# Neural Motion Compression with Frequency-adaptive Fourier Feature Network

Kenji Tojo, Yifei Chen, and Nobuyuki Umetani

The University of Tokyo

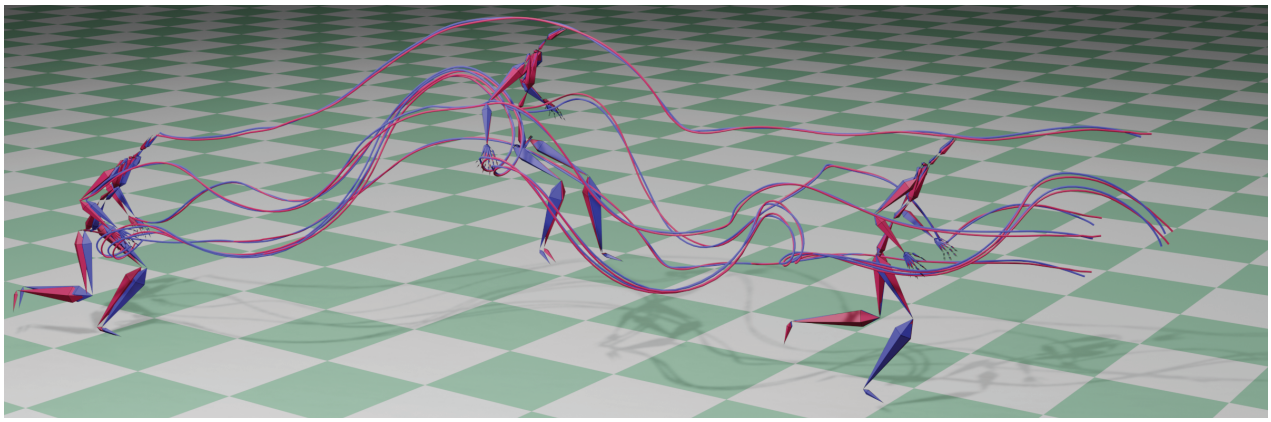

**Figure 1:** *Running and jumping motion of a character before (red) and after (blue) our neural motion compression. The size of the data is* 8.17 *times smaller while the differences of the coordinates of joint positions are below* 3.7% *of the character's height. Five frequencies are adaptively sampled for the Fourier feature to achieve high compression ratio.*

**Abstract**

*We present a neural-network-based compression method to alleviate the storage cost of motion capture data. Human motions such as locomotion, often consist of periodic movements. We leverage this periodicity by applying Fourier features to a multilayered perceptron network. Our novel algorithm finds a set of Fourier feature frequencies based on the discrete cosine transformation (DCT) of motion. During training, we incrementally added a dominant frequency of the DCT to a current set of Fourier feature frequencies until a given quality threshold was satisfied. We conducted an experiment using CMU motion dataset, and the results suggest that our method achieves overall high compression ratio while maintaining its quality.*

**CCS Concepts**

• ***Computing methodologies*** → ***Animation; Neural networks;***

## 1. Introduction

High-quality character animation is a vital element of immersive experiences in various interactive applications, such as video games and virtual reality systems. To make the animation of characters controllable, such applications often leverage a huge database consisting of motion data acquired from motion capture or created by artists [KGP02]. However, storing the motion data in the disk and streaming the data into the memory is often costly.

Motion compression is one approach to addressing this challenge. Researchers have investigated efficient methods for reducing the size of motion data using a variety of techniques such as a principal component analysis (PCA) [Ari06], wavelets [BPvdP07], and pattern indexing [GPD09]. However, these previous studies did not explore ways to exploit emerging neural representations.

Neural-network-based representations of images and shapes have recently gained significant interest. Among them, coordinate-based networks, which map spatial coordinates to the value at the point (e.g., color and occupancy) provide memory-efficient representations [MLL*21]. For such compact coordinate-based



DOI: 10.2312/egs.20221033

networks to learn high-frequency details, input encoding techniques such as Fourier features [TSM*20] are commonly adopted [MST*20].

Drawing inspiration from these studies, we present a method for compressing motion sequences using a neural network with Fourier features. The scalar time input is encoded using trigonometric functions and then mapped to the configuration of the articulated joints using a multilayer perceptron (MLP). Instead of stochastic sampling, as in the original study [TSM*20], we propose determining a set of frequencies for the Fourier feature based on the discrete cosine transformation (DCT) of the motion sequence. We demonstrate the effectiveness of our method by comparing it with a method based on singular value decomposition and a model with a naïve choice of frequencies in the Fourier feature.

## 2. Related Work

Researchers have tackled the problem of compressing motion sequences by leveraging both spatial and temporal redundancy in the data [LTYF19]. Arikan [Ari06] proposed the parameterization of a motion as cubic Bézier segments and then reducing the number of parameters using a PCA. Beaudoin et al. [BPvdP07] presented a wavelet-based compression method tailored for motion sequences to exploit their temporal coherence. Gu et al. [GPD09] introduced a method for detecting redundant patterns in a motion sequence and removing them using a hierarchical data structure. However, none of the previous studies considered the use of neural representations for motion compression.

With the rise of neural data processing, data compression using neural representations has gained significant attention. Emerging neural implicit representations for images and 3D shapes can reconstruct highly complicated signals with a compact network, drastically reducing the data size [MLL*21]. To facilitate the learning of high-frequency signals, several implicit networks periodically encode the input data, that is, encoding with Fourier features [TSM*20]. We have taken inspiration from these researches and designed a neural representation of motion sequences to reduce their data size.

Data-driven character animation controllers are another growing field of research applying neural networks to synthesize animation data. Fragkiadaki et al. [FLFM15] presented a model that extends recurrent neural networks (RNNs) with encoder and decoder networks to synthesize character locomotion sequences. In addition, to model complicated interactions between the character and its surroundings, Holden et al. [HKS17] changed the network weights based on the phase of motion. Although these studies attempt to generate an unseen animation from training data, we focus on obtaining a compact representation of existing motion without a quality degradation.

## 3. Method

Given a character motion sequence, we compute its compact representation using a neural network. Owing to inertia, character motion is generally temporally coherent, that is, the pose changes continuously over the frames. Hence, the neural network approximates a *continuous* function that takes the normalized time $t \in [0,1]$ as input to output the character pose. This multivariate nonlinear function is modeled by combining the Fourier feature [TSM*20] and a multilayer perceptron (MLP).

**Pose Parameterization** Similar to the typical rigged character representation, we assume that the pose of the character is described with a skeleton comprising a tree of bones, where the transformation of the bone is hierarchically applied from the root bone to the child bones. The pose of the character in one frame is represented by $m$ parameters $\mathbf{y} \in \mathbb{R}^m$, which contain the translation and rotation of the root bone and the relative rotation of the child bones with respect to their parent bone.

We carefully chose the numerical representation of the rotation to make it compact while avoiding a discontinuity. We represent rotation of the root bone with a quaternion to continuously parameterize large rotations exceeding $\pi$. Note that the negative of a quaternion represent the same rotation as the original quaternion. To avoid the discontinuity caused by switching between two representations in the adjacent frames, we choose from quaternion and its negative the one that is similar to the previous frame in the sense of the L2 norm. The 6D rotation representation [ZBJ*19] can be potentially used here, but our quarternion representation suffices for the character's root bone rotation. The relative rotations of the child bones are typically small because of joint limits. Hence, in this study, we parameterize them using an axis-angle representation.

### 3.1. Fourier Feature-based Motion Representation

**Fourier Feature** The input of scalar time $t$ is first mapped to the high-dimensional Fourier feature using trigonometric functions to facilitate high-frequency pose regression using the MLP. Given a set of $K$ angular velocities $\Omega = \{\omega_1, \omega_2, \dots, \omega_K\}$, we compute the Fourier feature $\mathbf{x} \in \mathbb{R}^{2K+1}$ at time $t$ as

$$\mathbf{x}(t;\Omega) = \{t, \sin\omega_1 t, \cos\omega_1 t, \dots, \sin\omega_K t, \cos\omega_K t\}^T. \quad (1)$$

The Fourier feature $\mathbf{x}(t)$ is then passed to the network $\mathcal{F}$ that outputs the pose of the character. The angular velocities in $\Omega$ are adaptively determined during training using the algorithm described in Section 3.2.

Although the original Fourier feature [TSM*20] consists only of sinusoidal components, we add the linear component $t$ to the Fourier feature. This corresponds to angular velocities small enough to make sinusoidal functions approximately linear for $t \in [0,1]$ and helps the network learn the part of the function that change linearly with the input time. Such a linear relationship is commonly seen in the translation component of the pose representation when the character walk/run at a constant speed.

**Network Architecture** We designed our MLP based on the network architecture tailored to represent the pose presented in [HKS17]. Specifically, we employ a simple three-layer neural network to learn the relationship between the input parameters $\mathbf{x} \in \mathbb{R}^{2K+1}$ and output pose parameters $\mathbf{y} \in \mathbb{R}^m$ as

$$\mathcal{F}(\mathbf{x};\alpha) = \mathbf{W}_2 \mathrm{ELU}\left(\mathbf{W}_1 \mathrm{ELU}\left(\mathbf{W}_0 \mathbf{x} + \mathbf{b}_0\right) + \mathbf{b}_1\right) + \mathbf{b}_2, \quad (2)$$

**Algorithm 1** Frequency-adaptive Fourier feature compression.

**Input:** sequence of the pose parameters $\{\mathbf{y}_1, \ldots, \mathbf{y}_N\}$
**Output:** network weights $\alpha$, angular velocities $\Omega$

Initialize $\Omega = \emptyset$
**while** True **do**
  Construct $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$ from $\Omega$ (1).
  Update $\alpha$ by training network (3).
  **if** Joint position error is below threshold **then**
    **break**
  Compute residual $\mathbf{R} = \mathbf{Y} - \mathcal{F}(\mathbf{X})$.
  Compute first component $\mathbf{p}$ via PCA on residual.
  Compute DCT on the first component in the residual.
  Append new frequency to $\Omega$.
  Construct network with bigger size and copy old weights.
**return** $\alpha$, $\Omega$

where $\alpha$ is the set of trainable parameters $\alpha = \{\mathbf{W}_0 \in \mathbb{R}^{h\times 2K+1}, \mathbf{W}_1 \in \mathbb{R}^{h\times h}, \mathbf{W}_2 \in \mathbb{R}^{h\times m}, \mathbf{b}_0 \in \mathbb{R}^h, \mathbf{b}_1 \in \mathbb{R}^h, \mathbf{b}_2 \in \mathbb{R}^m\}$. In this study, we chose the size of the hidden layer to be double the size of the input $h = 2(2K+1)$. Here, ELU represents the exponential rectified linear function [CUH16].

**Error Metric** The goal of the compression is to represent motion sequence with small number of parameters while keeping the error small. We measured the error of the motion compression by comparing the positions of the joints in the skeleton before and after compression. Specifically, we report the $L_\infty$ norm (i.e., maximum difference) of the position difference over all joints and all frames. The ratio of the compression is computed by the number of parameters in the network $\alpha$ divided by the number of parameters in the original representation.

**Network Training** It is extremely difficult to directly optimize the network weights $\alpha$ for the error defined by the $L_\infty$ norm. Instead, we optimize the network weights with a different loss function defined through the $L_2$ norm, using the originally defined error metric, as with the stopping condition of the entire pipeline.

We denote the sequence of $N$ animation frames with continuously parameterized poses as $\mathbf{Y} = \{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_N\} \in \mathbb{R}^{N\times m}$ and the corresponding Fourier features as $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\} \in \mathbb{R}^{N\times 2K+1}$. The loss function used to train the network can be written as

$$\mathcal{L}(\alpha) = \sum_{i=1}^{N} \| \{\mathbf{y}_i - \mathcal{F}(\mathbf{x}_i; \alpha)\} \circ \mathbf{w} \|^2, \tag{3}$$

where the "$\circ$" symbol represents element-wise multiplication (i.e., Hadamard product). The vector $\mathbf{w} \in \mathbb{R}^m$ is the weight of the pose parameter that encodes the extent to which the change in the parameter roughly results in a change in the joint positions. In this study, we set the weights 1 for the translational component and the distance between the farthest joint position in the child bones in the initial frame for the joint rotations. Multiplying this weight $\mathbf{w}$ unifies the physical dimensions of the pose parameters, that is, the mixture of the lengths (root bone translation) and angles (joint rotations).

### 3.2. Frequency-adaptive Fourier Feature

Character motion, such as human locomotion, cycles itself and therefore often has a few dominant frequencies. While the original Fourier feature in [TSM*20] chooses a set of frequencies $\Omega$ through stochastic sampling, our training pipeline adaptively identifies such dominant frequencies from the input motion sequence and uses them for the Fourier feature.

We start training the network using the feature $\mathbf{x}(t) = t$ with empty $\Omega$. After the network training is completed, the dominant frequency in the residual of the function fitting is added to $\Omega$ and the larger network is trained again. This process is iterated until the error of the joint position difference is under a given threshold. When the size of the network is changed due to addition of the new frequency, we copy all the weights from the previous smaller sized network to *warm start* the training instead of training from scratch. As a result, the network weights decrease steadily during entire fitting process even if the network size changes.

The dominant frequency is computed by combining a PCA with a discrete cosine transformation (DCT). First, we compute a residual matrix, which is the difference between the target data and the evaluated function $\mathbf{R} = \{\mathbf{Y} - \mathcal{F}(\mathbf{X})\} \in \mathbb{R}^{N\times m}$. Each row of the residual matrix is then weighted using $\mathbf{w}$. By applying the PCA to the weighted residual matrix, we obtain the first component $\mathbf{p} \in \mathbb{R}^m$. We then apply the DCT to the time history of the first component in the residual and find the dominant frequency as the peak of the output of the DCT.

In summary, our algorithm used for constructing neural network while adaptively choosing the Fourier feature frequencies can be written in a pseudo code as Algorithm 1.

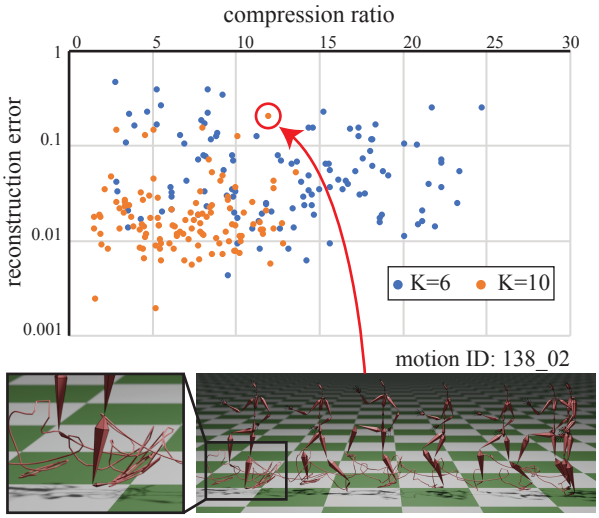


**Figure 2:** *Neural motion compression of* 116 *sequences randomly selected from the CMU motion capture database [cmu]. Data corruption is seen in the motion of the high compression error. For example, the character's toes vibrate and flip for the motion with highest error (maximum joint coordinate difference is* 20.3% *of the character's height which is about* 31.1 *cm).*

## 4. Results

We applied our method to CMU motion capture database [cmu] and compared to several baseline methods. To evaluate the compression methods, we used two metrics: the reconstruction error (i.e., maximum difference in the joint position) and the compression ratio (i.e., size of $\mathbf{Y}$ divided by the number of neural network parameters $\alpha$). The training takes about one minute and a half on average to train up to $K = 10$ for motion sequence with 900 frames.

In Figure 2, we present the results of our method for various motion sequences randomly chosen from CMU motion capture database [cmu]. We can see that our approach can reduce the storage size 2 to 15 times while maintaining the reconstruction error below 2%, which is about 3 cm, in most cases.

### 4.1. Comparison with baseline methods

We compared our method with a baseline approach based on a singular value decomposition (SVD). We decompose the matrix $\mathbf{Y}$ and compute its low-rank approximation using SVD. Another baseline approach is naïve deterministic choice of Fourier feature frequencies where their angular velocities are consecutive integers (i.e., $\Omega = \{1, 2, \ldots, K\}$). Note that angular velocity takes value two, for example, when its feature cycles twice throughout the sequence.

Figure 3 compares compression results of the three methods (compression using SVD, naïve choice of Fourier frequencies, and ours). The results suggest that our method achieves better performance (e.g., lower reconstruction error with higher compression ratio) than other two methods. This suggests that an appropriate choice of Fourier feature frequencies is of significant importance for a neural network to faithfully represent the original motion.

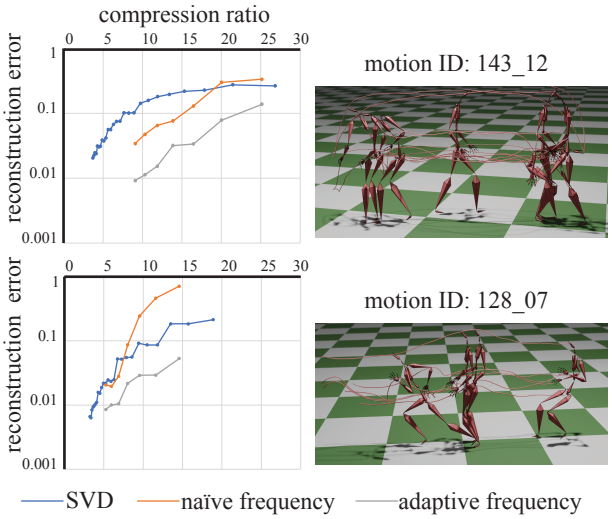


**Figure 3:** *Comparison of compression performance between singular value decomposition (SVD), naïve choice of Fourier feature frequencies (naïve frequency) where the angular velocities are integers (i.e., $\Omega = \{1, 2, \ldots, K\}$) and our adaptively chosen Fourier frequencies (adaptive frequency). We compress with various numbers of nonzero singular values and with various number of Fourier feature frequency K. Our compression outperforms the other methods in terms of trade-off between error and compression ratio.*

## 5. Conclusion

In this study, we used a neural network with the Fourier feature to obtain a compact representation of a given character animation sequence. We selected Fourier feature frequencies based on the DCT of the input sequence to capture periodicity in the data. Our method can compress motion sequences with a low reconstruction error than other baseline methods based on the SVD or naïve frequency choice of Fourier feature.

One fundamental limitation of our method is that it is difficult to represent discontinuously changing data, which is often seen in the corrupt capture. Currently, the training is performed independently for each motion sequence. In a future study, we are planning to achieve further compression by encoding common style of motion (e.g., walking or running) seen across many motion sequences.

**Acknowledgements** This work was supported by JSPS KAKENHI Grant Number 21K11910.

## References

[Ari06] ARIKAN O.: Compression of motion capture databases. *ACM Trans. Graph. 25*, 3 (July 2006), 890–897. 1, 2

[BPvdP07] BEAUDOIN P., POULIN P., VAN DE PANNE M.: Adapting wavelet compression to human motion capture clips. In *Proceedings of Graphics Interface 2007* (2007). 1, 2

[cmu] Carnegie Mellon University - CMU Graphics Lab - Motion Capture Library. URL: http://mocap.cs.cmu.edu/. 3, 4

[CUH16] CLEVERT D.-A., UNTERTHINER T., HOCHREITER S.: Fast and accurate deep network learning by exponential linear units (ELUs), 2016. arXiv:1511.07289. 3

[FLFM15] FRAGKIADAKI K., LEVINE S., FELSEN P., MALIK J.: Recurrent network models for human dynamics. In *ICCV* (2015). 2

[GPD09] GU Q., PENG J., DENG Z.: Compression of human motion capture data using motion pattern indexing. *Computer Graphics Forum 28*, 1 (2009), 1–12. 1, 2

[HKS17] HOLDEN D., KOMURA T., SAITO J.: Phase-functioned neural networks for character control. *ACM Trans. Graph. 36*, 4 (2017). 2

[KGP02] KOVAR L., GLEICHER M., PIGHIN F.: Motion graphs. *ACM Trans. Graph. 21*, 3 (2002), 473–482. 1

[LTYF19] LV N., TIAN J., YANG X., FAN X.: A survey on motion capture data compression algorithm. In *Proceedings of the 2nd International Conference on Big Data Technologies* (2019). 2

[MLL*21] MARTEL J. N. P., LINDELL D. B., LIN C. Z., CHAN E. R., MONTEIRO M., WETZSTEIN G.: Acorn: Adaptive coordinate networks for neural scene representation. *ACM Trans. Graph. 40*, 4 (2021). 1, 2

[MST*20] MILDENHALL B., SRINIVASAN P. P., TANCIK M., BARRON J. T., RAMAMOORTHI R., NG R.: NeRF: Representing scenes as neural radiance fields for view synthesis. In *ECCV* (2020). 2

[TSM*20] TANCIK M., SRINIVASAN P. P., MILDENHALL B., FRIDOVICH-KEIL S., RAGHAVAN N., SINGHAL U., RAMAMOORTHI R., BARRON J. T., NG R.: Fourier features let networks learn high frequency functions in low dimensional domains. *NeurIPS* (2020). 2, 3

[ZBJ*19] ZHOU Y., BARNES C., JINGWAN L., JIMEI Y., HAO L.: On the continuity of rotation representations in neural networks. In *CVPR* (2019). 2