Automatic Translation of Music-to-Dance for In-Game Characters

Supplementary Material

Yinglin Duan[1]*, Tianyang Shi[1]*, Zhipeng Hu[1,2], Zhengxia Zou[3], Changjie Fan[1], Yi Yuan[1]†, Xi Li[2]

[1]NetEase Fuxi AI Lab, [2]Zhejiang University, [3]University of Michigan, Ann Arbor

1 Definitions of Music-Dance

In this section, we give more details on the music phrases and dance phrases with the corresponding music and dance theories used in this work.

1.1 Choreography theory

In this work, we mainly follow the choreography theory derived from Doris Humphrey – one of the greatest American choreographers. She suggests matching music phrases and dance phrases to choreograph, rather than simply counting the beats. Humphrey has experimentally verified her theory [5]. Under the guidance of her theory, we focus on dance phrases rather than beats only. Besides, Eleanor Metheny also shared similar ideas "Choreographers do not make dances out of words. They compose them in the vocabulary of movement" [1].

Base on the above theories, senior choreographers usually follow three steps to choreograph for a piece of music: 1. collect dance movements from daily life, 2. design specific dance phrases based on music style and rhythm, 3. organize these dance phrases according to the music structural[1]. Our work is also inspired by this pipeline.

1.2 Dance phrase

According to Encyclopedia Britannica, the dance phrase is defined as a series of movements bound together by a physical impulse whose end and beginning are clearly recognizable. There are various factors that can help recognize a phrase, e.g. a movement may start with a sharp force, gradually slow down, and finally stop. The basic dance step sequence in Waltz, i.e. Waltz Box Step, is a typical dance phrase defined above, which contains six movements and draws a square on the ground.

In this work, our dance experts design 1,101 dance phrases by considering basic dance movements, re-usability, type diversity and style consistency with the game. When assigning music phrases for each dance phrase, they mainly consider the factors such as the fluency, rhythm, consistency, emotion, and style on both music and dance phrases.

1.3 Music phrase

A music phrase can be regarded as a separate musical entity within the melodic line [7]. Therefore, we separate a music phrase based on the following three criteria:

1. A musical rest. A music phrase should have some form of semi-termination or termination.
2. Long tone at the end of the phrase. At the end of the phrase, the melody no longer fluctuates up and down.
3. Melody or rhythm repetition: Sometimes there is no long note at the end of a phrase, and there is no rest, but a phrase can also be defined when the melody or rhythm before and after has is a repetitive form.

As mentioned in the main body and shown in Fig. 1, we segment music phrases by three steps:

*These authors contributed equally to this work.

†Corresponding author.

[1]Please refer to `https://www.britannica.com/art/dance/The-three-phase-choreographic-process#ref392649` for more choreography details.

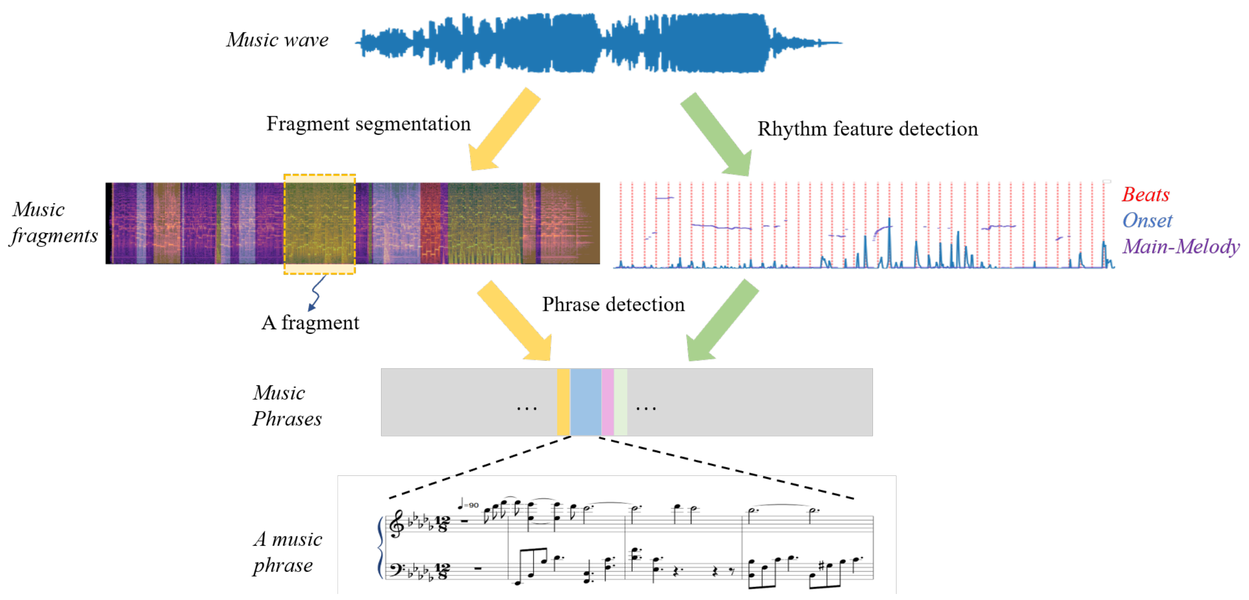

Figure 1: The processing pipeline of music phrase segmentation. We firstly segment music to fragments, and then extract features from music fragments. Finally, we split music phrases based these musical features.

Step 1: Coarse segmentation. We analyze the music structure by using spectral clustering and segment music into long fragments [9], where music slices with similar patterns will be gathered together (according to criterion 3).

Step 2: Rhythm detection. We detect beats using librosa and extract the main-melody by a deep learning method [3], where the main-melody can be used to recognize the musical rest (according to criterion 1 & 2).

Step 3: Phrase detection: For each phrase, we start from the end of the last one, merge detected beats at least 6 ones, and end at a breaking point of a long fragment (from step 1) or a music rest (from step 2) [6].

2 Details of Network Configuration

In this section, we list the configurations of all networks mentioned in our main paper, i.e. the encoder E, the predictor P, the 2D-decoder D_1 and two 1D-decoders D_2 & D_3. Our networks are implemented under PyTorch deep learning framework [10].

2.1 The configuration of Encoder E

A detailed configuration of ResNet50-based encoder E is listed in Table 1. The input size of the Encoder E is 128×128 pixels, where the Mel Spectrogram is therefore resized on the time dimension. The outputs of E contain a temporal feature $\boldsymbol{f_t} \in \mathbb{R}^{512\times4\times1}$ and an embedding $\boldsymbol{u} \in \mathbb{R}^{512\times1\times1}$ from *feature layer* and *embedding layer*.

Specifically, in a $c \times w \times w/s$ Convolution / Deconvolution layer, c denotes the number of filters, $w \times w$ denotes the filter's size and s denotes the filter's stride. In a $w \times w/s$ Maxpool layer, w denotes the pooling window size, and s denotes the pooling stride. In an n/s Bottleneck block [2], n denotes the number of planes, and s denotes the block's stride. In an (h, w) AdaptiveAvgPool2d layer, h and w denote the output dimension of height and width, and "None" means the size will be the same as the input.

	Layer	**Component**	**Configuration**	**Feature Size**
Encoder E	Conv_1	Conv2d + BN2d + ReLU	64x7x7 / 2	64x64
	MaxPool	MaxPool	3x3 / 2	32x32
	Conv_2	3 x Bottleneck	64 / 1	32x32
	Conv_3	4 x Bottleneck	128 / 2	16x16
	Conv_4	6 x Bottleneck	256 / 2	8x8
	Conv_5	3 x Bottleneck	512 / 2	4x4
	Conv_6	Conv2d	2048x1x1 / 1	4x4
	feature	AdaptiveAvgPool2d	(None, 1)	4x1
	embedding	AdaptiveAvgPool2d	(1, None)	1x1

Table 1: A detailed configuration of the Encoder E.

2.2 The configuration of Decoder D_1

A detailed configuration of Decoder D_1 is listed in Table 2. The input of D_1 is the embedding $\boldsymbol{u}$ with the length 512, and the output is reconstructed Mel Spectrogram with the size of 128×128 pixels.

	Layer	Component	Configuration	Feature Size
Decoder D_1	Layer_1	ConvTranspose2d + BN2d + ReLU	512x4x4 / 1	4x4
	Layer_2	ConvTranspose2d + BN2d + ReLU	512x4x4 / 2	8x8
	Layer_3	ConvTranspose2d + BN2d + ReLU	256x4x4 / 2	16x16
	Layer_4	ConvTranspose2d + BN2d + ReLU	256x4x4 / 2	32x32
	Layer_5	ConvTranspose2d + BN2d + ReLU	128x3x3 / 1	32x32
	Layer_6	ConvTranspose2d + BN2d + ReLU	128x4x4 / 2	64x64
	Layer_7	ConvTranspose2d + BN2d + ReLU	64x3x3 / 1	64x64
	Layer_8	ConvTranspose2d	1x4x4 / 2	128x128

Table 2: A detailed configuration of the Decoder D_1.

2.3 The configuration of Decoder D_2 and D_3

Detailed configurations of the Decoder D_2 and D_3 are listed in Table 2. The input of D_2 and D_3 is the temporal feature $\boldsymbol{f_t}$, and the output is the reconstructed Main-Melody and Rhythm with the length 128. Since rhythm prediction can be considered as a binary classification problem, we further add a sigmoid function at the end of the Decoder in this task. Similar to the above tables, in a $c \times w/s$ of 1D-Convolution / 1D-Deconvolution layer, c denotes the number of filters, w denotes the filter's length and s denotes the filter's stride.

	Layer	Component	Configuration	Feature Length
Decoder D_2 & D_3	Layer_1	ConvTranspose1d + ReLU	512x2 / 2	8
	Layer_2	ConvTranspose1d + ReLU	256x2 / 2	16
	Layer_3	ConvTranspose1d + ReLU	128x2 / 2	32
	Layer_4	ConvTranspose1d + ReLU	64x2 / 2	64
	Layer_5	ConvTranspose1d + ReLU	32x2 / 2	128
	Output	Conv1d	1x1 / 1	128

Table 3: A detailed configuration of the Decoders D_2 and D_3.

2.4 The configuration of Predictor P

In our predictor, we adopt three residual attention blocks and two fully connected layers. The detailed configuration is shown in Table 4, the (n, m) of a Linear and a "Res-Att" layer represents that the input and output channel number are n and m respectively, and K is the number of output dance phrases. We follow the ResNet [2] and SENet [4] and set the four fully connected layers in our residual attention blocks (as shown in Fig. 2) are orderly set to "Linear(512,1024)", "Linear(1024,512)", "Linear(512,16)" and "Linear(16,512)".

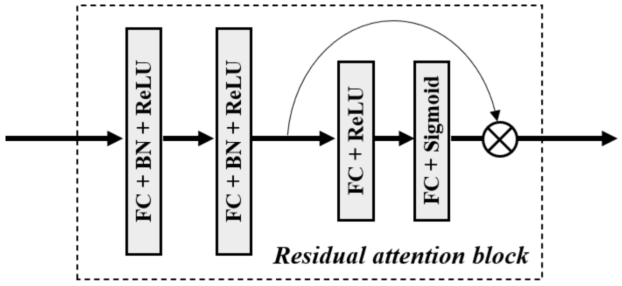

Figure 2: The details of residual attention blocks (Res-Att).

	Layer	Component	Configuration	Feature Channel
Predictor T	Layer_1	Linear	(512, 512)	512
	Layer_2	Res-Att	(512, 512)	512
	Layer_3	Res-Att	(512, 512)	512
	Layer_4	Res-Att	(512, 512)	512
	Output	Linear	(512, K)	K

Table 4: A detailed configuration of the Predictor P.

3 Details of Co-ascent Learning

In this section, we give a detailed description on our co-ascent learning method, which can notably improve the performance of our music-to-dance translation. The algorithm flow of co-ascent learning is shown in Alg. 1.

Algorithm 1: Co-ascent learning algorithm

```
Data: Labeled dataset $D_l$ with $K$ kinds of dance phrases, Unlabeled dataset $D_u$.
Init: Fix the Encoder $E$ and initialize the predictor $\hat{P}$ by training $\hat{P}$ on $D_l$. Calculate the transition matrix $\mathbf{M}_0$
on $D_l$ based on the style of dance phrases. Set the threshold $\tau = 0.9$ and momentum parameter $\alpha = 0.5$;
Var: epoch id $k = 0$;
while Not all of samples in $D_u$ are labeled do
    Run $E$ and $\hat{P}_k$ on $D_u$ and get output probability vector set $P$ of $K$ classes;
    for each temporal adjacent dance phrases $d_{t-1}$ and $d_t$, probability $P(d_{t-1})$ and $P(d_{t-1})$ in $D_u$, $P$ do
        Update $P(d_t)$: $P(d_t) \leftarrow P(d_t)M_k(d_{t-1} \rightarrow d_t)$;
    Get pseudo labels $L$ based on re-scaled $P(d_t)$;
    Initialize $D_u$'s subset $D'_u$ with a null set;
    for each dance movement $d$, label $l$, confidence $P(d)$ in $D_u$, $L$, $P$ do
        if P(d) > $\tau$ then
            Push $d$ and $l$ into $D'_u$;
    Fine-tune the networks $\hat{P}_k$ based on $D_l$ + $D'_u$ and get the new one $\hat{P}_{k+1}$;
    Initialize $\mathbf{M}_{k+1}$ with a zero matrix;
    for each temporal adjacent phrases $d_{t-1}$ and $d_t$, and Top-1 confidences $P(d_{t-1})$ and $P(d_t)$ in $D_u$, $P$ do
        $M_{k+1}(d_{t-1} \rightarrow d_t) = M_k(d_{t-1} \rightarrow d_t) + P(d_{t-1})P(d_t)$
    Update the $\mathbf{M}_k$ with momentum: $\mathbf{M}_{k+1} \leftarrow \alpha\mathbf{M}_k + (1-\alpha)\mathbf{M}_{k+1}$
    Update the epoch id: $k = k + 1$.
Result: Output optimized $\hat{P}^\star$ and $\mathbf{M}^\star$.
```

4 Music and Dance Style Distribution on Datasets

In Fig. 3 and Fig. 4, we show the statistics of music style on our two datasets. The music phrases in these two datasets can be roughly divided into 9 categories. For the dance phrases, we have dance styles including urban, jazz, hip-hop, popping, k-pop, locking, breaking and ACGN dance to match the music styles of our dataset. It is worth mentioning that, based on the choreography theory, the matching of music and dance is flexible, thus our music style and dance style are not one-to-one correspondence.

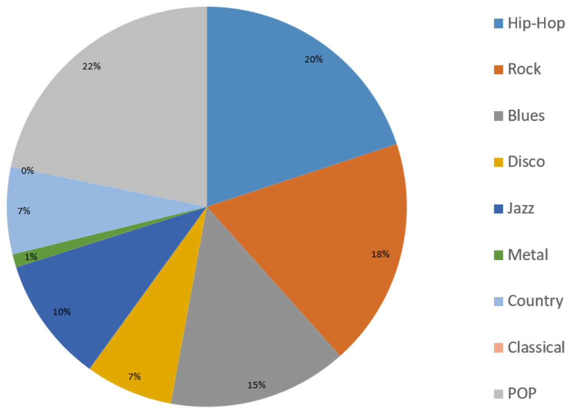

Figure 3: Music style distribution on our labeled dataset.

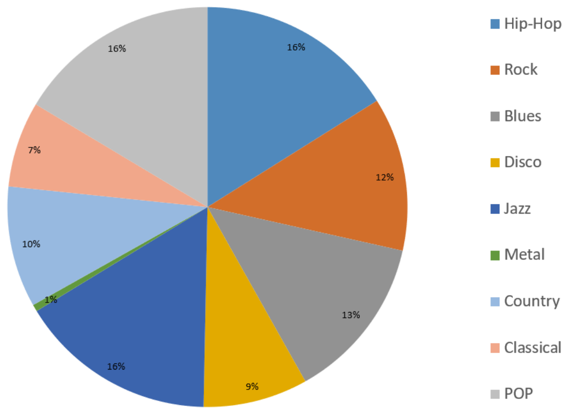

Figure 4: Music style distribution on our large-scale unlabeled dataset.

5 Dance Beat Detection Algorithm

In the quantitative evaluation experiments, we follow Lee *et al.*'s work [8] and implement a dance beat detection algorithm for 2D animations. Considering that the numerical value of animations generated by different methods may not share the same distribution, we further improve the detection in a self-adapting manner.

We follow Lee *et al.* [8] and track dance beats based on two criteria, i.e. $v_p = 0$ and $a_p > 0$, where p represents a joint of the human body. The dance beat detection algorithm contains the following steps:

1. Our algorithm takes in a whole generated motion sequence S with 2D coordinates of each joint.

2. We calculate the velocity and acceleration of the input motion sequence, i.e. $V = \dot{S}$, and $A = \ddot{S}$.

3. We detect coarse beats with an adaptive threshold method. For the time step t, $B_t = (\|V_t\| < T_V) \wedge (\|A_t\| > T_A)$, where the velocity threshold T_V is defined as $T_V = E(V) - \alpha_v \sigma(V)$ and the acceleration threshold is $T_A = \alpha_a T_V$ (α_v and α_a are hyper-parameters across different music-dance methods). Note that since the B is boolean, it represents a beat when it is True.

4. We further compute the beat strength at time t, i.e. $STR_t = (T_V - \|V_t\|) + (\|A_t\| - T_A)$

5. For each beat detected in step 3, we introduce Non-Maximum Suppression to remove false-alarms. For time t and time window w, if $B_t = True$ and $\arg\max\{STR_{t-w/2}, ..., STR_t, ..., STR_{t-w/2}\} = t$, we record this time code t as the one of final beats.

Since the librosa [9] extracts both strong and weak music beats, we adopt a relatively relaxed threshold to track both strong and weak dance beats for a better evaluation ("strong" and "weak" refer to a dance movement with large or small speed and strength). Besides, in order to find the best setting, we further invite an expert jury (with more than 10 years of dance experience) to verify the detected beats on Mocap data with different settings. Finally, we set $\alpha_v = 0.5$, $\alpha_a = \frac{1}{\sqrt{3}}$ and $w = 0.4s$ when evaluating the results of all three methods.

References

[1] L. Ellfeldt. *A primer for choreographers*. Waveland Press, 1988.

[2] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[3] T.-H. Hsieh, L. Su, and Y.-H. Yang. A streamlined encoder/decoder architecture for melody extraction. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 156–160. IEEE, 2019.

[4] J. Hu, L. Shen, and G. Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7132–7141, 2018.

[5] D. Humphrey. The art of making dances. 1959.

[6] T. Jehan. *Creating music by listening*. PhD thesis, 2005.

[7] T. R. Knösche, C. Neuhaus, J. Haueisen, K. Alter, B. Maess, O. W. Witte, and A. D. Friederici. Perception of phrase structure in music. *Human Brain Mapping*, 24(4):259–273, 2005.

[8] 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*, pages 3586–3596, 2019.

[9] 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*, volume 8, 2015.

[10] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.