Human Motion Generative Model Using Variational Autoencoder

Yuichiro Motegi, Yuma Hijioka, and Makoto Murakami

***Abstract*—We present a technique to learn large human motion data captured with optical motion capture system, represent it in a low dimensional latent space, so as to generate natural and various human motions from it. To extract human motion features we use a convolutional autoencoder, and to represent the extracted features as a probability density function in a latent space we use a variational autoencoder. Motion generator is modeled as a map from a latent variable sampled in the latent space to a motion capture data. We stack the convolutional decoder on top of the variational decoder, which can sample a latent variable and produce a motion. As a result, our system can generate natural and various human motions from a 32-dimensional latent space.**

***Index Terms*—Character animation, convolutional autoencoder, motion generative model, variational autoencoder.**

I. Introduction

Human motion control, edit, and synthesis are important tasks to create 3D computer graphics video games or movies, because some characters act like humans in most of them. Key frame interpolation method is useful for producing human motion. But the user has to set a lot of parameters of human joints in some key frames manually, and the produced motion is less realistic than the data captured with motion capture system. Therefore motion capture data-driven method is used for motion control, edit, and synthesis, and many techniques have been proposed [1].

Deep neural networks have been used in human motion control. Holden *et al*. [2] used convolutional autoencoders to learn human motion manifold from a large motion dataset captured with an optical motion capture system. And they [3] stack deep neural networks on top of the autoencoders, which can map from high level parameters to the motion manifold. The proposed system can synthesize character motion from given trajectories over the floor that the character should follow, and can edit motion by optimizing in the motion manifold with some constraints.

On the other hand, some generative models using deep neural networks have been proposed. Kingma *et al*. [4], [5] proposed variational autoencoder, and applied it for image generation. The proposed system can generate natural and various images. Radford *et al*. [6] proposed deep generative adversarial networks, which can also generate natural and various images. Bowman *et al*. [7] proposed RNN-based variational autoencoder for natural language sentences, which can generate diverse sentences that interpolate between learned sentences. Fabius *et al*. [8] proposed variational recurrent autoencoder, which can be applied to time series data. The proposed method can generate MIDI format music data. Sabaththe *et al*. [9] used LSTM-based variational autoencoder for automatic music composition and it can generate various music pieces that represent some musical characteristics and properties.

In this paper we propose a human motion generative model using convolutional autoencoder and variational autoencoder, which can represent motion capture data in a low dimensional latent space and can generate natural and various human motions from it.

Manuscript received November 15, 2017; revised January 12, 2018.

Yuichiro Motegi and Makoto Murakami are with Graduate School of Information Sciences and Arts, Toyo University, Saitama, Japan (e-mail: s3b101710012@toyo.jp, murakami_m@toyo.jp).

Yuma Hijioka is with Graduate School of Science and Engineering, Toyo University, Saitama, Japan (e-mail: yuma.hijioka@ieee.org).

II. Motion Data

In this section we describe human motion dataset and preprocessing of it for learning motion generative model.

We use the CMU Graphics Lab Motion Capture Database [10], which consists of 2,505 recordings of human motion captured with an optical motion capture system. We downsample this original data with 120 fps to 30 fps.

The original motion data is represented as 3-DOF local rotations of 19 joints and 3-DOF global translation of root joint (hip) as shown in Fig. 1. We convert them into the global positions $\boldsymbol{p}_j^{(g)}(t)$, where j is a joint and t is time.

Given the global position of hip $\boldsymbol{p}_h^{(g)}(t)$, left shoulder $\boldsymbol{p}_{ls}^{(g)}(t)$, and right shoulder $\boldsymbol{p}_{rs}^{(g)}(t)$ in time t, the forward vector of body $\boldsymbol{v}_f(t)$ is calculated as

$$\boldsymbol{v}_l(t) = \boldsymbol{p}_{ls}^{(g)}(t) - \boldsymbol{p}_h^{(g)}(t)$$
$$\boldsymbol{v}_r(t) = \boldsymbol{p}_{rs}^{(g)}(t) - \boldsymbol{p}_h^{(g)}(t)$$
$$\boldsymbol{v}_f(t) = \boldsymbol{v}_l(t) \times \boldsymbol{v}_r(t).$$

The basis vectors of local coordinate system at time t are calculated as

$$\boldsymbol{e}_y = [0 \quad 1 \quad 0]^T$$
$$\boldsymbol{e}_x(t) = \frac{\boldsymbol{v}_f(t) \times \boldsymbol{e}_y}{\left\| \boldsymbol{v}_f(t) \times \boldsymbol{e}_y \right\|}$$
$$\boldsymbol{e}_z(t) = \boldsymbol{e}_y \times \boldsymbol{e}_x(t).$$

The rotation matrix at time t is represented as

DOI: 10.7763/IJMO.2018.V8.616

$$R_{gl}(t) = \begin{bmatrix} \boldsymbol{e}_x(t) & \boldsymbol{e}_y & \boldsymbol{e}_z(t) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Let $p^{(g)}_{\min,y}(0)$ be the minimum of y-coordinate over all joints in the initial frame, which indicates the floor height if one of joints is on the ground in the initial frame. We set the origin in local coordinate system at time t as a point on the ground where the root joint position is projected onto. The translation matrix at time t is represented as

$$T_{gl}(t) = \begin{bmatrix} 1 & 0 & 0 & p^{(g)}_{h,x}(t) \\ 0 & 1 & 0 & p^{(g)}_{\min,y}(0) \\ 0 & 0 & 1 & p^{(g)}_{h,z}(t) \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

where $p^{(g)}_{h,x}(t)$ and $p^{(g)}_{h,z}(t)$ are x-coordinate and z-coordinate of the root joint at time t respectively. The transformation matrices between local coordinate system and global coordinate system are represented as

$$M_{gl}(t) = T_{gl}(t)R_{gl}(t)$$
$$M_{lg}(t) = R^T_{gl}(t)T^{-1}_{gl}(t),$$

and the local positions of joints at time t is calculated as

$$\boldsymbol{p}^{(l)}_j(t) = M_{lg}(t)\boldsymbol{p}^{(g)}_j(t).$$

We use y-coordinate of the root joint, and xyz-coordinates of the other 18 joints. We also use velocity in the xz plane and angular velocity around the y axis, which are calculated from the matrix $\Delta M(t)$, which is

$$\Delta M(t) = M_{lg}(t-1)M_{gl}(t).$$

The motion data at each frame is a 58-dimensional vector.

We separate each sequence of frames into windows of 120 frames (about 4 seconds), overlapped by 60 frames. Finally we get 14,122 motions, and we subtract the mean from them and divide them by the standard deviation to standardize the data.

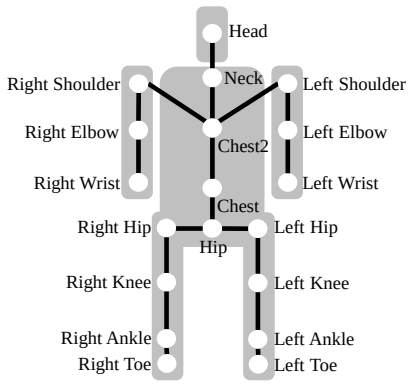

Fig. 1. Joint structure in motion data.

III. Convolutional Autoencoder for Motion Feature Extraction

Holden *et al.* [2], [3] use convolutional autoencoder to construct a human motion manifold which includes of various types of human motion. We use convolutional autoencoder to extract various motion features. In this section we describe the structure of the convolutional autoencoder.

A. Network Structure

We use one-layer convolutional autoencoder. An overview of the network structure is shown in Fig. 2.

The 120×58 dimensional motion vector is convolved with 32 spatiotemporal filters, the size of which is 15×58 and the stride is $(1, 58)$, and it is put through the ReLU function. And temporal max pooling is used to get 60×32 motion feature vector. Given a motion vector $\boldsymbol{x}$, the convolution operator $*$, max pooling operator Ψ, filter weights W and biases $\boldsymbol{b}$, the convolutional encoder is represented as

$$\Phi(\boldsymbol{x}) = \Psi(\mathrm{ReLU}(\boldsymbol{x} * W + \boldsymbol{b})).$$

And the 60×32 motion feature vector is upsampled in temporal axis, and it is convolved with 58 filters, the size of which is 15×32 and the stride is (1, 32), to get 120×58 motion vector. Given an encoded feature vector $\boldsymbol{y}$, upsampling operator Ψ', filter weights W' and biases $\boldsymbol{b}'$, the convolutional decoder is represented as

$$\Phi'(\boldsymbol{y}) = \Psi'(\boldsymbol{y}) * W' + \boldsymbol{b}'.$$

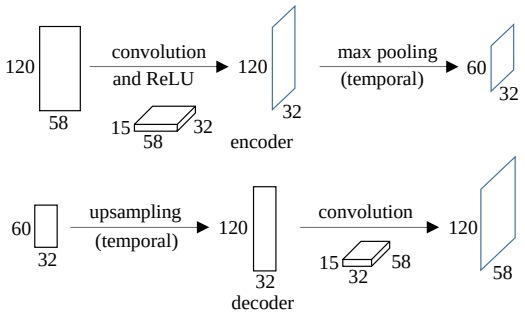

Fig. 2. Structure of convolutional autoencoder.

B. Training

Given a training set $X = \boldsymbol{x}^{(n)}(n = 1, \cdots, N)$, to minimize the cost function $C(X; W, \boldsymbol{b}, W', \boldsymbol{b}')$, which is mean square error between input vectors and the outputs of the convolutional autoencoder with parameters $W, \boldsymbol{b}, W', \boldsymbol{b}'$,

$$C(X; W, \boldsymbol{b}, W', \boldsymbol{b}') = \sum_{n=1}^{N} \left\| \boldsymbol{x}^{(n)} - \Phi'(\Phi(\boldsymbol{x}^{(n)})) \right\|^2,$$

We estimate the network parameters $W, \boldsymbol{b}, W', \boldsymbol{b}'$. The number of the training data is 12,710, and the number of the test data is 1,412. We use gradient descent with Adam optimization algorithm, the batch size is 100, and the optimization is performed for 200 epochs.

IV. Variational Autoencoder for Motion Representation

Kingma *et al.* [4], [5] proposed variational autoencoder, which can represent high dimensional data in a low

dimensional latent space. We use the variational autoencoder to represent the motion features as a probability density function in a latent space.

A. Network Structure

Let $\boldsymbol{y}$ and $\boldsymbol{z}$ be a motion feature vector and a latent variables vector, respectively. As shown in Fig. 3 and 4, we assume that probabilistic encoder $q_\phi(\boldsymbol{z}|\boldsymbol{y})$ and probabilistic decoder $p_\theta(\boldsymbol{y}|\boldsymbol{z})$ are normal distributions. The parameters $\boldsymbol{\mu}_\phi, \boldsymbol{\sigma}_\phi^2$ of the encoder $q_\phi(\boldsymbol{z}|\boldsymbol{y})$ are modeled by neural networks with the weights W_ϕ. And the parameters $\boldsymbol{\mu}_\theta, \boldsymbol{\sigma}_\theta^2$ of the decoder $p_\theta(\boldsymbol{y}|\boldsymbol{z})$ are modeled by neural networks with the weights W_θ. The encoder and decoder are represented as

$$q_\phi(\boldsymbol{z}|\boldsymbol{y}) = \mathcal{N}\left(\boldsymbol{z}|\boldsymbol{\mu}_\phi(\boldsymbol{y}; W_\phi), \mathrm{diag}\left(\boldsymbol{\sigma}_\phi^2(\boldsymbol{y}; W_\phi)\right)\right),$$

$$p_\theta(\boldsymbol{y}|\boldsymbol{z}) = \mathcal{N}\left(\boldsymbol{y}|\boldsymbol{\mu}_\theta(\boldsymbol{z}; W_\theta), \mathrm{diag}\left(\boldsymbol{\sigma}_\theta^2(\boldsymbol{z}; W_\theta)\right)\right).$$

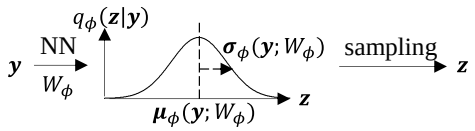

Fig. 3. Variational encoder (sampling).

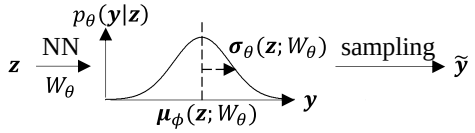

Fig. 4. Variational decoder.

As shown in Fig. 5, the neural networks as a model of $q_\phi(\boldsymbol{z}|\boldsymbol{y})$ have one hidden layer, and the number of units in the input and hidden layers is 1,920 and 256 respectively. And the output layer has 32 units for $\boldsymbol{\mu}_\phi$, and 32 units for $\boldsymbol{\sigma}_\phi^2$. Each unit in a layer is connected with each unit in the next layer, the activation function in the hidden layer is ReLU. The neural networks as a model of $p_\theta(\boldsymbol{y}|\boldsymbol{z})$ have one hidden layer, and the number of units in the input and hidden layers is 32 and 256 respectively. And the output layer has 1,920 units for $\boldsymbol{\mu}_\theta$ and 1,920 units for $\boldsymbol{\sigma}_\theta^2$. Each unit in a layer is connected with each unit in the next layer, the activation function in the hidden and output layer is ReLU.

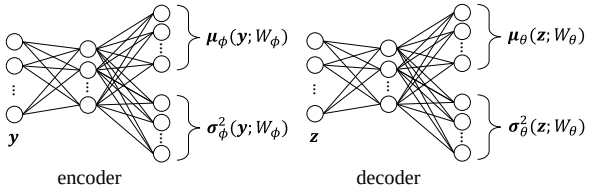

Fig. 5. Neural network structure in variational autoenoder.

B. Training

Given the training set $Y = \boldsymbol{y}^{(n)}(n = 1, \cdots, N)$, the reconstruction error $E(Y; W_\phi, W_\theta)$ is represented as a log likelihood:

$$E(Y; W_\phi, W_\theta) = -\mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{y})}[\log p_\theta(\boldsymbol{y}|\boldsymbol{z})]$$

$$= -\frac{1}{N}\sum_{n=1}^{N} \log p_\theta(\boldsymbol{y}^{(n)}|\boldsymbol{z})$$

$$= -\frac{1}{N}\sum_{n=1}^{N} \log \mathcal{N}\left(\boldsymbol{y}^{(n)}|\boldsymbol{\mu}_\theta(\boldsymbol{z}^{(n)}; W_\theta), \mathrm{diag}\left(\boldsymbol{\sigma}_\theta^2(\boldsymbol{z}^{(n)}; W_\theta)\right)\right).$$

As shown in Fig. 3, $\boldsymbol{z}$ is a random variable and is generated from a sampling process. Therefore we cannot calculate the gradient of the reconstruction error. As shown in Fig. 6, we express the random variable $\boldsymbol{z}$ as a deterministic variable:

$$\boldsymbol{z}^{(n)} = \boldsymbol{\mu}_\phi(\boldsymbol{y}^{(n)}; W_\phi) + \boldsymbol{\epsilon} \odot \boldsymbol{\sigma}_\phi^2(\boldsymbol{y}^{(n)}; W_\phi),$$

$$\boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{0}, I),$$

where $\odot$ is an element-wise product operator.

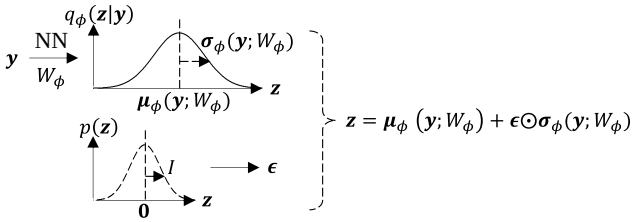

Fig. 6. Variational encoder (deterministic).

And KL Divergence $D(Y; W_\phi)$ between $q_\phi(\boldsymbol{z}|\boldsymbol{y})$ and the prior distribution $p(\boldsymbol{z})$ is represented as

$$D(Y; W_\phi)$$

$$= \frac{1}{N}\sum_{n=1}^{N} D_{\mathrm{KL}}\left(q_\phi(\boldsymbol{z}|\boldsymbol{y}^{(n)})||p(\boldsymbol{z})\right)$$

$$= \frac{1}{2N}\sum_{n=1}^{N}\sum_{m=1}^{M}\left(1 + \log\left(\left(\sigma_{\phi,m}^{(n)}\right)^2\right) - \left(\mu_{\phi,m}^{(n)}\right)^2 - \left(\sigma_{\phi,m}^{(n)}\right)^2\right),$$

when the prior is normal distribution.

We set a cost function $C(Y; W_\phi, W_\theta)$ as summation of the reconstruction error and the KL divergence:

$$C(Y; W_\phi, W_\theta) = E(Y; W_\phi, W_\theta) + D(Y; W_\phi).$$

To minimize the cost function $C(Y; W_\phi, W_\theta)$, we estimate the weights W_ϕ, W_θ. The number of the training data is 12,710, and the number of the test data is 1,412. We use gradient descent with Adam optimization algorithm, the batch size is 100, and the optimization is performed for 200 epochs.

V. Experiment

In this section we describe a motion generation experiment to evaluate the constructed generative model.

We sample a latent variable $\tilde{\boldsymbol{z}}$ from standard normal distribution:

$$\tilde{\boldsymbol{z}} \sim \mathcal{N}(\boldsymbol{0}, I),$$

and decode it with the variational decoder and the convolutional decoder:

$$\tilde{\boldsymbol{x}} = \Phi'(\boldsymbol{\mu}_\theta(\tilde{\boldsymbol{z}}; W_\theta); W', \boldsymbol{b}').$$

Fig. 7 shows 64 motions decoded from 64 randomly sampled latent variables $\tilde{\boldsymbol{z}}$ with our generative model. Natural and various motions can be generated from our model.

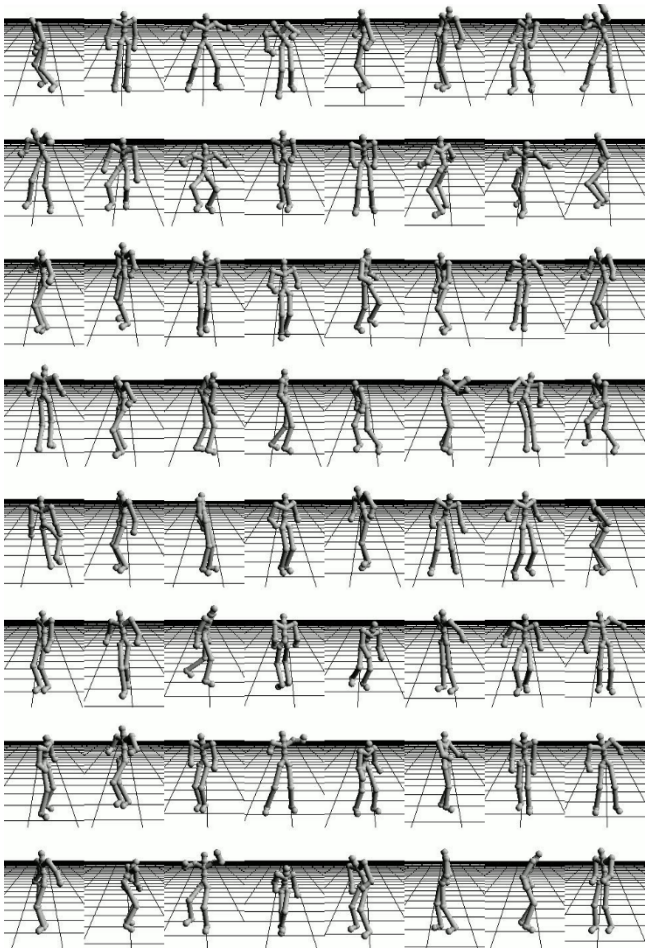

Fig. 7. 64 randomly generated motions.

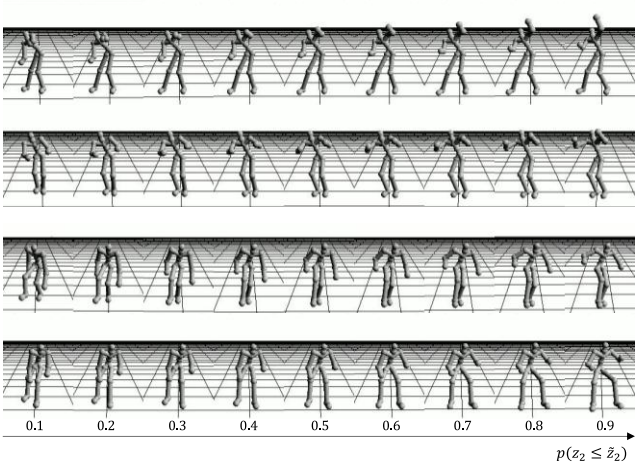

Fig. 8. Controlled motions along the 2nd axis in the latent space.

We sampled a 32-dimensional latent vector $\tilde{\boldsymbol{z}}$, chose one element of the vector $\tilde{z}_i$, and changed the element whose cumulative probability $p(z_i \leq \tilde{z}_i)$ is from 0.1 to 0.9 with step size of 0.1. And we decoded the latent vectors with our decoder. Fig. 8 and 9 show controlled motions along one axis (the 2nd and 24th axis in our model) in the latent space. A row in each figure shows 9 motions generated by controlling a randomly sampled motion in the latent space. The heights of arms (hands and elbows) in more right motions in each row of Fig. 8 are larger than the ones in more left motions. And the angles of knees and the stance of legs in more right motions in each row of Fig. 9 are larger than the ones in more left motions. Some axes in the latent space have some meanings such as controlling arms or legs.

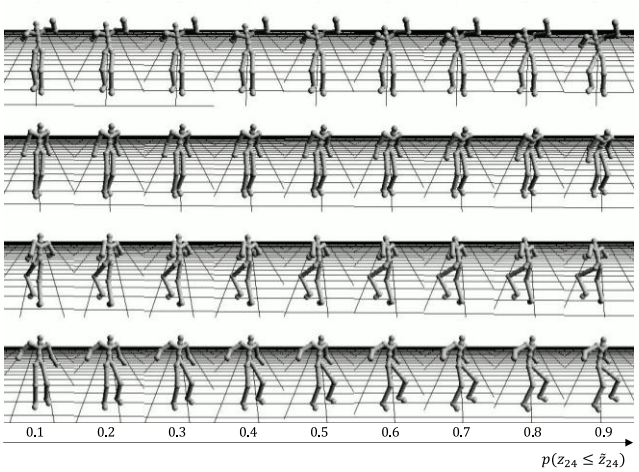

Fig. 9. Controlled motions along the 24th axis in the latent space.

VI. Conclusions

We constructed a human motion generative model using convolutional autoencoder for feature extraction and variational autoencoder for motion representation in a latent space. The model can represent human motions in the 32-dimensional latent space. And the model can generate natural and various motions, and some axes in the latent space have some meanings such as controlling arms or legs.

But some of the generated motions are not natural. Especially the global position and direction sometimes don't match the body motion. We used 58-dimensional vector to represent human motion at each frame, three of them are velocity in the xz plane and angular velocity around the y axis, and the rest of them are local joints' positions. We convoluted them with the same filters. But we should use different filters for different measurements.

Our model generates motion with a fixed window size, but motion itself is a sequence of joints' positions or rotations with a different length. We will apply RNN-based or LSTM-based variational autoencoder for motion generation, which will be able to represent sequences of variable length.

References

[1] X. Wang, Q. Chen, and W. Wang, "3D human motion editing and synthesis: A survey," *Computational and Mathematical Methods in Medicine*, 2014.

[2] D. Holden, J. Saito, T. Komura, and T. Joyce, "Learning motion manifolds with convolutional autoencoders," *SIGGRAPH Asia 2015 Technical Briefs*, 2015.

[3] D. Holden, J. Saito, and T. Komura, "A deep learning framework for character motion synthesis and editing," *ACM Trans. Graph.*, vol. 35, no. 4, 2016.

[4] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," in *Proc. International Conference on Learning Representations 2013*, 2013.

[5] D. P. Kingma, D. J. Rezende, S. Mohamed, and M. Welling, "Semi-supervised learning with deep generative models," in *Proc. Neural Information Processing Systems 2014*, 2014.
[6] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," in *Proc. International Conference on Learning Representations 2016*, 2016.
[7] S. R. Bowman, L. Vilnis, O. Vinyals, A. M. Dai, R. Jozefowicz, and S. Bengio, "Generating sentences from a continuous space," in *Proc. SIGNLL Conference on Computational Natural Language Learning*, 2016.
[8] O. Fabius and J. R. V. Amersfoort, "Variational recurrent auto-encoders," in *Proc. International Conference on Learning Representations 2014*, 2014.
[9] R. Sabaththe, E. Coutinho, and B. Schuller, "Deep recurrent music writer: Memory-enhanced variational autoencoder-based musical score composition and an objective measure," in *Proc. International Joint Conference on Neural Networks 2017*, pp. 3467-3474, 2017.
[10] CMU. Carnegie Mellon University - CMU Graphics Lab – motion. [Online]. Available: http:// mocap.cs.cmu.edu

Yuichiro Motegi was born in Japan in 1994. He received the bachelor degree in information sciences and arts from Toyo University in 2017. His research interests are computer graphics and machine learning. He is a master student of Toyo University in Japan.

Yuma Hijioka was born in Japan in 1989. He received the bachelor degree in information sciences and arts from Toyo University in 2013, and the master degree in open information systems from Toyo University in 2015. His research interests are machine learning and human motion recognition. He is a PhD student in electricity, electronics and communications course of Toyo University in Japan. Mr. Hijioka is a member of IEEE.

Makoto Murakami was born in Japan in 1972. He received the bachelor degree in information and computer science from Waseda University of Tokyo, Japan in 1997, the master degree in information and computer science from Waseda University in 1999, and the doctor degree from Waseda University in information and computer science in 2003. He deals with research in the field of computer graphics, machine learning, and human motion analysis. He was a lecturer in the Department of Information and Computer Sciences, Toyo University in Japan from 2002 to 2005, an associate professor in Department of Information and Computer Sciences, Toyo University from 2005 to 2009, and he is an associate professor in the Department of Information Sciences and Arts, Toyo University from 2009. Dr. Murakami is a member of IEEE and ACM.