BRACE: The Breakdancing Competition Dataset for Dance Motion Synthesis

Supplementary material

Davide Moltisanti[1⋆†], Jinyi Wu[2†], Bo Dai[3⋆], and Chen Change Loy[4]

[1] University of Edinburgh, davide.moltisanti@ed.ac.uk
[2] S-Lab, Nanyang Technological University, jinyi002@e.ntu.edu.sg
[3] Shanghai AI Laboratory, daibo@pjlab.org.cn
[4] S-Lab, Nanyang Technological University, ccloy@ntu.edu.sg

1 Supplementary Material Video

We prepared a video to showcase our dataset. For a better viewing we outline the video below. We overlay our keypoints onto the original videos, however we only release keypoints since video copyright belongs to Red Bull, to which we are not affiliated. The video is available at https://youtu.be/-5N6uBDfMoM.

Active dancer detection We show the input/output of our active dancer detection algorithm, i.e. all the poses obtained with the pose estimation models and the automatically detected performing dancer. Notice the abundant number of boxes and keypoints and the accurate detection of the active dancer.

Merging and interpolating sequences We show a sequence of raw automatic poses and the corresponding sequence obtained merging and interpolating automatic and manual keypoints. Notice here the noisy estimations in the raw sequence. Noisy poses were automatically rejected, and a few of them were manually labelled. The final sequence merges good automatic poses and manually annotated keypoints, interpolating the rejected frames we did not annotate due to our discount method. This example shows how our approach is able to produce good sequences with a low annotation cost. We recommend slowing down the video during this section. We added a text colour legend to illustrate which keypoints were rejected or missing, manually annotated or were good automatic keypoints.

Dataset samples We show a few toprock, footwork and powermove segments. We display the interpolated keypoint sequences and play the corresponding audio. Notice the main characteristics that make keypoint extraction challenging: multiple moving cameras, fast lighting changes, occlusions, tangled poses and fast movements. Complex postures and dynamic moves also make BRACE a unique data source for dance motion synthesis.

⋆ Work done while at Nanyang Technological University. † Equal contribution.

Experiment results We compare sequences generated with the tested baselines: Dance Revolution [1], AIST++ [3] and Dancing to Music [2]. The input for testing these models is an audio snippet. In the video we show the corresponding footage segment for reference. Here we provide a few comments about these examples:

- Dance Revolution generates sequences "that never stop", i.e. the dancer is constantly moving throughout the video. While this does not reflect a real breakdancing sequence very well, the continuous motion and the plausible movements help this model achieve the highest pose FID amongst the evaluated baselines.
- AIST++ is able to produce sequences following a more natural tempo, i.e. it generates both fast and slow motions as well as a good mixture of toprock, footwork and powermove segments. However AIST++ also picks up much of the motion induced by the recording setting, i.e. moving/zooming/panning cameras and shot changes, which explains the lower pose FID obtained by the model. Note that due to the required inference seeding, the first 2 seconds of the sequences generated by AIST++ are identical to the ground truth (more details in Section 2).
- Dancing to Music can generate reasonable movements, however these are severely disconnected and consequently the produced sequences do not look realistic. This is due to the automatic decomposition of the dance as we explained in the main text, and is the main cause of the poorer performance of this baseline. This model too picks up camera-induced motion, although to a lesser extent compared to AIST++.

While in the supplementary video we show only a few examples, our findings apply to all the generated sequences we reviewed.

2 Experimental Setup

When testing models on BRACE we ensured training convergence and sensible test output. Here we provide details about our experimental setup. We used the same acoustic features extracted with Librosa [4] following [1] for all baselines.

Dance Revolution [1] For training we set learning rate to 1e-4 and batch size to 32. We evaluated different numbers of layers in the encoder, namely 2, 3 and 4, as well as different sizes for the self-attention window: 15, 25, 100, 200. Training sequence length was set to 200 frames (longer sequences are split accordingly). Pose and frame embedding sizes were set to 34 (number of nodes times 2) and 200 respectively. The number of hidden units for the decoder was set to 1024. The condition step q and λ parameters for the dynamic auto-condition learning scheme were set to 10 and 0.1. Please refer to [1] for details about these hyper-parameters.

Table 1. Extended results obtained with Dance Revolution [1]. Here we show the impact of varying the number of layers in the encoder as well as the size of the self-attention window.

Layers	Attention size	Pose FID ↓	Beat alignment score ↑	Beat DTW cost ↓	Toprock avg	Footwork avg	Powermove avg
2	15	0.5239	0.269	11.89	6.92	54.20	38.88
	25	1.1174	0.273	11.80	6.12	55.82	38.06
	50	0.9663	0.271	11.75	6.52	55.53	37.95
	100	1.0816	0.266	11.70	6.70	55.59	37.71
	200	0.7924	0.265	11.67	11.11	52.13	36.77
3	15	0.5158	0.264	11.88	10.59	51.69	37.72
4	15	0.5723	0.263	11.85	10.25	52.64	37.12

AIST++ [3] For training we follow the implementation details reported in [3], except for the sequence length. In [3] the authors used a seed motion sequence of 120 frames and a music sequence of 240 frames to predict the future 20 frames, where L2 loss is used to optimise the network. However, since their videos are 60 fps while ours are mostly 25 fps, we adjust the seed sequence length to be 50 frames for motion and 100 frames for music, and supervise the L2 loss with the future 8 frames. AIST++ requires a seeding sequence for inference as well. For testing we use the first 50 frames of the motion sequence from the ground truth to generate future frames, hence in our supplementary video it can be observed that the first two seconds of AIST++'s sequences are identical to the ground truth. We exclude these initial 50 frames when calculating the reported metrics.

Dancing to Music [2] We split sequences into segments of 32 frames (roughly 1.2 seconds at 25 fps) using kinematic beats. These were obtained by finding peaks in second order derivatives of keypoint displacements. We used the same approach to find kinematic beats to calculate the beat alignment score and DTW cost, as mentioned in the paper. We use the split segments to train the decomposition network. Specifically, we extract the first four segments from each sequence. These four segments together with the corresponding music features form a training sample for the composition network. The used hyper-parameters are the same as those specified in the official code repository.

3 Dance Revolution extended results

Table 1 reports extended results obtained with Dance Revolution [1]. We trained the model varying the number of layers in the encoder as well as the size of the self-attention window. While the number of layers plays a limited role (best results obtained with 3 layers), the attention window size has a stronger impact on performance (best results with size 15). In particular, we observe that pose FID degrades as the window is enlarged. This parameter controls the receptive field of the encoder, i.e. the temporal neighbourhood each element in the input sequence can attend to. Since our sequences display complex motion with large displacement, a smaller receptive field is best suited at modelling the dance patterns. We do not see noticeable changes in beat alignment score and beat

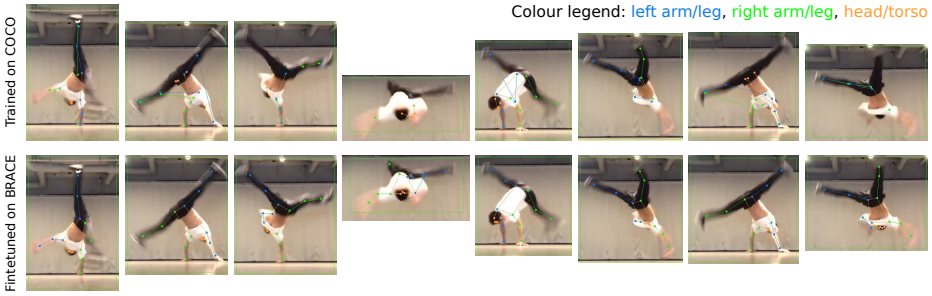

Fig. 1. Qualitative results for pose estimation on the external powermoves video.

DTW cost, which suggests that the ability of the network to align dance and music is not affected by the number of encoding layers and the attention window size.

4 Pose Estimation

We show here qualitative results for the pose estimation experiment on the external powermoves video [1]. Figure 1 compares poses estimated with the base model (HRNet [7]) trained on COCO [5] (top row) and the model finetuned on BRACE (bottom row). We display here particularly difficult cases where the base model struggled due to the complex pose and the prominent motion blur. Notice how the model finetuned on BRACE is able to predict very accurate keypoints thanks to the additional training on our manually labelled frames. Our data processing pipeline picks the hardest frames for manual labelling. Our high quality labels for such difficult frames prove effective for the model to learn to predict accurate keypoints even for very tangled postures in the presence of motion blur. While the main scope of our work is dance motion synthesis, we believe BRACE can also be a valuable resource for pose estimation.

5 Bézier interpolation

A Bézier curve is a parametric curve based on Bernstein polynomials. Given a set of control points $P = (P_0, P_1, \ldots, P_d)$, a Bézier curve can be obtained multiplying the Bernstein matrix by the control points. For a given order d and a vector τ of m values uniformly spaced between 0 and 1, the Bernstein matrix $M \in \mathrm{R}^{m \times (d+1)}$ is defined as follows:

$$M_{i,j+1} = \binom{d}{j} (1 - \tau_i)^{d-j} \tau_i^j \tag{1}$$

where $0 \leq i < m$, $0 \leq j \leq d$ and $\tau \in [0, 1]$. In our case, P is unknown and is what we need to find to fit a Bézier curve to a node trajectory. Let $R \in \mathrm{R}^{q \times 2}$ be the

[1] `https://www.youtube.com/watch?v=q5Xr0F4a0iU`

matrix formed stacking the 2D coordinates of a node trajectory, i.e. $R_i = (x_i, y_i)$. R can contain a mix of automatic and manually annotated 2D points. Note that keypoints are not expected to be temporally continuous due to missing or discarded poses. In other words, if $T(i)$ is a function returning the corresponding frame number of the i-th point in the trajectory, $T(i+1) = T(i) + 1$ does not necessarily hold true. Following [6] we compute P using the least square method. Namely, we first calculate M with a τ vector of q values and predefined order d. We then calculate the Moore–Penrose inverse M^+ of M. Finally, the fitted control points are given by $P = M^+R$. Once P is obtained we can generate the interpolated trajectory by multiplying the previously calculated M by P. We adopt a sliding window approach to interpolate keypoint sequences. Specifically, we slide a window of 15 frames with stride 14. The overlapping frame between windows allows for a smoother transition. We set the degree of the curve to 7.

References

1. Huang, R., Hu, H., Wu, W., Sawada, K., Zhang, M., Jiang, D.: Dance revolution: Long-term dance generation with music via curriculum learning. International Conference on Learning Representations (2021)
2. Lee, H.Y., Yang, X., Liu, M.Y., Wang, T.C., Lu, Y.D., Yang, M.H., Kautz, J.: Dancing to music. Neural Information Processing Systems (2019)
3. Li, R., Yang, S., Ross, D.A., Kanazawa, A.: AI Choreographer: Music Conditioned 3D Dance Generation with AIST++. International Conference on Computer Vision (2021)
4. Librosa: Librosa. https://librosa.org/doc/main/index.html
5. Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft COCO: Common objects in context. European Conference on Computer Vision (2014)
6. Pastva, T.A.: Bezier curve fitting. Tech. rep., Naval Postgraduate School, Monterey CA (1998)
7. Sun, K., Xiao, B., Liu, D., Wang, J.: Deep high-resolution representation learning for human pose estimation. Conference on Computer Vision and Pattern Recognition (2019)