3591.

[24] W. Zhuang, C. Wang, J. Chai, Y. Wang, M. Shao, and S. Xia, "Music2dance: Dancenet for music-driven dance generation," *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, vol. 18, no. 2, pp. 1–21, Feb. 2022.

[25] W. Zhuang, Y. Wang, J. Robinson, C. Wang, M. Shao, Y. Fu, and S. Xia, "Towards 3D dance motion synthesis and control," *arXiv preprint arXiv:2006.05743*, 2020.

[26] G. Sun, Y. Wong, Z. Cheng, M. S. Kankanhalli, W. Geng, and X. Li, "Deepdance: Music-to-dance motion choreography with adversarial learning," *IEEE Transactions on Multimedia*, vol. 23, pp. 497–509, Mar. 2021.

[27] A. Hernandez, J. Gall, and F. Moreno-Noguer, "Human motion prediction via spatio-temporal inpainting," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, Los Alamitos, CA, USA, Oct. 2019, pp. 7133–7142.

[28] M. R. Ronchi and P. Perona, "Benchmarking and error diagnosis in multi-instance pose estimation," in *Proceedings of the IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, Venice, Italy, 2017, pp. 369–378.

[29] M. Loper, N. Mahmood, J. Romero, G. Pons-Moll, and M. J. Black, "SMPL: A skinned multi-person linear model," *ACM Trans. Graph.*, vol. 34, no. 6, pp. 1–16, Nov. 2015.

[30] B. McFee, C. Raffel, D. Liang, D. P. Ellis, M. McVicar, E. Battenberg, and O. Nieto, "librosa: Audio and music signal analysis in Python," in *Proceedings of the 14th Python in science conference*, vol. 8, Austin, Texas, USA, 2015, pp. 18–25.

[31] D. Pavllo, D. Grangier, and M. Auli, "Quaternet: A quaternion-based recurrent model for human motion," in *Proceedings of British Machine Vision Conference*, Newcastle, UK, 2018, pp. 1–14.

[32] P. Beaudoin, S. Coros, M. van de Panne, and P. Poulin, "Motion-motif graphs," in *Proceedings of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, ser. SCA '08. Goslar, DEU: Eurographics Association, 2008, pp. 117–126.

[33] S. Kucuk and Z. Bingul, *Robot kinematics: Forward and inverse kinematics*. INTECH Open Access Publisher London, UK, 2006.

[34] L. Kovar and M. Gleicher, "Flexible automatic motion blending with registration curves," in *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, ser. SCA '03. Goslar, DEU: Eurographics Association, 2003, pp. 214–224.

[35] R. Huang, H. Hu, W. Wu, K. Sawada, M. Zhang, and D. Jiang, "Music-conditioned 2D dance choreography," GitHub, [Accessed: July 19, 2021]. [Online]. Available: https://stonyhu.github.io/dancerev/

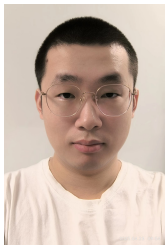
[36] R. Li, S. Yang, D. A. Ross, and A. Kanazawa, "AI choreographer: Music conditioned 3D dance generation with AIST++," GitHub, [Accessed: April 18, 2023]. [Online]. Available: https://github.com/google-research/mint

[37] L. Siyao, W. Yu, T. Gu, C. Lin, Q. Wang, C. Qian, C. C. Loy, and Z. Liu, "Bailando," GitHub, [Accessed: April 18, 2022]. [Online]. Available: https://github.com/lisiyao21/Bailando

[38] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "GANs trained by a two time-scale update rule converge to a local nash equilibrium," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, ser. NIPS'17. Red Hook, NY, USA: Curran Associates Inc., 2017, pp. 6629–6640.

[39] K. Onuma, C. Faloutsos, and J. K. Hodgins, "FMDistance: A fast and effective distance function for motion capture data," in *Proceedings of the Eurographics (Short Papers)*, Crete, Greece, Aug. 2008, pp. 83–86.

[40] M. Müller, T. Röder, and M. Clausen, "Efficient content-based retrieval of motion capture data," in *Proceedings of the ACM SIGGRAPH 2005 Papers*, ser. SIGGRAPH '05. New York, NY, USA: Association for Computing Machinery, Jul. 2005, pp. 677–685.

**Shuhong Lin** received his bachelor's degree in automation from the Nanjing University of Science and Technology and master's degree in electrical information engineering from the City University of Hong Kong. He is currently pursuing his Ph.D. degree with the Department of Electrical Engineering, City University of Hong Kong, Hong Kong. His research interests include computer graphics, machine learning and deep learning.

**Moshe Zukerman** (M'87–SM'91–F'07–LF'20) received the B.Sc. degree in industrial engineering and management, an M.Sc. degree in operations research from the Technion – Israel Institute of Technology, Haifa, Israel, and a Ph.D. degree in engineering from University of California, Los Angeles, in 1985. He was an independent consultant with the IRI Corporation and a Postdoctoral Fellow with the University of California, Los Angeles, in 1985– 1986. During 1986–1997, he was with Telstra Research Laboratories (TRL), first as a Research Engineer and, during 1988–1997, as a Project Leader. He also taught and supervised graduate students at Monash University in 1990–2001. During 1997-2008, he was with The University of Melbourne, Victoria, Australia. Since December 2008 he has been with the Electronic (now Electrical) Engineering Department of City University of Hong Kong (CityU) as a Chair Professor of Information Engineering and a team leader. Between 2020 - 2022, he also served as the Acting Chief Information Officer of CityU. He has over 300 publications in scientific journals and conference proceedings.

**Hong Yan** received his PhD degree from Yale University. He was Professor of Imaging Science at the University of Sydney and currently is Wong Chun Hong Professor of Data Engineering and Chair Professor of Computer Engineering at City University of Hong Kong. Professor Yan's research interests include image processing, pattern recognition, and bioinformatics. He has over 600 journal and conference publications in these areas. Professor Yan is an IEEE Fellow and IAPR Fellow. He received the 2016 Norbert Wiener Award from the IEEE SMC Society for contributions to image and biomolecular pattern recognition techniques. He is a member of the European Academy of Sciences and Arts and a Fellow of US National Academy of Inventors.