Supplementary material: WalkTheDog: Cross-Morphology Motion Alignment via Phase Manifolds

Peizhuo Li
ETH Zurich
Switzerland
peizhuo.li@inf.ethz.ch

Sebastian Starke
Meta Reality Labs
United Kindom
sstarke@meta.com

Yuting Ye
Meta Reality Labs
USA
yuting.ye@meta.com

Olga Sorkine-Hornung
ETH Zurich
Switzerland
sorkine@inf.ethz.ch

1 IMPLEMENTATION DETAILS

In this section, we give a detailed description of the phase calculation module and the motion matching (without frequency scaling) algorithm.

1.1 Phase Calculation

We use the equations presented by Mason [2022] to calculate the phase ϕ using the entries y_i from 1-channel signal $\mathbf{Y}$ and the relative timing t_i from $\mathcal{T}$:

$$\begin{aligned} s_x &= \sum_{i=1}^{T} y_i \cos(2\pi f t_i), \\ s_y &= \sum_{i=1}^{T} y_i \sin(2\pi f t_i), \\ \phi &= \frac{\operatorname{atan2}(s_y, s_x)}{2\pi}, \end{aligned} \tag{1}$$

where $\operatorname{atan2}(y, x)$ is the argument of the complex number $x + iy$. This equation helps us avoid dealing with the fact that ϕ is not a continuous parameterization of the phase manifold.

1.2 Motion Matching on Phase Manifolds

Given the phase embedding sequence $\mathbf{P}$ of an input motion sequence with T frames, we use Algorithm 1 to retrieve the motion sequence from the database, using phase embedding as the control signal in the classical motion matching algorithm. We denote the length of replay after each match with T_0, the phase sequences of the database with $\mathbf{Q}$, the pose descriptor used to measure the similarity between frames with $\mathbf{J}$ and the pose with $\mathbf{Y}$.

In our experiment, we use a simple setup where normalized joint positions are chosen as our pose descriptor $\mathbf{J}$ to ensure a smooth transition between different replays. We also apply inertialization at each transition. The cost function in Algorithm 1 can be customized depending on the exact application.

Algorithm 1 Phase Manifold Motion Matching

$i \leftarrow 1$
$\mathbf{J}_{\text{start}} \leftarrow$ initial pose descriptor
while $i < T$ **do**
  $k = \arg\min_k \|\mathbf{P}_{i:i+T_0} - \mathbf{Q}_{k:k+T_0}\|_2^2 + \lambda \|\mathbf{J}_{\text{start}} - \mathbf{J}_k\|_2^2$
  Output $\mathbf{Y}_{k:k+T_0}$
  $i \leftarrow i + T_0$
  $\mathbf{J}_{\text{start}} \leftarrow \mathbf{J}_{k+T_0}$
end while

Table 1: Details of the datasets used in our experiments.

Name	Framerate	# of Frames
Dog [2018]	60	151k
Human-Locomotion [2019]	60	186k
MOCHA-Clown [2023]	120	486k
MOCHA-Ogre [2023]	120	500k
MOCHA-Princess [2023]	120	501k

Since skeleton-aware networks [Aberman et al. 2020] require homeomorphic skeletons, we remove the tail of the dog skeleton when compared with it. In addition, we specify a correspondence between the forelegs and arms, and the hindlegs and legs. SAN heavily requires end-effector velocity consistency between the source and target characters and struggles to transfer motion with a large difference in skeletal structure.

1.3 Datasets

A detailed description of datasets is listed in Table 1. The training time on Dog and Human-Locomotion is around 40 minutes while on MOCHA-Clown and MOCHA-Ogre is around 2 hours, proportional to the number of frames. Wehen training on all the datasets together, it takes around 2 hours and 40 minutes.

SIGGRAPH Conference Papers '24, July 27-August 1, 2024, Denver, CO, USA

ACM ISBN 979-8-4007-0525-0/24/07.
https://doi.org/10.1145/3641519.3657508

2 NETWORK ARCHITECTURE AND HYPERPARAMETERS

In this section, we describe the network architecture and the hyperparameters used to train the network.

2.1 Architecture Details

The convolutional encoder and decoder share the same architecture of two-layer 1D convolutions with kernel size 23 with ELU as activation. The MLP producing raw amplitude $\tilde{\mathbf{A}}$ has 5 layers and the hidden units are of the same size as the amplitude. The MLP in the phase calculation module also has 5 layers. The hidden units share the size as the input frequency bin powers produced by FFT. The MLP used in learning average poses has 8 layers, and the hidden units are of the same size as the pose. LeakyReLU with a negative slope 0.2 is used as activation for all aforementioned MLPs.

2.2 Hyperparameters

Our VQ-PAE is implemented in PyTorch [Paszke et al. 2019], and the experiments are performed on NVIDIA GeForce RTX 3090 GPU. We optimize the parameters of our network using the Adam optimizer [Kingma and Ba 2014]. We set the learning rate to 1×10^{-4} and the batch size to 32. We choose to use joint velocity in the local coordinate of the character as the input feature $\mathbf{X}$. We choose the size of the input motion sequence during training to correspond to 1 second. The exact size is different for different datasets depending on their framerate. For the hyperparameters used in frequency-scaled motion matching, we set $\lambda_1 = 0.5$ and $\lambda_2 = 1$.

REFERENCES

Kfir Aberman, Peizhuo Li, Dani Lischinski, Olga Sorkine-Hornung, Daniel Cohen-Or, and Baoquan Chen. 2020. Skeleton-aware networks for deep motion retargeting. *ACM Transactions on Graphics (TOG)* 39, 4 (2020), 62–1.

Deok-Kyeong Jang, Yuting Ye, Jungdam Won, and Sung-Hee Lee. 2023. MOCHA: Real-Time Motion Characterization via Context Matching. In *SIGGRAPH Asia 2023 Conference Papers*. 1–11.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

Ian Mason. 2022. Periodic Autoencoder - Explanation and Addendum. https://www.ianxmason.com/posts/PAE/

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (Eds.). Curran Associates, Inc., 8024–8035. http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

Sebastian Starke, He Zhang, Taku Komura, and Jun Saito. 2019. Neural state machine for character-scene interactions. *ACM Trans. Graph.* 38, 6 (2019), 209–1.

He Zhang, Sebastian Starke, Taku Komura, and Jun Saito. 2018. Mode-adaptive neural networks for quadruped motion control. *ACM Transactions on Graphics (TOG)* 37, 4 (2018), 1–11.