A Music-driven Deep Generative Adversarial Model for Guzheng Playing Animation

Jiali Chen, Changjie Fan, Zhimeng Zhang, Gongzheng Li,
Zeng Zhao, Zhigang Deng, *Senior Member, IEEE*, Yu Ding

Abstract—To date relatively few efforts have been made on the automatic generation of musical instrument playing animations. This problem is challenging due to the intrinsically complex, temporal relationship between music and human motion as well as the lacking of high quality music-playing motion datasets. In this paper, we propose a fully automatic, deep learning based framework to synthesize realistic upper body animations based on novel guzheng music input. Specifically, based on a recorded audiovisual motion capture dataset, we delicately design a generative adversarial network (GAN) based approach to capture the temporal relationship between the music and the human motion data. In this process, data augmentation is employed to improve the generalization of our approach to handle a variety of guzheng music inputs. Through extensive objective and subjective experiments, we show that our method can generate visually plausible guzheng-playing animations that are well synchronized with the input guzheng music, and it can significantly outperform the state-of-the-art methods. In addition, through an ablation study, we validate the contributions of the carefully-designed modules in our framework.

Index Terms—deep learning, generative adversarial networks, motion capture, guzheng animation, music-driven, data augmentation

1 Introduction

While playing music with an instrument, musicians are generally in continuous motion [1], often involving facial expression, hand gesture, torso movement, etc. Such visual behaviors are not only dedicated to touching a musical instrument at the right place for matching the score [2] but also visually consistent with the music rhythm to convey musical expression and thoughts to the audience [3], [4]. These visual cues reflect the musician's interpretation of the music [5]. On the other hand, human observers are intrinsically skilled at perceiving the conveyed emotion and intention from such visual behaviors of music playing.

To generate instrument-playing animations in concert with given music, manually making such animations or direct motion capture of the musician's instrument playing performances are two potential solutions. However, manually making such animations are labor-intensive, non-trivial, and less accurate. Collecting large-scale, quality instrument-playing motion capture data not only is expensive but also requires overwhelming efforts on manual data cleaning and correction. To this end, these two methods are at most limited to few delicately planned scenarios. Another direction to solve this issue would be to automatically generate musical instrument playing animations based on novel inputted music, without human intervention. In [6], researchers analyzed the creativity of computers in generating expressive music performances and proved that certain aspects of personal styles are recognizable. This suggests that it is potentially possible to directly generate music playing performances based on novel inputted music. Also, several previous works have attempted to model the relationships between music and the corresponding instrument playing behavior at low-level representations [7], [8].

A straightforward data-driven solution to the automated generation of instrument playing animations would be to select pre-defined action segments according to the features of novel inputted music, and then concatenate and interpolate them as the final animation. Such a method requires carefully collecting and processing action segments, and even so it is difficult to ensure good synchronization between the input music and actions. With the rapid advances of machine learning techniques in recent years, in particular, deep learning algorithms, researchers started to exploit deep learning for the automatic generation of instrument playing animations. For example, Shlizerman et al. [9] utilized the classical temporal model of deep learning, long short-term memory (LSTM) [10], to generate 2D skeleton animations of playing the piano or violin from novel inputted music. However, the LSTM-based network is time-consuming due to the inherently sequential computation [11]. Moreover, in their approach, only the regression loss is used, but the regression loss focuses on the generated animation at frame-level. More importantly, the adversarial loss that can enforce the distribution of synthetic human motions to be close to that of real human motions is not utilized, which affects the quality of the resultant animations in their approach. Indeed, to date, automatically generating high quality instrument playing animations for novel inputted music is still considered a wide open problem.

Deep learning techniques have been successfully applied to many fields and applications. For example, in recent years, the Unet [12], a variant of the CNN net-

- *J. Chen, C. Fan, Z. Zhang, G. Li, Z. Zhao, J. Bu, and Y. Ding are with the NetEase Inc., China.*
  Yu Ding is the corresponding author
  E-mail: dingyu01@corp.netease.com
- *Z. Deng is with the Department of Computer Science, University of Houston, Houston, TX, USA 77204-3010.*
  Email: zdeng4@central.uh.edu

This manuscript was accepted to IEEE Transactions on Visualization and Computer Graphics in September 2021.

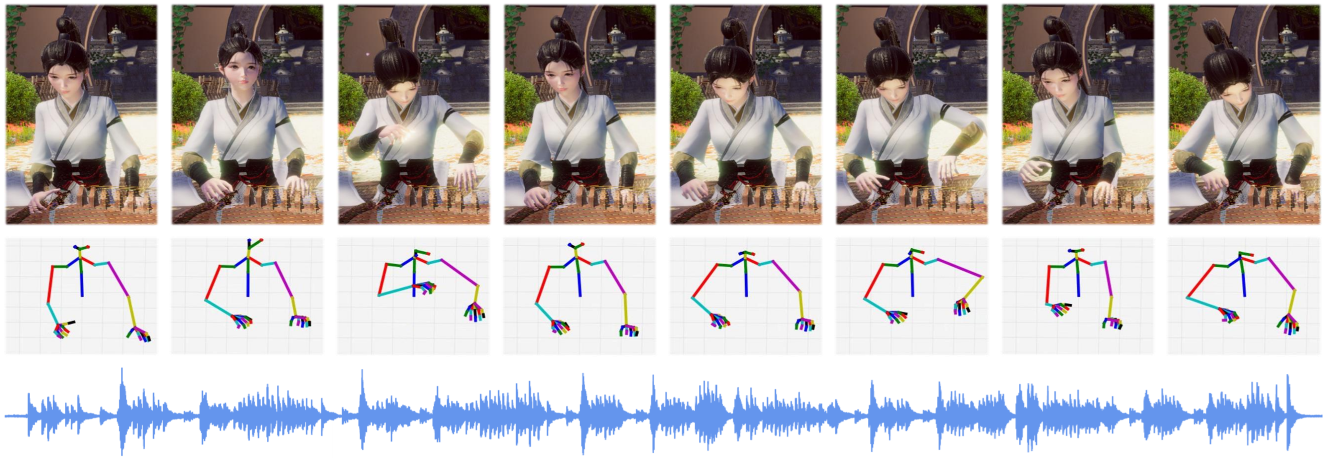


Fig. 1. Frames of a generated Guzheng-playing animation. The bottom row shows the input music; the middle row shows the outputted skeletal animation of the upper body; and the top row shows the corresponding virtual character animation.

work, demonstrated noticeable successes in multiple tasks, including medical imaging [13], [14], image generation [15], and conversational gesture generation [16], due to its powerful capacity of capturing multi-scale input. In addition, Generative Adversarial Networks (GAN) [17] has been proved to be an effective framework for generating realistic images [15], [18] and video-realistic facial expressions [19], especially for producing high-frequency details in images/video.

In this paper, taking advantage of the recently developed Unet [12] and GAN [17], we propose a Unet-based, end-to-end, music-to-motion GAN to synthesize the upper body motion of playing the Guzheng (a widely-known musical instrument in China) for any input music. Specifically, our method extends the Unet from the image domain to the animation domain, in order to capture the short-time dependence and the long-time dependence relationships between music and motion. Based on a recorded audiovisual dataset of Guzheng playing, acquired by an in-house motion capture system, a carefully-designed, Unet-based GAN is developed to model the dynamic correlation between the music and motion in the dataset, and the trained GAN can be used to generate realistic upper body animations given novel inputted music. Via various objective and subjective experiments, we demonstrated that our approach can generate natural and visually-plausible upper body animations of Guzheng playing, and it can soundly outperform the state-of-the-art LSTM-based and CNN-based methods [9], [12]. Figure 1 shows some frames of a synthesized Guzheng playing animation by our approach.

The motion data used in [9] were captured via the OpenPose library [20], but such data are well-known to suffer from the problems of mis-detection [9] and bone distortion [21]. In this work, to preserve the accurate temporal relationship between music and motion, we collected an in-house, high-quality dataset of Guzheng-playing motions, with the aid of a professional motion capture system. The collected Guzheng-playing motion dataset is publicly released for the purpose of research[1].

1. https://github.com/FuxiVirtualHuman/Guzheng-Playing

The main contributions of this work can be summarized as follows:

- Drawing on the benefits of both the Unet and the GAN, we propose an end-to-end, music-to-motion GAN framework to synthesize visually-plausible upper body motion based on novel inputted Guzheng music;
- we build the first-of-its-kind, high quality, Guzheng-playing motion dataset, which will be released for the research purpose in the research community.

2 Related Work

Since our task is essentially an animation generation problem, in this section we first review recent related works on the synthesis of facial animation, conversational gesture animation, dance animation, and musical instrument playing animation, then report the recent developments on deep generative adversarial networks.

Facial animation. Many researchers have made great efforts to explore the synthesis of facial animations and expressions [22]–[47]. In recent years deep learning techniques have been exploited for facial animation synthesis. For example, Karras et al. [36] utilize CNN to learn the cross modal mapping between audio and facial animation. Pham et al. [39] employ the LSTM to capture the temporal dependencies and futher combine the CNN and LSTM to improve the model performance [42]. Readers of interest can refer to recent comprehensive surveys on facial animations [48], [49].

Conversational gesture animation. With recent developments on deep learning, researchers also employ deep neural models to synthesize conversational gestures from speech. For example, both Ferstl et al. [50] and Ginosar et al. [16] use LSTM based structures to synthesize 3D joint angles and 2D joint positions, respectively, from input speech prosody. Kucherenko et al. [51] first take an encoder-decoder structure to map 3D joint position into a lower dimensional, pose embedding space to remove pose noise, and then utilize a LSTM-based framework to regress the pose embedding. Recently, Jin et al. [52] proposed a LSTM-based approach to generate realistic three-party head and eye motions based on novel acoustic speech input together with speaker marking (i.e., speaking time for each interlocutor). Considering that there is a many-to-many mapping between speech and gesture, Rodriguez et al. [53] introduce a generative adversarial network to improve the quality of the generated gestures. Our work also leverages the adversarial training strategy and replaces LSTM with CNN to accelerate the framework without sacrificing the synthetic quality.

Dancing Animation. Existing dancing animation synthesis works can be roughly divided into motion segment based methods and generative model based methods. Generally, the motion segment based methods [54]–[59] first cut several pre-defined choreography dance segments from a database, and then select and parse these segments into a dance sequence. However, different methods use distinct segmentation strategies, e.g., Shirator et al. [54] design matching rules according to rhythm features. Lee et al. [55] first compute the music similarity and then select the best matched dance segments. Fukayama et al. [56] simulate the matching criterion with Gauss processes. Berman et al. [57] leverage motion graphs to optimize the selection of motion segments. Ye et al. [58] utilize LSTM to learn the matching between music and dance segments. Chen et al. [59] add extra information of music style and rhythm signatures in the matching process.

The generative model based methods [60]–[66] aim to directly synthesize the dance frame at each time step. Ofli et al. [60] employ hidden markov models (HMM) to generate dance motion. Alemi et al. [61] utilize Factored Conditional Restricted Boltzmann Machines (FCRBM) and RNN to generate joint angles. Tang et al. [62] proposed a LSTM-autoencoder framework to synthesize joint positions. Lee et al. [63] proposed a framework to produce action units instead of action frames to synthesize smooth motions. Lee et al. [64] proposed a CNN based encoder-decoder framework to generate 2D skeleton coordinates. Huang et al. [65] proposed a self-attention based network to learn a cross-modal mapping and utilize curriculum learning to reduce error accumulation. Wallace et al. [66] treat the dance generation from music as a one-to-multimodal distribution mapping, and proposed a Mixture Density Recurrent Neural Network(MDRNN) to learn the mapping.

Both the above motion segmentation based methods and the generative model based methods may not be suitable for the synthesis of instrument playing animations. The main reason is that dancing animation synthesis generally only considers the rhythm and style in the music, while instrument playing animation needs the mining of the musical scores. For instance, pianists are able to translate piano music into MIDI files easily, but it is very difficult to analogously infer dance motion from music.

Musical instrument playing animation. In recent years researchers developed many deep learning methods to automatically synthesize various musical instrument playing animations [9], [67]–[70]. For instance, Li et al. [67] combine the CNN and LSTM to produce pianist body movements from MIDI note streams and additional metric structures. In their work, the CNN is used to extract musical features and LSTM is employed to capture temporal dependencies. Shlizerman et al. [9] utilize the vanilla LSTM to automatically generate 2D skeletal animations of piano or violin playing given music input, and then further use the skeletal animations to drive the animation of pre-defined 2D textured characters. Considering the existence of different motion patterns in different body parts, Liu et al. [68] proposed a three branch framework to synthesize the violin playing motion for the right hand, the left hand, and the upper body, respectively. Bogaers et al. [69] explored more music features in piano animation generation and demonstrated the usefulness of MFCC features. Kao et al. [70] design a two-branch network to synthesize the movements of the right hand and body according to the characteristics of violin playing. In the right hand branch, they proposed a framework combined with the Unet, LSTM and self-attention, which is similar to our method. In our work, we also do comparative experiments with their framework.

Deep Generative Models. In comparison with the LSTM model [10], the CNN-based network has the capability of parallel computation. CNN has been widely used in temporal sequence processing [11], [71] and image processing [12], [72], [73]. Recently Bai et al. [71] proposed a temporal convolutional (neural) network (TCN) on sequence modeling and demonstrated that the TCN model can substantially outperform generic LSTM models. Dauphin et al. [11] use one linear mapping path in each convolutional layer to reduce both the vanishing gradient problem and convergence time. Researchers also explored to use the identity path, which is similar to linear mapping path in image processing [72], [73]. The above works show that the delicately-designed, CNN-based networks are also capable of modeling temporal sequences.

Ronneberger et al. [12] proposed the Unet network, a variant of CNN, for image segmentation. Later, due to its special structure of down-sampling layers and up-sampling layers with the skipped connections between them, the Unet network has been successfully extended for multiple tasks, including medical imaging [13], [14], image generation [15], and conversational gesture generation [16]. The module of the down-sampling layers is a CNN-based encoder, mainly consisting of convolution layers and max pooling layers. The module of the up-sampling layers is a CNN-based decoder, mainly consisting of convolution layers and deconvolution layers. Oktay et al. [14] further use attention mechanisms in the Unet network for medical image segmentation.

The GAN model was first proposed in [17] to generate images from random noise. The GAN network consists of a generator G and a discriminator D. In the training stage, G is trained to confuse D, and D is trained to correctly distinguish whether the output of G is real or fake. Mirza et al. [18] proposed the conditional GAN (cGAN) to control image synthesis according to input conditions. The discriminator in the cGAN distinguishes not only the synthetic image is real or fake but also whether the image matches the conditions. Isola et al. [15] proposed a patch discriminator that has the benefit of fewer parameters and runs faster. For a comprehensive review on GAN models and their latest applications, please refer to the recent survey article [74].

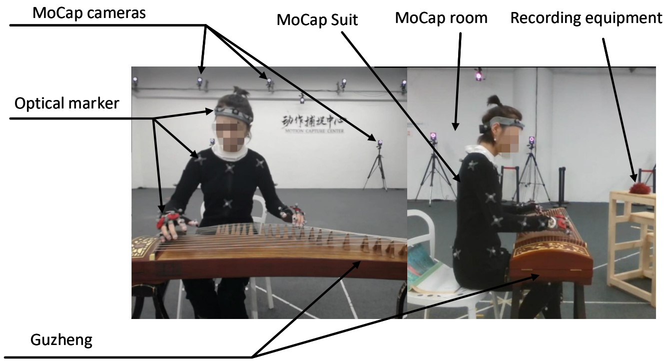


Fig. 2. Snapshot of the in-house motion capture setup for collecting data in this work.

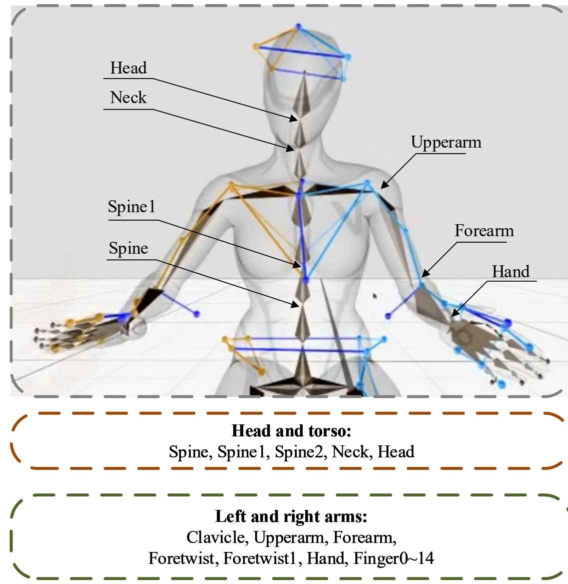


Fig. 3. Illustration of the human joints used in this work.

3 Data Collection And Processing

Our deep learning based framework needs to use a quality dataset for model training. Therefore, in this work we used an in-house motion capture setup to record a dataset that contains both the audio (music) and motion of the musician who plays Guzheng. Note that the music and human motion were acquired simultaneously. In this section we describe the data collection and processing step.

3.1 Data Collection

To obtain a high-quality dataset, we invited a Guzheng musician to play 36 pieces of Guzheng music. Each piece lasts from 45 seconds to 6 minutes, and the total recording time is 1 hour 6 minutes 23 seconds. This dataset was collected in a VICON motion capture room. As shown in Figure 2, the musician wears a motion capture suit with 59 optical mocap markers at specific locations of the human body, including joints, hips, elbows, wrists, etc.

When the musician plays the Guzheng instrument, the motion capture system records the movements of the human joints, from which we can further extract the rotations and displacements of the joints. Since the focus of our work is on the upper body motion, 47 joints of the upper body, including the torso, head, arms, and fingers, are used in this work, illustrated in Figure 3. The motion capture data were captured at 30 frames per second (*fps*), and the Guzheng music was recorded with 44.1 kHz through a professional audio recording device. Due to the limitations of the used optical motion capture system (e.g., limited capability to handle occlusions), the recorded finger motion data cannot accurately reflect the fingers' movements. Therefore, we manually corrected finger motions in our data processing step.

Although our data were collected in the professional motion capture room, a few reasons make the relative positions of both the hands and the strings of the Guzheng are not correct sometimes. Specifically, the skeleton scales of the real human and the virtual character are different. Motion retargeting from the human to the virtual character sacrifices the accuracy of motion, to a certain extent. Moreover, the size of the virtual Guzheng is also different from the real one. Also, the recorded music and motion were manually aligned by a professional annotator, with the aid of the ELAN software [75].

3.2 Data Processing

To meet the need of our model training task, we processed the recorded music and motion sequences separately and then composed them into a set of audiovisual data.

Music processing. Widely used in audio feature extraction, spectrogram has been successfully used for a variety of applications, including voice conversion [76], speaking gesture generation [16], etc. In this work, we extracted 768-dimensional spectrogram features as the input to our model. Compared to MFCC features used in [9], spectrogram features have higher dimensions and keep more information from the raw audio data. Each sample of music audio is represented by a sequence of spectrogram features $\boldsymbol{f} = \{f_1, f_2, ..., f_t, ..., f_T\}$, where f_t is a 768-dimensional vector of spectrogram features and T is the total number of frames in $\boldsymbol{f}$.

Motion processing. Since the virtual character animation is controlled by a bound skeleton, we represent and store the rotation of each joint as a Quaternion (4 dimensions). Quaternion is chosen over other rotation representations (such as Euler angles) since it is more suitable for smooth rotation interpolation and prevents the Gimbal lock [77], [78].

Based on the above quaternion representation, the upper body motion is represented by a sequence ($\boldsymbol{m}$) of 188-dimensional vectors (denoted as m_i), $\boldsymbol{m} = \{m_1, m_2, ..., m_t, ..., m_T\}$, where T is the length of sequence $\boldsymbol{m}$, and m_t is the 188-dimensional motion features at time t (47 joints $\times$ 4 per quaternion = 188). Moreover, through the forward kinematics algorithm, the positions of 3 end effectors (i.e., the left hand, the right hand, and the head) were calculated and represented by a sequence ($\boldsymbol{p}$) of 9-dimensional vectors (denoted as p_t), $\boldsymbol{p} = \{p_1, p_2, ..., p_t, ..., p_T\}$, and p_t is composed of the xyz positions of the 3 end effectors. Specifically, we calculated the position of each end effector relative to the skeletal root,

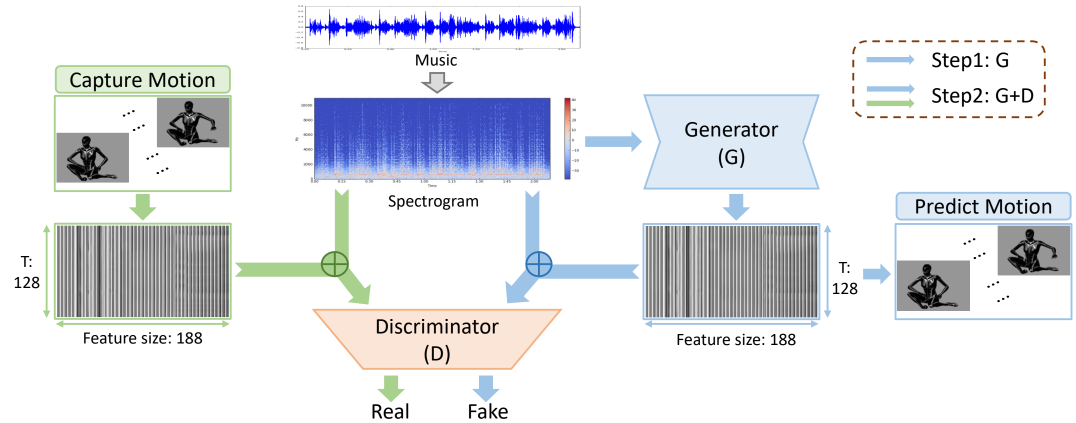


Fig. 4. Pipeline overview of the proposed music-to-motion framework. The framework consists of a generator and a discriminator. Please see Figures 5 and Figure 9 for more details on the generator and the discriminator, respectively.

and then further normalized the position data in order to enforce the resultant data distributing between [0,1].

To this end, we created an audiovisual dataset, $\{\boldsymbol{f}, \boldsymbol{m}, \boldsymbol{p}\}$, consisting of 36 audiovisual pieces. Each piece has a different length. To meet the training requirement, each piece was split into audiovisual segments, each of which has empirically-chosen 128 frames (about 4 seconds). Finally, the dataset includes a total of 108,372 audiovisual segments, denoted as $S = \{\boldsymbol{f}^{128\times768}, \boldsymbol{m}^{128\times188}, \boldsymbol{p}^{128\times9}\}$.

4 Music-to-Motion GAN

The goal of our approach is to automatically synthesize realistic upper body motion, including torso motion, head motion, arm motion, and finger motion, based on a Guzheng music as the given input.

Our music-to-motion model is a GAN-based framework, illustrated in Figure 4. Specifically, an animation generator G is built to synthesize upper body motion sequences $\tilde{\boldsymbol{m}} \in R^{128\times188}$ from the spectrogram features of the input Guzheng audio, $\boldsymbol{f} \in R^{128\times768}$. Furthermore, the audio spectrogram f is separately concatenated with the real motion sequence m and the synthetic motion sequence $\tilde{m}$ to form two tuples: $\{f, \tilde{m}\} \in R^{128\times956}$ and $\{f, m\} \in R^{128\times956}$. A discriminator D is designed to determine whether $\{f, \tilde{m}\}$ and $\{f, m\}$ is real or fake. G and D are CNN-based neural networks where one-dimensional convolutions are utilized over the audio spectrogram f_t or the motion m_t and carried out along with the time dimension t. In the training, G produces realistic upper body motions as much as possible to fool D. Meanwhile, D is updated to correctly distinguish the synthetic tuple $\{f, \tilde{m}\}$ from the real tuple $\{f, m\}$. G is trained under the supervision of D. This adversarial training process aims to enforce the synthetic joint movements $\tilde{m}$ more realistic and natural. The details of our framework are described in the remainder of this section.

To refine the generated animation, G is supervised by two regression losses and a GAN loss. The regression losses govern both the rotations of the joints and the positions of the end-joints, and they are designed to make the synthetic animation as close to the real data. The GAN loss performs via a pyramid discriminator, which is designed to prevent the resultant animation falling into the mean value and to enforce the generated animation follows the distribution of the real motion. The combination of the regression losses and GAN loss is able to ensure the generated motion more natural and realistic.

4.1 Data Augmentation

In general, training a deep learning framework requires a large amount of data. Considering the limited amount of our recorded music/motion data and the generalization of the generator for various music inputs, we carry out a data augmentation step by expanding the diversity of the input music and by slightly stretching/shrinking the simultaneously recorded music and motion data.

Stretching and shrinking audiovisual data. A piece of music may be played relatively fast or slow. To satisfy the requirement of differential music speeds, the recorded music and motion data are slightly stretched or shrank simultaneously. Our experiments found that too much scaling would negatively affect the result because the musician's upper body could move differently when s/he plays music at different music tempos. For example, when fast-tempo music is played, the arm's action space is relatively small; in contrast, while slow-tempo music is played, the arm's action space could be relatively large, which cannot be achieved by simply stretching the motion in the temporal dimension. By randomly sampling the scaled music and motion to the Audition software, we can find a suitable scaling factor. To maintain the quality of both the resultant music and motion, we use a scaling factor between 0.75 and 1.25 for shrinking and stretching, respectively. We use a constant-speed adjustment method for music; we stretch or shrink the motion data using the cubic Spline Interpolation [79].

4.2 Generator: Mapping from Audio to Body Motion

The upper body motion generator in this work aims to map the spectrogram features of Guzheng music to the joint angles of the upper body (represented by quaternions). Our generator is based on a U-shaped deep neural network, as illustrated in Figure 5. In the down-sampling layers, 4 1D residual blocks and 4 max pooling layers are employed. Each 1D residual block is followed by a max

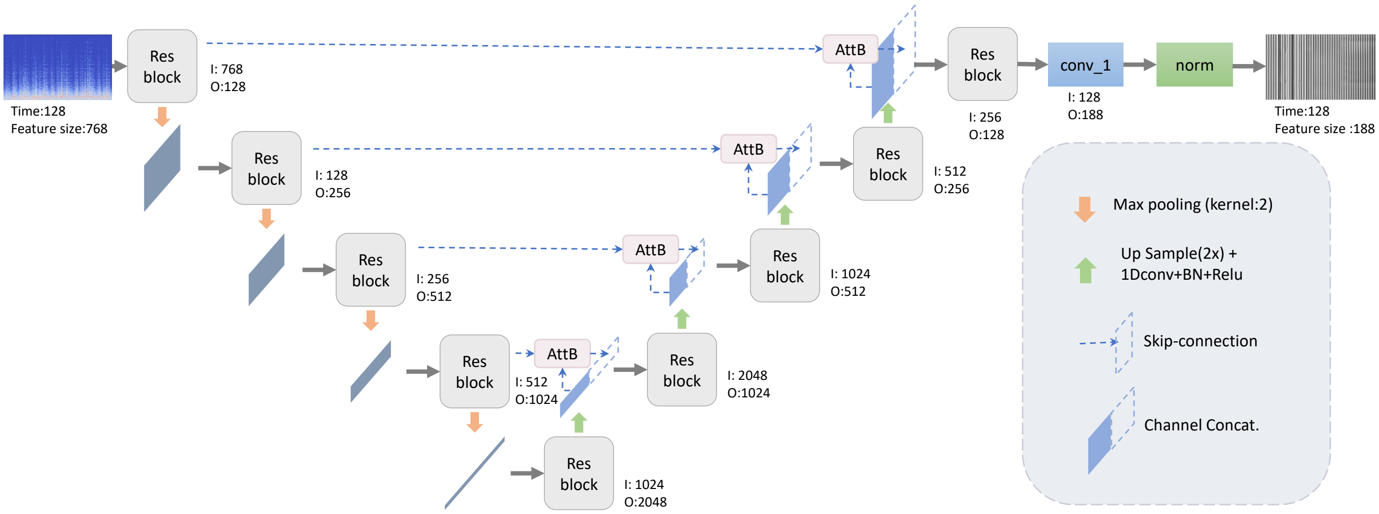


Fig. 5. Pipeline illustration of the Music-to-Motion generator in this work.

pooling layer. The 1D residual block and the max pooling layer are repeated alternately to extract long-range temporal contextual information and high-level abstract information from the input audio. The output of each max pooling layer is denoted as a down-sampling feature map. Along with the down-sampling operations, multi-scale down-sampling feature maps are also obtained.

To improve the performance of the generator, we use attention blocks (AttB in Figure 5) and "upsample + 1D conv" layers. The attention blocks manipulate the temporal weights to control the flow from the down-sampling layers to the up-sampling layers. "upsample + 1D conv" layers are used to avoid the problem of checkerboard artifacts [80].

In the up-sampling layers, 5 1D residual blocks and 4 "upsample + 1D conv" layers are applied to compute the quaternion sequence of each upper body joint. Each of the first four 1D residual blocks is followed by one "upsample + 1D conv" layer. The output of each "upsample + 1D conv" layer is denoted as a up-sampling feature map. Along with the up-sampling operations, multi-scale up-sampling feature maps are obtained. In particular, the down-sampling and up-sampling feature maps with the same scales in those symmetric layers are fused as the input to the 1D residual blocks in the up-sampling path. The fusion is done by concatenating the up-sampling feature map and the weighted down-sampling feature map. The weights are calculated through an attention block.

The up-sampling feature map of the 5-*th* 1D residual block is fed into a linear transformation on the feature channels through a convolutional layer (kernel size is 1, stride is 1) to fit the dimension of the motion m. Then, a normalization operation is performed on the feature map to satisfy the rotation constraint.

1D residual blocks. Our 1D residual block (ResB) consists of two paths: a residual path and an identity mapping path. Its structure is illustrated in Figure 6. In the residual path, 1D convolutional layer, the BN operation [81] and relu function [82] are stacked to extract non-linear features. In the identity mapping path, a 1D convolutional layer (kernel size is 1, stride is 1) is used to make the channel size the

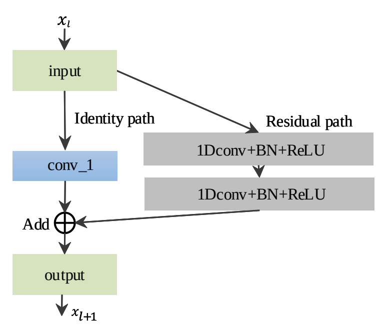


Fig. 6. Schematic illustration of the used Residual block (abbreviated as *Res block* or *ResB*).

same as that of the residual path. The results from both the residual path and the identity mapping path are added as the output of the 1D Residual block. Our 1D residual block can be represented as

$$x_{l+1} = \mathcal{I}(x_l, \theta_i) + \mathcal{R}(x_l, \theta_r), \tag{1}$$

where x_l, x_{l+1} are the input and output of the l^{th} 1D residual block; $\mathcal{I}$ represents the identity mapping path; $\mathcal{R}$ denotes the residual path; θ_i and θ_r are the parameters of the two paths respectively. Our 1D residual block design has the benefits of both reducing the training loss and improving the performance, which is described in our ablation experiments (refer to Section 5.)

Upsample + 1D conv. In the up-sampling layers of the generator, deconvolutional layers generally cause the problem of checkerboard artifacts [80] due to uneven overlapped placement of each deconvolutional pattern. This always leads to the motion jittering of the joints. Inspired by the work of [80], we employ linear interpolations to enlarge the hidden features, which avoids the uneven overlapped pattern placement in the deconvolution. Moreover, a 1D convolutional operation with the BN and relu operations [81], [82] are followed by the enlarged hidden features to perform a non-linear feature mapping. The "upsample + 1D convolutional" layers contribute to the generation of more natural motion than the deconvolutional layers. Figure 7 shows several comparison examples of the motion trajectories generated with the deconvolutional layers and the "upsample + 1D convolutional" layers. As shown in this figure, the "upsample + 1D convolutional" layers can generate smoother motion trajectories than the deconvolution layers.

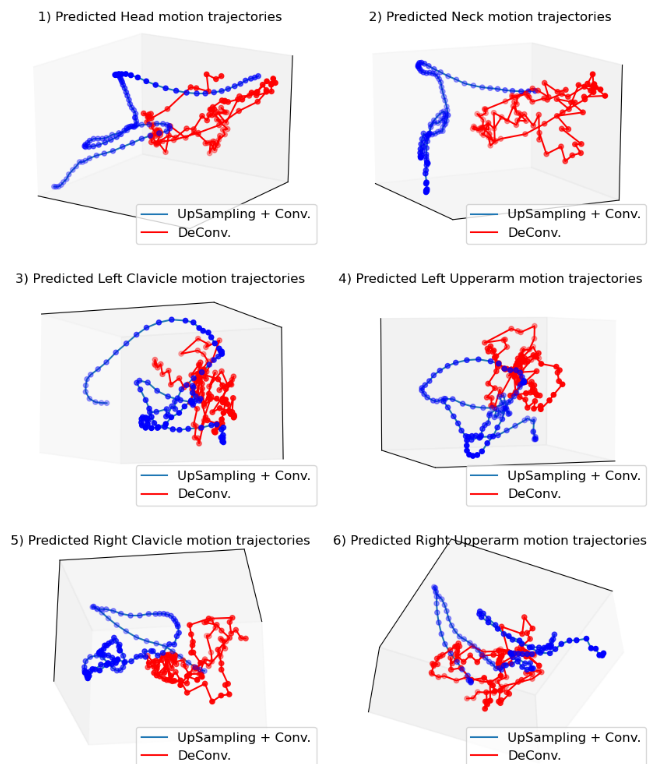


Fig. 7. Comparisons of the generated motion trajectories in 3D space. The red and blue trajectories are generated with the deconvolution and "upsampling + convolution" layers, respectively, in the decoder.

Attention blocks. The attention block (AttB) [14] is used to control the flow of the down-sampling feature maps f_d into the concatenation with the up-sampling feature maps f_u that have the same scale. The schematic illustration of the AttB is illustrated in Figure 8. AttB outputs f_d that is weighted along with the temporal dimension. The weights are computed by first mapping f_d and f_u to the same hidden feature space with two linear transformation matrix W_d and W_u, respectively; then by fusing them through an element-wise addition; and finally by feeding the summation into a stack of a non-linear function of relu, a convolutional layer (kernel size is 1, stride is 1) and a sigmoid function. The down-sampling feature map is element-wise multiplied with the weights in the temporal dimension as the output of the AttB, described in the following equation (2).

$$o = f_d \cdot \sigma(conv(relu(W_d^T f_d + W_u^T f_u))), \qquad (2)$$

where o denotes the output of the AttB, and W_d and W_u are the linear transformation matrices for f_d and f_u, respectively. Our attention block design has the benefits of both highlighting the salient latent features and suppressing the irrelevant parts of the latent features.

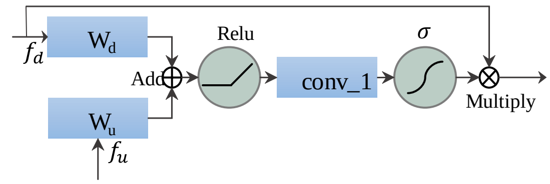


Fig. 8. Schematic illustration of the Attention block (AttB).

4.3 Discriminator: Multi-scale Patch Discrimination

A multi-scale patch discriminator, D, is designed to supervise the training process of the generator. It contributes to refine the realistic movements of the upper body joints. It is illustrated in Figure 9. It consists of four sub-discriminators, denoted as $D = \{D_1, D_2, D_3, D_4\}$. $D_i, i = 1, 2, 3, 4$ is a patch discriminator [15] with the multi-scale receptive fields of 1 for D_1, 12 for D_2, 48 for D_3 and 128 for D_4. The multi-scale receptive fields govern the output of the generator at different scales and help to refine the output motion trajectories. D_i consists of multiple convolution layers, each of which is built with a 1D convolution layer, batch normalization, and the reLu activation function. In our work, the numbers of the neural layers in $\{D_i\}$ are different: 4 layers for D_1, 3 layers for D_2, 5 layers for D_3, and 5 layers for D_4. Each D_i outputs the binary probability distribution of true or false, denoted as p_i. The average value of $\{p_i\}, i = 1, 2, 3, 4$, is considered as the output of D, the final probability distribution of true or false.

4.4 Loss Functions

In the training process, the generator is supervised with three loss functions: the joint rotation loss L_{jr}, the end-effector position loss L_{ejp}, and the GAN loss L_{GAN}; and the discriminator is supervised only with L_{GAN}.

We first consider L_{jr} in our experiments, since L_{jr} contributes to govern the accuracy of all the joints. However, using L_{jr} alone would easily cause inaccurate positions of the end-effectors, due to the accumulated errors in the forward kinematics process. This also means that the joints closer to the root typically have a greater impact on the positions of the end-effectors. To solve this issue, we add an extra loss L_{ejp} to constrain the positions of the end-effectors. L_{ejp} has influence on all the joints due to the fitting of the joint probability distribution of all the joints. The GAN loss L_{GAN} has the benefits of both encouraging high-frequency details [15] and synthesizing realistic upper body motion due to the joint modeling of motion and music signals. So we add L_{GAN} in our design. We detail the three loss functions below.

Joint rotation loss. L_{jr} is the L_1-norm distance between the synthetic joint rotation sequence $\tilde{m}^{128\times 9}$ and the real joint rotation sequence $m^{128\times 9}$, computed as:

$$L_{jr} = \left\| m - \tilde{m} \right\|_1 . \qquad (3)$$

End-effector position loss. To guarantee the hands to touch the Guzheng instrument and the plausible position of the head, the end-effector position loss, L_{ejp}, is designed. L_{ejp} computes the distance between the synthetic and real positions of the three end-effectors. Specifically, the

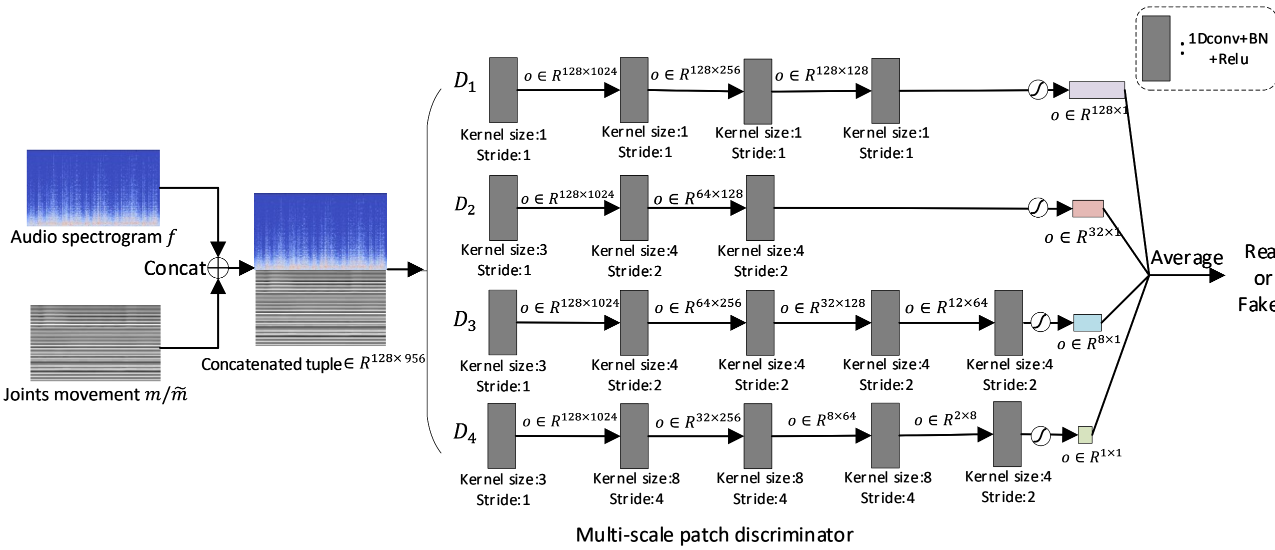


Fig. 9. Schematic illustration of the multi-scale patch discriminator.

differentiable forward kinematics (FK) algorithm and the quaternion representations of the joint rotations are utilized to compute the positions of the end-effectors, as follows.

$$L_{ejp} = \|p - FK(\tilde{m})\|_1 \,, \tag{4}$$

where $\boldsymbol{p}^{128\times 9}$ is defined in Section 3.2 and it refers to the ground truth positions of the end-effectors; $FK(.)$ denotes the iterative forward kinematics function and it estimates the 9-dimensional (two hands and the head) end-effector positions for 128 frames according to a synthetic joint rotation sequence $\tilde{m}$.

GAN loss. The GAN loss is formulated as:

$$L_{GAN} = \min_G \max_D F_{GAN}, \tag{5}$$

where:

$$\begin{aligned} F_{GAN} &= \frac{1}{4}\sum_{i=1}^{4} \mathbb{E}_{m,f}\left[logD_i(\{f,m\})\right] \\ &+ \frac{1}{4}\sum_{i=1}^{4} \mathbb{E}_{\tilde{m},f}\left[log(1 - D(\{G(f),f\}))\right]. \end{aligned} \tag{6}$$

Our final objective function is computed as:

$$G^* = arg\min_G (L_{GAN} + \lambda_{jr}L_{jr} + \lambda_{ejp}L_{ejp}), \tag{7}$$

where λ_{jr} and λ_{ejp} are the weights for the joint rotation loss L_{jr} and the end-effector position loss L_{ejp}, respectively. In our experiments, both λ_{jr} and λ_{ejp} are set to 100.

4.5 Implementation Details

In our experiments, the training process was split into two alternate steps. At step 1, the generator G was trained with the regression loss functions only (including the joint rotation loss L_{jr} and the end-effector position loss L_{ejp}), while the discriminator D was kept without any update. At step 2, D was trained with not only L_{jr} and L_{ejp} but also the GAN loss L_{GAN}. Meanwhile, L_{GAN} was used to train the discriminator D. In our experiments, we trained the generator through 150k iterations at step 1, and then use the GAN loss at step 2 through 40k iterations. The learning rate was set to 0.0001, and the batch size was set to 128. The Adam solver [83] was used to optimize the network parameters. All the models were implemented using PyTorch [84]. Since our model uses a full-convolution network, the network can be adapted to any length of time during the animation generation.

5 Experiment Results and Evaluations

To evaluate our approach, we conducted both quantitative and qualitative evaluations. In this section, we first describe various baseline methods used in our evaluations, and then describe the quantitative result and qualitative evaluation (via a user study).

Baselines. The baseline methods in this work include the LSTM network [9], the CNN forward network [36], the CNN and LSTM combination network [42],the TCN network [71], the Unet network [12], and the R2Unet network [85]. Particularly, the CNN forward network baseline is inspired by [36] and ignores the emotional state input layer used in [36]; and the CNN and LSTM combination baseline [42] was proposed to generate 3D facial animations, where CNN is used to extract abstract audio features in each frame and LSTM is applied to model the temporal relation between frames. Additionally, as the generator in our model is an extension of the Unet network, we take the Unet network as a baseline; Further, a latest extension of the Unet called R2Unet, short for Recurrent Residual Unet [13], is also considered as a baseline in this comparison. It benefits from the advantages of a residual structure and a convolution recurrent network [86].

Ablation study design. To look into the contribution of each major module in our framework, we conducted an ablation study to investigate the contributions of the Res block, the Attention block, and the GAN framework. Therefore, three framework conditions are defined: $M^{\overline{GAN}}$ is the proposed generator trained with both the joint rotation loss and the end-effector position loss but without the GAN

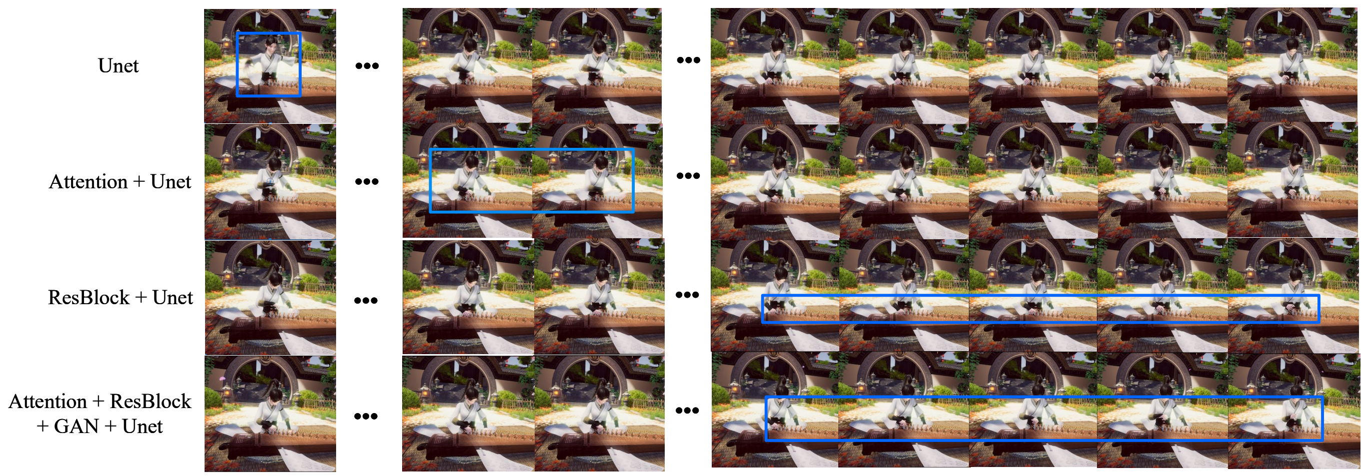


Fig. 10. Some frames of synthetic results by different versions in our ablation study.

loss; differing from the proposed framework, $M^{\overline{Att}}$ fuses the downsampling and upsampling layers with the concatenation operation instead of the attention block; $M^{\overline{Res}}$ takes the convolutional layers in the downsampling and upsampling paths to replace the Res blocks in the proposed framework.

To have a fair comparison among all the methods, we maintain the consistency of the input and output features when comparing the baselines and our method. The inputs are audio spectrogram features and the outputs are the joint rotations of the upper body. All of the methods were uniformly trained on the augmented dataset (described in Section 4.1).

5.1 Quantitative Evaluation

We used quantitative measures, including the test loss, Dynamic Time Warping distance (DTW) [87], and the Longest Common Subsequence similarity (LCS) [88], to compare our method to the baselines. Specifically, the test loss, DTW distance and LCS similarity compute the skeletal joint trajectory distance between the generated motion sequence and the ground truth motion sequence on the test data. Table 1 shows the averages of the test loss, DTW distance and the LCS similarity on the test data. A lower test loss indicates that the method has a better capacity of modeling the temporal relationship between the audio channel and the motion channel in the data; a lower DTW distance or a higher LCS similarity indicates the generated animation is closer to the ground truth (motion capture data).

Results. As mentioned in Section 3.2, the collected audiovisual dataset S consists of 108372 segments. In each experiment, 80% (86697) segments are randomly selected as the training data and the rest 20% (21675) ones are taken as the test data. The quantitative results are reported in Table 1, referring to the averages of 30 experiments.

As shown in this table, our method outperforms all the baseline methods in terms of both the DTW distance and the LCS similarity. Furthermore, our method achieves a smaller test loss and a smaller DTW distance than $M^{\overline{GAN}}$, $M^{\overline{AttB}}$ and $M^{\overline{Res}}$; Also, our method achieves a higher LCS similarity than $M^{\overline{GAN}}$, $M^{\overline{AttB}}$ and $M^{\overline{Res}}$. In other words,

TABLE 1
Quantitative comparision among our method, the baselines, and the ablation study versions. The used quantitative metrics include the average of test loss, DTW distance (DTW, $\times 10e4$) and LCS similarity (LCS) on the test data. The check marks refer to the employed operations in the method.

	Model				Test Loss	DTW	LCS
Baselines	LSTM [9]				.0608	.3534	.0561
	CNN [36]				.0390	.2556	.1045
	CNN+LSTM [42]				.0632	.2687	.1070
	TCN [71]				.0438	.2593	.1033
	Unet [12]				.0184	.2304	.1218
	R2Unet [85]				.0376	.3186	.0999
	Music2Body [70]				.0296	.2543	.1178
		ResB	**AttB**	**GAN**			
Ablation Study	$M^{\overline{GAN}}$	√	√		.0138	.2032	.1314
	$M^{\overline{Att}}$	√		√	.0132	.2046	.1325
	$M^{\overline{Res}}$		√	√	.0179	.2057	.1330
Ours		√	√	√	**.0118**	**.2016**	**.1358**

our method performs better than all the ablation study versions, which implies that each of the supervision of the discriminator in the GAN framework, the Res block, and the attention block makes a positive contribution to the overall performance of our method. Figure 10 shows the synthetic results of each block. The vanilla Unet produces playing motions with invalid gestures and temporal jitter occasionally (the first row). The attention block removes the invalid gestures but still retains the temporal jitter (the second row). The Res block removes both the invalid gestures and temporal jitter while sacrifices the magnitude of action (the third row). The GAN framework leads to larger motions to improve the naturalness (the fourth row).

In order to evaluate the effectiveness of the data augmentation in our method, we compared the performances of our method with and without the data augmentation. Figures 11 and 12 illustrate the training loss, along with the number of training iterations, achieved by our method with and without the data augmentation, respectively. From Figure 11, we can see that in the initial stage of training, the training loss is reduced more slowly by our method with the data augmentation than by our method without the data augmentation. However, as the training progresses, our method with the data augmentation achieves noticeably smaller training loss than our method without the data augmentation. Now if we look into the validation loss comparison in Figure 12, we can see that our method with the data augmentation achieves a substantially smaller validation loss (on the test dataset) than our method without the data augmentation. The above results provide solid evidence that the data augmentation step (Section 4.1) substantially improves the generalization ability of our model.

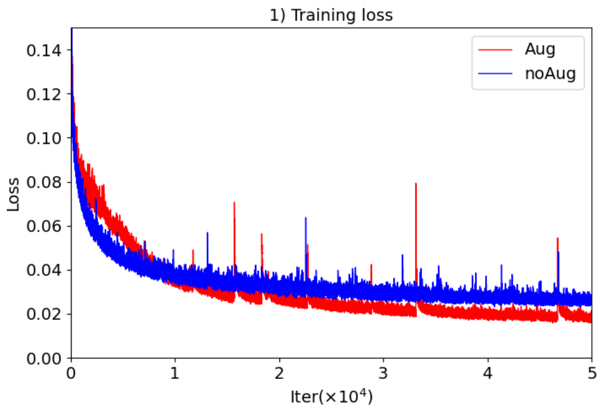


Fig. 11. Comparison of the effect of data augmentation on the training loss in the training stage. “Aug” represents “with data augmentation”, and “noAug” represents “without data augmentation”.

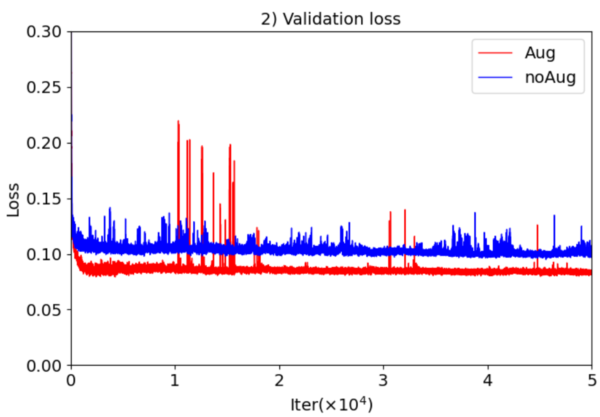


Fig. 12. Comparison of the effect of data augmentation on the validation loss. ‘Aug” represents “with data augmentation”, and “noAug” represents “without data augmentation”.

Table 2 shows the average of test loss, DTW distance and LCS similarity, when our method includes data augmentation or does not include data augmentation. The results show that the case w/ data augmentation results in a smaller test loss (0.0184 vs. 0.0280), a smaller DTW distance (230.4361 vs. 240.2176) than the case w/o data augmentation. In terms of the average LCS similarity, the case w/o data augmentation is very slightly higher (approximately 0.03%) than the case w/ data augmentation. Actually, their results are very close to each other (0.8782 vs. 0.8799). The results in Table 2 further confirm that the data augmentation step positively contributes to the performance of our method.

TABLE 2
Comparison of the quantitative measures for the cases w/ data augmentation and w/o data augmentation. All the numbers reported in this table are achieved after 50,000 iterations.

Model	Test Loss	Avg. DTW	Avg. LCS
w/o data augmentation	0.0280	240.2176	0.8799
w/ data augmentation	**0.0184**	**230.4361**	**0.8782**

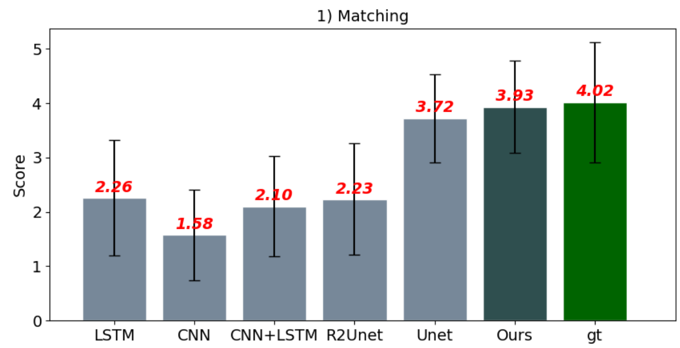


Fig. 13. The average scores and standard deviations of all the comparison methods in the user study in terms of the matching perception.

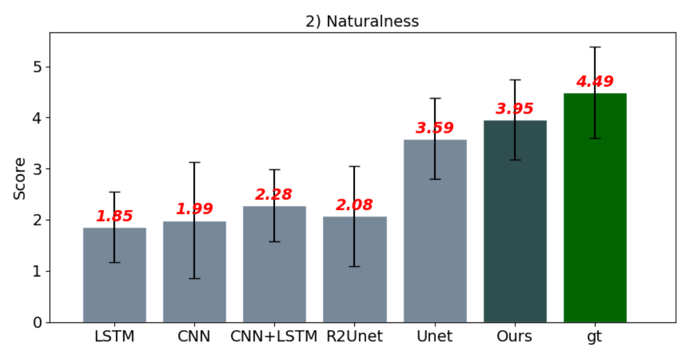


Fig. 14. The average scores and standard deviations of all the comparison methods in the user study in terms of the naturalness perception.

5.2 User Study

We conducted an online user study to compare our method with the aforementioned baseline methods (i.e., LSTM, CNN, CNN+LSTM, Unet, R2Unet) and the ground truth data (abbreviated as “gt”). In other words, a total of 7 different methods (or called conditions) were compared. Specifically, we randomly selected two recorded Guzheng music pieces in the test data. The two music pieces last 27 seconds and 31 seconds, respectively. Then, we used the 7 different methods to generate corresponding character animations based on the 2 Guzheng music inputs. To this end, we generate a total of 14 Guzheng-playing video clips (2 × 7 = 14) for our user study, and we also created an online website for participants to view and rate them. To counterbalance the potential influence from the clip order,

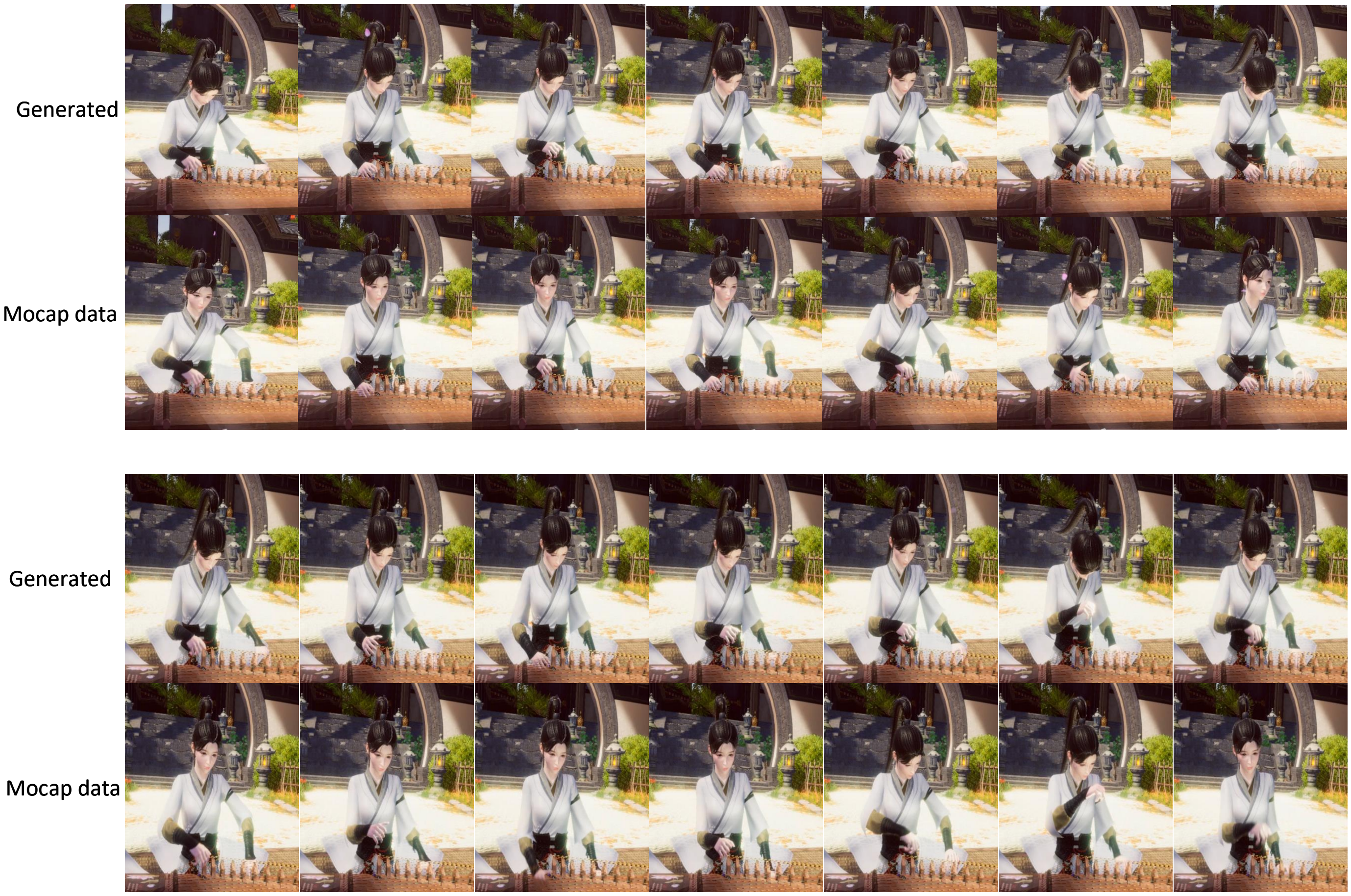


Fig. 15. Snapshots from the generated and ground truth (motion caption) animation trajectories accompanied with the same music.

TABLE 3
The results of the paired Mann-Whitney U test for the obtained matching scores and naturalness scores by all the methods.

	ours		**ground truth**	
	matching	naturalness	matching	naturalness
LSTM [9]	U = 5049.0; p = 4.96e-37	U = 1754.5; p = 1.22e-55	U = 1916.0; p = 1.93e-22	U = 457.5; p = 2.32e-37
CNN [36]	U = 1986.5; p = 2.61e-52	U = 4586.0; p = 8.00e-42	U = 828.0; p = 8.64e-33	U = 824.0; p = 3.46e-33
CNN+LSTM [42]	U = 3868.0; p = 8.64e-52	U = 3138.0; p = 3.24e-48	U = 1460.0; p = 2.24e-26	U = 729.5; p = 1.84e-34
Unet [12]	U = 17140.5; p = 3.93e-3	U = 14702.0; p = 3.86e-7	U = 5330.5; p = 2.06e-3	U = 2741.0; p = 7.94e-17
R2Unet [85]	U = 4535.0; p = 2.03e-39	U = 3315.0; p = 6.09e-47	U = 1848.5; p = 2.80e-23	U = 820.5; p = 3.13e-33
ground truth	U = 18279.0; p = 5.25e-2	U = 11483.5; p = 1.17e-14	-	-
ours	-	-	U = 18279.0; p = 5.25e-2	U = 11483.5; p = 1.17e-14

we randomly select and display video clips for each participant.

58 participants aged 20-45 years were invited to participate in our user study. Their average age is 32.12 and the standard deviation is 6.20. Four volunteers are professional music artists and the rest are amateurs in Guzheng. After watching each video clip, they were instructed to use a 5-point Likert scale to rate the matching between music and animation and rate the naturalness of the animation. Before the experiment starts, they were instructed that the "matching" refers to the synchronization between the played music and animation, and the "naturalness" refers to the quality of the visual animation.

Furthermore, they were particularly instructed to ignore minor visual artifacts (errors) from the inaccurate positions of the fingers that touch the instrument, since the size and position of the virtual Guzheng instrument are not the same as those of the real Guzheng instrument used in our data recording step and also the collected finger motions have a low accuracy (as mentioned in Section 3.1). Each participant took about 25 to 30 minutes to complete the user study.

Figure 13 and Figure 14 show the average scores and standard deviations of the ratings obtained by all the methods, in terms of the perception on matching and naturalness, respectively. As clearly observed in the two figures, our method can soundly outperform all the baseline methods in terms of both the matching and naturalness. Compared to the ground-truth data (the "gt" case in the two figures), the averaged ratings obtained by our method are not good as but reasonably close to the ground-truth animation (driven by the recorded motion capture data), in terms of both matching (3.93 vs. 4.02) and naturalness (3.95 vs. 4.49).

Fig. 16. Snapshots of the virtual character and a real musician playing the same music. The music in the musician video is extracted as input to our music-to-body generator and then the generated animations are used to drive the virtual character.

Based on the obtained user rating data, we also performed paired Mann-Whitney U test [89] between different conditions. The statistical results are shown in Table 3 (for the matching perception and naturalness perception). As shown in Table 3 and two figures (Figure 13 and Figure 14), our method outperform all the baseline methods in a statistically significant way, in terms of the perception of both matching and naturalness. On the other hand, in terms of naturalness, our method is still significantly inferior to the ground-truth motion data. This indicates that there is still room for our method to improve to produce more realistic and natural Guzheng-playing animations.

In order to show the qualitative results conveniently, we employ a professional technical artist to build a virtual character. Figure 15 shows the comparison of some selected frames from the animations generated by our method and from the generated ground-truth animations (driven by recorded motion capture data). Figure 16 shows the comparison of some selected frames from the animations generated by our method and from the recorded ground-truth video of Guzheng-playing (acquired in our data capture stage). As shown in the two figures, the results by our method are visually similar to the ground truth (both the generated ground-truth animation and the recorded ground-truth video). For the generated animation results in this user study, please refer to the supplemental demo video.

6 Discussion and Conclusion

In this paper we present a novel GAN-based framework to learn the temporal relationship between Guzheng music and the upper body motion of Guzheng-playing. Given novel Guzheng music as the input, our trained model can automatically generate the corresponding natural and realistic Guzheng-playing character animations. Specifically, at the training step, besides a multi-scale patch discriminator, we also propose a music-to-motion generator that is supervised with both the joint rotation loss and the end-effector position loss on top of the conventional GAN loss. In addition, attention blocks and “upsample+1D conv” layers are also designed to refine the generated motion trajectories.

For this work, we specifically capture a large scale, Guzheng-playing audiovisual dataset using our in-house motion capture setup. We also introduce a novel data augmentation step to increase the generalizability of our dataset and thus our trained model. The effectiveness of our proposed data augmentation was validated by our quantitative evaluation. We plan to release this unique audiovisual dataset for research purpose in the research community after the work is published.

We also conducted extensive studies, including both quantitative and qualitative (via a user study) experiments, to compare our method with five state of the art methods as well as the ground-truth animation. Our results validate that our method can outperform all the five state-of-the-art methods in a statistically significant way. Also, via an ablation study, we confirm that each of our modules (i.e., the data augmentation, the Res blocks, the attention blocks, and the GAN-based module) makes a positive contribution to the overall performance of our method.

Our current approach only utilizes the recorded data of a single artist. In the future, we plan to record Guzheng-playing data of more artists, and then design effective algorithms to model artist-specific styles of Guzheng-playing and also to create new styles by smoothing transferring from one artist-specific style to another. We will improve the recording pipeline of finger motions and augment the accuracy of the recorded finger motions, to synthesize high-quality finger playing animations. Moreover, due to the gaps in scale and position between the virtual and the real musical instruments, the mocap and generated data cannot reflect a good performance on touching the instrument. To address this issue, we plan to make efforts on developing a new animation generation which is adaptive to the virtual instrument. In addition, we will collect and release other audiovisual data playing other musical instruments for the academic community and explore many widely-open research problems regarding the motion of other musical instrument-playing, e.g., a unified framework of synthesizing playing animations for various musical instruments.

REFERENCES

[1] S. Dahl and A. Friberg, "Expressiveness of musician's body movements in performances on marimba." *Lecture Notes in Computer Science*, vol. 2915, pp. 479–486, 2004.

[2] Y. Zhu, A. Ramakrishnan, B. Hamann, and M. Neff, "A system for automatic animation of piano performances," *Computer Animation and Virtual Worlds*, vol. 24, 09 2013.

[3] S. Dahl and A. Friberg, "Visual perception of expressiveness in musicians' body movements," *Music Perception: An Interdisciplinary Journal*, vol. 24, no. 5, pp. 433–454, 2007.

[4] J. W. Davidson, "Visual perception of performance manner in the movements of solo musicians," *Psychology of music*, vol. 21, no. 2, pp. 103–113, 1993.

[5] J. Sundberg and B. Brunson, "A fuzzy analyzer of emotional expression in music performance and body motion," in *Proceedings of Music and Music Science*, vol. 10, 2004, pp. 28–30.

[6] G. Widmer, S. Flossmann, and M. Grachten, "Yqx plays chopin," *AI magazine*, vol. 30, no. 3, pp. 35–35, 2009.

[7] C. Cadoz and M. M. Wanderley, "Gesture - Music," in *Trends in Gestural Control of Music*, 2000.

[8] M. R. Thompson and G. Luck, "Exploring relationships between pianists' body movements, their expressive intentions, and structural elements of the music," *Musicae Scientiae*, vol. 16, no. 1, pp. 19–40, 2012.

[9] E. Shlizerman, L. Dery, H. Schoen, and I. Kemelmacher-Shlizerman, "Audio to body dynamics," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 7574–7583.

[10] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[11] A. F. M. A. Dauphin, Yann N. and D. Grangier., "Language modeling with gated convolutional networks," in *International Conference on Machine Learning*, 2017, pp. 933–941.

[12] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*. Springer, 2015, pp. 234–241.

[13] M. Z. Alom, M. Hasan, C. Yakopcic, T. M. Taha, and V. K. Asari, "Recurrent residual convolutional neural network based on u-net (r2u-net) for medical image segmentation," *arXiv preprint arXiv:1802.06955*, 2018.

[14] O. Oktay, J. Schlemper, L. L. Folgoc, M. Lee, M. Heinrich, K. Misawa, K. Mori, S. McDonagh, N. Y. Hammerla, B. Kainz *et al.*, "Attention u-net: Learning where to look for the pancreas," *arXiv preprint arXiv:1804.03999*, 2018.

[15] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," in *IEEE conference on computer vision and pattern recognition*, 2017, pp. 1125–1134.

[16] S. Ginosar, A. Bar, G. Kohavi, C. Chan, A. Owens, and J. Malik, "Learning individual styles of conversational gesture," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 3497–3506.

[17] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

[18] M. Mirza and S. Osindero, "Conditional generative adversarial nets," *arXiv preprint arXiv:1411.1784*, 2014.

[19] L. Ma and Z. Deng, "Real-time facial expression transformation for monocular rgb video," in *Computer Graphics Forum*, vol. 38, no. 1. Wiley Online Library, 2019, pp. 470–481.

[20] Z. Cao, G. Hidalgo, T. Simon, S.-E. Wei, and Y. Sheikh, "Openpose: realtime multi-person 2d pose estimation using part affinity fields," *IEEE transactions on pattern analysis and machine intelligence*, vol. 43, no. 1, pp. 172–186, 2019.

[21] M. Liao, S. Zhang, P. Wang, H. Zhu, and R. Yang, "Personalized speech2video with 3d skeleton regularization and expressive body poses," *arXiv preprint arXiv:2007.09198*, 2020.

[22] M. Brand, "Voice puppetry," in *Conference on Computer graphics and interactive techniques*, 1999, pp. 21–28.

[23] C. Busso, Z. Deng, and S. Narayanan, "Natural head motion synthesis driven by acoustic prosodic features," *Journal of Visualization and Computer Animation*, vol. 16, no. 3-4, pp. 283–290, 2005.

[24] Z. Deng and U. Neumann, "efase: expressive facial animation synthesis and editing with phoneme-isomap controls," in *Symposium on Computer Animation*, 2006, pp. 251–260.

[25] C. Busso, Z. Deng, M. Grimm, U. Neumann, and S. Narayanan, "Rigid head motion in expressive speech animation: Analysis and synthesis," *IEEE Trans. on Audio, Speech and Language Processing*, vol. 15, no. 3, pp. 1075–1086, 2007.

[26] S. Mariooryad and C. Busso, "Generating human-like behaviors using joint, speech-driven models for conversational agents," *IEEE Transaction on Audio, Speech and Language Processing*, vol. 20, no. 8, pp. 2329–2340, 2012.

[27] B. H. Le, X. Ma, and Z. Deng, "Live speech driven head-and-eye motion generators," *IEEE TVCG*, vol. 18, pp. 1902–1914, 2012.

[28] Y. Ding, M. Radenen, T. Artières, and C. Pelachaud, "Eyebrow motion synthesis driven by speech," in *Workshop Affect, Compagnon Artificiel, Interaction (WACAI)*, 2012, pp. 103–110.

[29] Y. Ding, M. Radenen, T. Artières, and C. Pelachaud, "Speech-driven eyebrow motion synthesis with contextual Markovian models." in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2013, pp. 3756–3760.

[30] Y. Ding, C. Pelachaud, and T. Artières, "Modeling multimodal behaviors from speech prosody," in *Intelligent Virtual Agents*, 2013, pp. 217–228.

[31] Y. Ding, K. Prepin, J. Huang, C. Pelachaud, and T. Artières, "Laughter animation synthesis," in *Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems*, 2014, pp. 773–780.

[32] Y. Ding and C. Pelachaud, "Lip animation synthesis: a unified framework for speaking and laughing virtual agent." in *AVSP*, 2015, pp. 78–83.

[33] F. Pecune, M. Mancini, B. Biancardi, G. Varni, Y. Ding, C. Pelachaud, G. Volpe, and A. Camurri, "Laughing with a virtual agent," in *AAMAS*, 2015, pp. 1817–1818.

[34] H. Van Welbergen, Y. Ding, K. Sattler, C. Pelachaud, and S. Kopp, "Real-time visual prosody for interactive virtual agents," in *International Conference on Intelligent Virtual Agents*, 2015, pp. 139–151.

[35] P. Edwards, C. Landreth, E. Fiume, and K. Singh, "Jali: an animator-centric viseme model for expressive lip synchronization," *ACM Transactions on Graphics (TOG)*, vol. 35, no. 4, pp. 1–11, 2016.

[36] T. Karras, T. Aila, S. Laine, A. Herva, and J. Lehtinen, "Audio-driven facial animation by joint end-to-end learning of pose and emotion," *ACM Transactions on Graphics (TOG)*, vol. 36, no. 4, pp. 1–12, 2017.

[37] S. Taylor, T. Kim, Y. Yue, M. Mahler, J. Krahe, A. G. Rodriguez, J. Hodgins, and I. Matthews, "A deep learning approach for generalized speech animation," *ACM Transactions on Graphics (TOG)*, vol. 36, no. 4, pp. 1–11, 2017.

[38] M. Mancini, B. Biancardi, F. Pecune, G. Varni, Y. Ding, C. Pelachaud, G. Volpe, and A. Camurri, "Implementing and evaluating a laughing virtual character," *ACM Transactions on Internet Technology (TOIT)*, vol. 17, no. 1, pp. 1–22, 2017.

[39] H. X. Pham, S. Cheung, and V. Pavlovic, "Speech-driven 3d facial animation with implicit emotional awareness: a deep learning approach," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2017, pp. 80–88.

[40] Y. Ding, J. Huang, and C. Pelachaud, "Audio-driven laughter behavior controller," *IEEE Transactions on Affective Computing*, vol. 8, no. 4, pp. 546–558, 2017.

[41] Y. Ding, T. Artières, and C. Pelachaud, "Laughter animation generation," *Handbook of Human Motion*, 2017.

[42] H. X. Pham, Y. Wang, and V. Pavlovic, "End-to-end learning for 3d facial animation from speech," in *ACM International Conference on Multimodal Interaction*, 2018, pp. 361–365.

[43] D. Cudeiro, T. Bolkart, C. Laidlaw, A. Ranjan, and M. J. Black, "Capture, learning, and synthesis of 3d speaking styles," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 10 101–10 111.

[44] J. Chen, Y. Liu, Z. Zhang, C. Fan, and Y. Ding, "Text-driven visual prosody generation for embodied conversational agents," in *ACM International Conference on Intelligent Virtual Agents*, 2019, pp. 108–110.

[45] L. Li, S. Wang, Z. Zhang, Y. Ding, Y. Zheng, X. Yu, and C. Fan, "Write-a-speaker: Text-based emotional and rhythmic talking-head generation," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 3, 2021, pp. 1911–1920.

[46] S. Wang, L. Li, Y. Ding, C. Fan, and X. Yu, "Audio2head: Audio-driven one-shot talking-head generation with natural head motion," *arXiv e-prints*, pp. arXiv–2107, 2021.

[47] Z. Zhang, L. Li, Y. Ding, and C. Fan, "Flow-guided one-shot talking face generation with a high-resolution audio-visual dataset," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 3661–3670.

[48] Z. Deng and J. Noh, "Computer facial animation: A survey," in *Data-driven 3D facial animation*. Springer, 2008, pp. 1–28.

[49] J. P. Lewis, K. Anjyo, T. Rhee, M. Zhang, F. H. Pighin, and Z. Deng, "Practice and theory of blendshape facial models." *Eurographics (State of the Art Reports)*, vol. 1, no. 8, p. 2, 2014.

[50] Y. Ferstl and R. McDonnell, "Investigating the use of recurrent motion modelling for speech gesture generation," in *International Conference on Intelligent Virtual Agents*, 2018, pp. 93–98.

[51] T. Kucherenko, D. Hasegawa, G. E. Henter, N. Kaneko, and H. Kjellström, "Analyzing input and output representations for speech-driven gesture generation," in *ACM International Conference on Intelligent Virtual Agents*, 2019, pp. 97–104.

[52] A. Jin, Q. Deng, Y. Zhang, and Z. Deng, "A deep learning-based model for head and eye motion generation in three-party conversations," *ACM on Computer Graphics and Interactive Techniques*, vol. 2, no. 2, pp. 1–19, 2019.

[53] I. Rodriguez, J. M. Martínez-Otzeta, I. Irigoien, and E. Lazkano, "Spontaneous talking gestures using generative adversarial networks," *Robotics and Autonomous Systems*, vol. 114, pp. 57–65, 2019.

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

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

[56] S. Fukayama and M. Goto, "Automated choreography synthesis using a gaussian process leveraging consumer-generated dance motions," in *Proceedings of the 11th Conference on Advances in Computer Entertainment Technology*, 2014, pp. 1–6.

[57] A. Berman and V. James, "Kinetic imaginations: exploring the possibilities of combining ai and dance," in *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.

[58] 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*, 2020, pp. 744–752.

[59] J. L. S.-H. Z. Y.-C. G. W. Z. S.-M. H. Chen Kang, Zhipeng Tan, "Choreomaster : Choreography-oriented music-driven dance synthesis," *ACM Transactions on Graphics (TOG)*, vol. 40, no. 4, 2021.

[60] 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, 2011.

[61] O. Alemi, J. Françoise, and P. Pasquier, "Groovenet: Real-time music-driven dance movement generation using artificial neural networks," *Workshop on Machine Learning for Creativity, 23rd ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, vol. 8, no. 17, p. 26, 2017.

[62] T. Tang, J. Jia, and H. Mao, "Dance with melody: An lstm-autoencoder approach to music-oriented dance synthesis," in *ACM international conference on Multimedia*, 2018, pp. 1598–1606.

[63] H.-Y. Lee, X. Yang, M.-Y. Liu, T.-C. Wang, Y.-D. Lu, M.-H. Yang, and J. Kautz, "Dancing to music," in *Advances in Neural Information Processing Systems*, 2019, pp. 3581–3591.

[64] J. Lee, S. Kim, and K. Lee, "Automatic choreography generation with convolutional encoder-decoder network." in *ISMIR*, 2019, pp. 894–899.

[65] R. Huang, H. Hu, W. Wu, K. Sawada, M. Zhang, and D. Jiang, "Dance revolution: Long-term dance generation with music via curriculum learning," *arXiv preprint arXiv:2006.06119*, 2020.

[66] B. Wallace, C. P. Martin, J. Torresen, and K. Nymoen, "Towards movement generation with audio features," *arXiv preprint arXiv:2011.13453*, 2020.

[67] B. Li, A. Maezawa, and Z. Duan, "Skeleton plays piano: Online generation of pianist body movements from midi performance." in *ISMIR*, 2018, pp. 218–224.

[68] J.-W. Liu, H.-Y. Lin, Y.-F. Huang, H.-K. Kao, and L. Su, "Body movement generation for expressive violin performance applying neural networks," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 3787–3791.

[69] A. Bogaers, Z. Yumak, and A. Volk, "Music-driven animation generation of expressive musical gestures," in *Companion Publication of the 2020 International Conference on Multimodal Interaction*, 2020, pp. 22–26.

[70] H.-K. Kao and L. Su, "Temporally guided music-to-body-movement generation," in *Proceedings of the 28th ACM International Conference on Multimedia*, 2020, pp. 147–155.

[71] S. Bai, J. Z. Kolter, and V. Koltun, "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling," *arXiv preprint arXiv:1803.01271*, 2018.

[72] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[73] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *IEEE conference on computer vision and pattern recognition*, 2017, pp. 4700–4708.

[74] A. Creswell, T. White, V. Dumoulin, K. Arulkumaran, B. Sengupta, and A. A. Bharath, "Generative adversarial networks: An overview," *IEEE Signal Processing Magazine*, vol. 35, no. 1, pp. 53–65, 2018.

[75] P.-E. Aguera, K. Jerbi, A. Caclin, and O. Bertrand, "Elan: A software package for analysis and visualization of meg, eeg, and lfp signals," *Intell. Neuroscience*, vol. 2011, pp. 5:1–5:11, Jan. 2011.

[76] J.-c. Chou, C.-c. Yeh, H.-y. Lee, and L.-s. Lee, "Multi-target voice conversion without parallel data by adversarially learning disentangled audio representations," *arXiv preprint arXiv:1804.02812*, 2018.

[77] D. Pavllo, C. Feichtenhofer, M. Auli, and D. Grangier, "Modeling human motion with quaternion-based neural networks," *International Journal of Computer Vision*, vol. 128, no. 4, pp. 855–872, 2020.

[78] D. Pavllo, D. Grangier, and M. Auli, "Quaternet: A quaternion-based recurrent model for human motion," *arXiv preprint arXiv:1805.06485*, 2018.

[79] C. A. Hall and W. W. Meyer, "Optimal error bounds for cubic spline interpolation," *Journal of Approximation Theory*, vol. 16, no. 2, pp. 105–122, 1976.

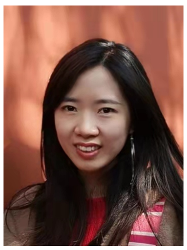
[80] A. Odena, V. Dumoulin, and C. Olah, "Deconvolution and checkerboard artifacts," *Distill*, 2016. [Online]. Available: http://distill.pub/2016/deconv-checkerboard

[81] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

[82] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *International conference on machine learning*, 2010, pp. 807–814.

[83] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[84] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," 2017.
[85] M. Z. Alom, M. Hasan, C. Yakopcic, and T. M. Taha, "Inception recurrent convolutional neural network for object recognition," *arXiv preprint arXiv:1704.07709*, 2017.
[86] M. Liang and X. Hu, "Recurrent convolutional neural network for object recognition," in *IEEE conference on computer vision and pattern recognition*, 2015, pp. 3367–3375.
[87] F. Itakura, "Minimum prediction residual principle applied to speech recognition," *IEEE Transactions on acoustics, speech, and signal processing*, vol. 23, no. 1, pp. 67–72, 1975.
[88] D. S. Hirschberg, "Algorithms for the longest common subsequence problem," *Journal of the ACM (JACM)*, vol. 24, no. 4, pp. 664–675, 1977.
[89] P. E. McKnight and J. Najab, "Mann-whitney u test," *The Corsini encyclopedia of psychology*, pp. 1–1, 2010.

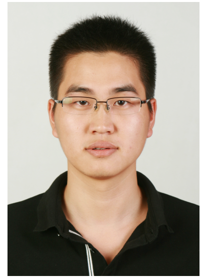


Jiali Chen received the B.E. degree from Zhejiang University in 2012, and the M.S. degree from National Tsing Hua Uniersity in 2014. She is currently a senior researcher in Netease Fuxi AI Lab, Hangzhou, China. She is current research interests include compution vision, artificial intelligence, face and motion generation.

Changjie Fan is the Director of NetEase FUXI AI Lab. He received his doctor's degree in Computer Science from University of Science and Technology of China. His research interest is in machine learning, including multiagent systems, deep reinforcement learning, game theory and knowledge discovery.

Zhimeng Zhang received the B.E. degree in automation and the M.S. degree in pattern recognition and intelligent system from Shandong University, Shandong, China, in 2016 and 2019, respectively. He is currently a research scientist working with Netease Fuxi AI Lab, Hangzhou, China. His current research interests include compution vision, animation generation and talking face generation.

Gongzheng Li is a Deep learning RD engineer at NetEase Fuxi AI Lab. He graduated with a master's degree in software engineering from University of Science and Technology of China. His work mainly focus on engineering and optimization for computer vision and natural language processing, including the acceleration of inference and training, model compression, neural architecture search.



Zeng Zhao is a RD expert of MLSYS in Fuxi Lab of NetEase. He received his PhD degree in the School of Computer Science from University of Science and Technology of China. His research focuses on efficient deep learning computing and MLSys.

Zhigang Deng is Moores Professor of Computer Science at University of Houston, Texas, USA. His research interests include computer graphics, computer animation, virtual humans, human computer conversation, and robotics. He earned his Ph.D. in Computer Science at the Department of Computer Science at the University of Southern California in 2006. Prior that, he also completed B.S. degree in Mathematics from Xiamen University (China), and M.S. in Computer Science from Peking University (China). Besides serving as the conference general co-chair for both CASA 2014 and SCA 2015, he has been an Associate Editor for IEEE Transactions on Visualization and Computer Graphics, Computer Graphics Forum, Computer Animation and Virtual Worlds Journal, etc. He is a senior member of ACM and a senior member of IEEE.

Yu Ding is currently an artificial intelligence expert at Netease Fuxi AI Lab, Hangzhou, China. His research interests include deep learning, image and video processing, talking-head generation, animation generation, multimodal computing, affective computing, nonverbal communication (face, gaze, and gesture), and embodied conversational agent. He received Ph.D. degree in Computer Science (2014) at Telecom Paristech in Paris (France), M.S. degree in Computer Science at Pierre and Marie Curie University (France), and B.S. degree in Automation at Xiamen University (China).