Walking Motion Model based on Quaternion-valued Recurrent Neural Network for Powered Exoskeleton

2018 Joint 10th International Conference on Soft Computing and Intelligent Systems and 19th International Symposium on Advanced Intelligent Systems

Walking Motion Model based on Quaternion-valued Recurrent Neural Network for Powered Exoskeleton

Fumihito Murata
Graduate School of Engineering
University of Fukui
3-9-1, Bunkyo, Fukui-shi,
Fukui, 910-8507, Japan
fmurata@ir.his.u-fukui.ac.jp

Yasutake Takahashi
Faculty of Engineering
University of Fukui
3-9-1, Bunkyo, Fukui-shi,
Fukui, 910-8507, Japan
yasutake@ir.his.u-fukui.ac.jp

***Abstract*—In this paper, we studied a quaternion neural network model for a pattern of human walking. Recently, the field of machine learning, including neural networks, has been developed remarkably. Multiple-dimensional neural networks, such as complex-valued and quaternion-valued, have been actively studied as well. A quaternion can handle information of multiple variables naturally. We propose a quaternion-based recurrent neural network. In robotics research, modeling a motion is useful for various applications such as avoidance of danger, detection of abnormal motion, generation of motion trajectory in a powered exoskeleton. In this experiment, we focus on a motion of a powered exoskeleton. The sequence of the angles of the hip and the knee of the powered exoskeleton was recorded when a human operator walked wearing the powered exoskeleton. The proposed quaternion-valued recurrent neural network learns the data. We verified the prediction accuracy of the walk motion to investigate the usefulness.**

***Index Terms*—motion prediction, quaternion neural network, self-organizing maps, neural network, recurrent neural network**

I. Introduction

Currently, researches on machine learning to realize human-like learning functions on a computer are actively conducted in the field of artificial intelligence. In particular, the realization of general purpose artificial intelligence in "Deep Learning" which is a multi-layered neural network is greatly expected. On the other hand, many high-dimensional neural networks that handle numerical values with high dimensional values, for example, complex values or quaternions, have also been proposed. Their application to the information processing capability, learning performance, applications, etc. have been studied so far. Md.Faijul.Amin et al. adopt the complex-valued neural network to real-valued classification problems [1]. Hata et al. devised a Fuzzy-rule based learning method for a quaternion-valued neural network(QVNN) [2]. Kusamichi et al. suggests that a QVNN restores a dark image brightly [3]. A quaternion is an extension of a complex number. It consists of one real part and three imaginary parts and called quaternion unit. Quaternions are often used to calculate rotation in three dimensions, especially in 3D graphics and computer vision. Complex values are used in various applications such as control in robotics and mechanical engineering, circuit analysis in electronic engineering, vibration and wave phenomena in physics. There are many useful scenes where one variable is expressed using high dimensional numbers in the real environment. In other words, it is reasonable to process information and signals expressed in higher dimensions with a higher dimensional number.

A recurrent neural network model is good at handling time-series data. Unlike a feedforward neural network system, recurrent neural networks have a closed structure inside. This recurring structure enables to learn sequential information. It is used in forecasting the future from current or past information such as natural language processing and speech recognition. Another type of recurrent neural networks called LSTM that can hold long-term information has been proposed and studied in recent years.

We propose a Quaternion-Valued Recurrent Neural Network(QVRNN) combining the quaternion and the recurrent neural network and evaluate the network with a walk motion of a powered exoskeleton. The powered exoskeleton has four joints, then, it has four joint angles. Therefore, it is reasonable to represent the four joint angles of the powered exoskeleton with a quaternion. Since the walk motion is circular time-series data, compression is performed using the Self-organizing maps(SOM) for its circulating part, then QVRNN is learned. We predict the future joint angle of the wearer using the network model and investigate the prediction accuracy of the model and verify the effectiveness of the QVRNN.

II. Quaternion-Valued Recurrent Neural Network

A. Quaternion

Quaternion is a number system extended from complex number. It is represented as follow:

$$Q = q^R + iq^I + jq^J + kq^K \tag{1}$$

where q^R and q^I,q^J,q^K are real numbers. R denotes a real part, I, J, K denote imaginary parts, and the quaternion

consists of the four numbers in total. The three imaginary parts, i, j, k, follow the equations below:

$$\begin{aligned} i^2 = j^2 = k^2 &= -1 \\ ij &= -ji = k \\ jk &= -kj = i \\ ki &= -ik = j \end{aligned} \tag{2}$$

Addition and subtraction of $A = a^R + ia^I + ja^J + ka^K$ and $B = b^R + ib^I + jb^J + kb^K$ are defined follows:

$$A \pm B = (a^R \pm b^R) + i(a^I \pm b^I) + j(a^J \pm b^J) + k(a^K \pm b^K) \tag{3}$$

As shown in Eq.2, the product of quaternions doesn't hold the Commutative property ($A \times B \neq B \times A$), so it is defined to satisfy Eq.2 and Distributive property. The quaternion product is shown follow:

$$\begin{aligned} A \times B &= (a^R b^R - a^I b^I - a^J b^J - a^K b^K) \\ &+ i(a^R b^I + a^I b^R + a^J b^K - a^K b^J) \\ &+ j(a^R b^J - a^I b^K + a^J b^R + a^K b^I) \\ &+ k(a^R b^K + a^I b^J - a^J b^I + a^K b^R) \end{aligned} \tag{4}$$

The conjugate quaternion is defined as follow:

$$A^* = a^R - ia^I - ja^J - ka^K \tag{5}$$

$$A \times A^* = A^* \times A = (a^R)^2 + (a^I)^2 + (a^J)^2 + (a^K)^2 \tag{6}$$

B. Quaternion Neuron

Before explaining Quaternion-Valued Recurrent Neural Network (QVRNN), we first briefly explain Quaternion Neurons (QNs). QN expresses all the parameters (weights, input, output, and bias) of the real-valued neural network by quaternions, and calculates based on quaternion operation such as Eqs.3 and 4. Figure1 shows a quaternion neuron model. where i is

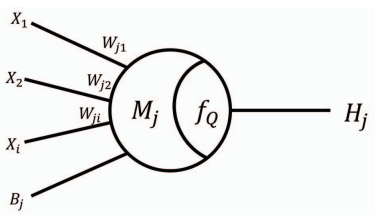


Fig. 1: Quaternion Neuron Model

the index of the input neuron, j is the index of the output neuron, $X_i = x_i^R + ix_i^I + jx_i^J + kx_i^K$ is quaternion-valued inputs, $W_{ji} = w_{ji}^R + iw_{ji}^I + jw_{ji}^J + kw_{ji}^K$ is quaternion-valued weights, and $B_j = b_j^R + ib_j^I + jb_j^J + kb_j^K$ is quaternion-valued bias. The internal state of the neuron M_j and its output H_j are calculated as follows:

$$\begin{aligned} M_j &= \sum_i W_{ji} X_i + B_j \\ &= \left\{ \sum_i \left(w_{ji}^R x_i^R - w_{ji}^I x_i^I - w_{ji}^J x_i^J - w_{ji}^K x_i^K \right) + b_j^R \right\} \\ &+ i \left\{ \sum_i \left(w_{ji}^R x_i^I + w_{ji}^I x_i^R + w_{ji}^J x_i^K - w_{ji}^K x_i^J \right) + b_j^I \right\} \\ &+ j \left\{ \sum_i \left(w_{ji}^R x_i^J - w_{ji}^I x_i^K + w_{ji}^J x_i^R + w_{ji}^K x_i^I \right) + b_j^J \right\} \\ &+ k \left\{ \sum_i \left(w_{ji}^R x_i^K + w_{ji}^I x_i^J - w_{ji}^J x_i^I + w_{ji}^K x_i^R \right) + b_j^K \right\} \\ &= m_j^R + im_j^I + jm_j^J + km_j^K \end{aligned} \tag{7}$$

$$\begin{aligned} H_j &= f_Q(M_j) \\ &= f\left(m_j^R\right) + if\left(m_j^I\right) + jf\left(m_j^J\right) + kf\left(m_j^K\right) \end{aligned} \tag{8}$$

For output of the neuron, many activation functions have been proposed. As shown in Eq.8, we used an activation function that separately inputs the real part and the imaginary part (divided into three imaginary parts in the quaternion) separately called split-type of activation function in this study. The split-type of activation function is introduced in [4]. Using the teaching data $T_j = t_j^R + it_j^I + jt_j^J + kt_j^K$, the loss function was defined as follows. It should be noted that this is an loss function in the case of a single-layer QVNN.

$$\begin{aligned} E &= \frac{1}{2} \sum_j (T_j - H_j)^2 \\ &= \frac{1}{2} \sum_j (e_j)^2 \\ &= \frac{1}{2} \sum_j \left(e_j e_j^* \right) \end{aligned} \tag{9}$$

where $e_j = (t_j^R - h_j^R) + i(t_j^I - h_j^I) + j(t_j^J - h_j^J) + k(t_j^K - h_j^K)$, and e_j^* is indicates a quaternion conjugata as Eq.5. We used the stochastic gradient descent (SDG) to update learning parameters. In this paper, only the derivation of the basic gradient by the quaternion is shown.

$$\Delta W_{ji} = \frac{\partial E}{\partial w_{ji}^R} + i\frac{\partial E}{\partial w_{ji}^I} + j\frac{\partial E}{\partial w_{ji}^J} + k\frac{\partial E}{\partial w_{ji}^K} \tag{10}$$

$$\Delta B_j = \frac{\partial E}{\partial b_j^R} + i\frac{\partial E}{\partial b_j^I} + j\frac{\partial E}{\partial b_j^J} + k\frac{\partial E}{\partial b_j^K} \tag{11}$$

The gradient for each weights and biases can be written as follows.

$$\begin{aligned} \frac{\partial E}{\partial w_{ji}^A} &= \frac{\partial E}{\partial h_j^R}\frac{\partial h_j^R}{\partial m_j^R}\frac{\partial m_j^R}{\partial w_{ji}^A} + \frac{\partial E}{\partial h_j^I}\frac{\partial h_j^I}{\partial m_j^I}\frac{\partial m_j^I}{\partial w_{ji}^A} \\ &+ \frac{\partial E}{\partial h_j^J}\frac{\partial h_j^J}{\partial m_j^J}\frac{\partial m_j^J}{\partial w_{ji}^A} + \frac{\partial E}{\partial h_j^K}\frac{\partial h_j^K}{\partial m_j^K}\frac{\partial m_j^K}{\partial w_{ji}^A} \end{aligned} \tag{12}$$

$$\frac{\partial E}{\partial b_j^A} = \frac{\partial E}{\partial h_j^A}\frac{\partial h_j^A}{\partial m_j^A}\frac{\partial m_j^A}{\partial b_j^A} \tag{13}$$

where $A = R, I, J, K$. Considering the gradient calculation when adding the number of layers, we define the δ as follows.

$$\delta_j^A = \frac{\partial E}{\partial h_j^A} \frac{\partial h_j^A}{\partial m_j^A} \tag{14}$$

Therefore, the gradient can be rewritten as follows using Eqs.12, 13, and 14.

$$\frac{\partial E}{\partial w_{ji}^A} = \delta_j^R \frac{\partial m_j^R}{\partial w_{ji}^A} + i\delta_j^I \frac{\partial m_j^I}{\partial w_{ji}^A} + j\delta_j^J \frac{\partial m_j^J}{\partial w_{ji}^A} + k\delta_j^K \frac{\partial m_j^K}{\partial w_{ji}^A} \tag{15}$$

By using Eq.15, it is possible to introduce from normal quaternion Back-propagation into Back-propagation through time (BPTT) which is one of update methods of the real-valued recurrent neural network.

C. Recurrent Neural Network based on Quaternion Neurons

The QVRNN that we proposed is a network composed of three layers; an input layer, a hidden layer, and an output layer. We adopted a Jordan network in this work. The recurrent paths are connected from the output units to the hidden units. The obtained output is feedback as a weighted sum to the hidden units, and it predicts taking into account the past joint angles information. Figure.2 shows the process of the forward-propagation of the model. As for the activation function, we

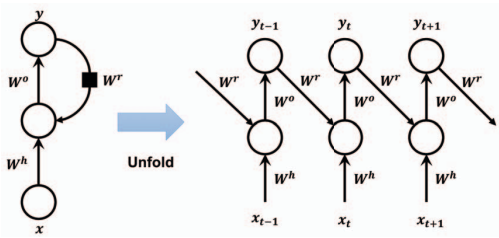


Fig. 2: Progress of Quaternion forward-propagation

used a sigmoid function $f(x) = 1/(1 + e^{-x})$ in the hidden-layer, an identity function $f(x) = x$ or sigmoid function in the output layer.

We used a method suitable for time series data called BPTT as learning method with reference to the following papers [5]. We briefly explain BTPP. In the forward-propagation calculation of the recurrent neural network, the output (from the hidden layer or the output layer) at the step $t-1$ is taken into consideration, so it is necessary to consider the error at the step $t-1$ also in the back-propagation calculation. The forward-propagation calculation of the model is shown below:

$$M_j^t = \sum_i W_{ji} X_i^t + \sum_k W_{jk} Y_k^{t-1} \tag{16}$$

$$H_k^t = f_Q^{hid}\left(H_j^t\right) \tag{17}$$

$$Z_k^t = \sum_j W_{kj} H_k^t \tag{18}$$

$$Y_k^t = f_Q^{out}\left(Z_k^t\right) \tag{19}$$

Since the internal state of the hidden layer includes the output of 1 step before as in Eq.16, Eq.17, Eq.18, it must also be taken into consideration in the back-propagation. It is necessary to calculate the gradient including the δ^t at the previous step in addition to the δ^{t-1} at back-propagating at step t. The process of BPTT is shown in the following equations.

$$\begin{aligned} \frac{\partial E}{\partial W_{kj}^{hid}} &= \sum_t^{\tau} \frac{\partial E}{\partial Z_k} \frac{\partial Z_k}{\partial W_{kj}^{hid}} \\ &= \sum_t^{\tau} \delta_k^{out,t} H_j^t \end{aligned} \tag{20}$$

$$\begin{aligned} \frac{\partial E}{\partial W_{jk}^{rec}} &= \sum_t^{\tau} \frac{\partial E}{\partial H_j} \frac{\partial H_j}{\partial W_{jk^{rec}}} \\ &= \sum_t^{\tau} \delta_j^{hid,t} Y_k^{t-1} \end{aligned} \tag{21}$$

$$\begin{aligned} \frac{\partial E}{\partial W_{ji}^{in}} &= \sum_t^{\tau} \frac{\partial E}{\partial H_j} \frac{\partial H_j}{\partial W_{ji}^{in}} \\ &= \sum_t^{\tau} \delta_j^{hid,t} X_i^t \end{aligned} \tag{22}$$

$$\delta_k^{out,t-1} = \delta_k^{out,t} f_Q^{'out}\left(Z_k^t\right) + \delta_j^{hid,t-1} W_{jk}^{rec} \tag{23}$$

$$\delta_j^{hid,t} = \delta_k^{out,t} W_{kj}^{hid} f_Q^{'hid}\left(M_j^t\right) \tag{24}$$

We can update the weight corresponding to the time series by using the above Eq.20, 21, 22, 23, 24. The above equations are all expressed in quaternions. The hyperparameter τ is adjusted to control the size of previous steps to be affected. In this experiment, we set $\tau = 5$ concerning the gradient elimination problem.

III. Powered Exoskeleton

We choose an experiment with a powered exoskeleton to validate the performance of the proposed quaternion-valued recurrent neural network. We apply the proposed model to learn a walk motion of the powered exoskeleton and evaluate the performance. We used a powered exoskeleton originally developed by ActiveLink Co., Ltd., Japan, PLL-01 [6]. We have replaced the motor drivers of the powered exoskeleton with PID controllers provided by Maxon motor ag [7]. Figure 3 shows a model of our powered exoskeleton. This powered exoskeleton has four geared motors with encoders for detecting the joint angles at the left and right hip, knee joints. There is no motor on the ankle joints. It also has force sensors on the back and both feet respectively. These force sensors can measure the load on the shoulder and both feet of the wearer from the powered exoskeleton.

Figure 4 shows the powered exoskeleton lifted by a gantry. The gantry enables the wearer to walk without the load of the powered exoskeleton because the gantry lifts up the exoskeleton accordingly. The sequential data on each joint angle was acquired during the walk. Figure 5 shows a user who wears the powered exoskeleton. Conventional powered exoskeletons bind both upper limbs and lower limbs of the wearer. But our powered exoskeleton has no binding on the limbs. The data used this experiment uses numerical value measured in real time for each joint angle using the encoder mounted on the motor.

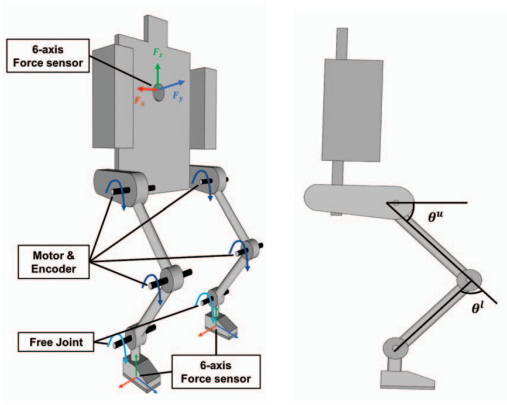


Fig. 3: Model of the powered exoskeleton

(a) front view (b) side view

Fig. 4: Powered exoskeleton

Figure 6 show the QVRNN model that we used in the experiment.

IV. Walk Motion Learning based on QVRNN

We conducted experiments of learning walk motion of the powered exoskeleton based on the proposed QVRNN. The data of the experiment consist of the sequential joint angles of the powered exoskeleton during a walk. The data were normalized; the joint angle values were regularized in the range from 0.2 to 0.8. The input to the QVRNN is the joint angles based on quaternion representation, that is, the real part represents one of the joint angles and the three imaginary part represent the other joint angles. The output from the QVRNN was designed in two types. The one type of the output is the quaternion that represents the joint angles at one future step $t+1$ while the input is the one representing the joint angles of the current step t. The activation function of the output layer was a standard sigmoid function which output range is $[01]$ for the former type and a hyperbolic tangent function which output range is $[-11]$ in the latter type. The number of units in the hidden-layer is 50.

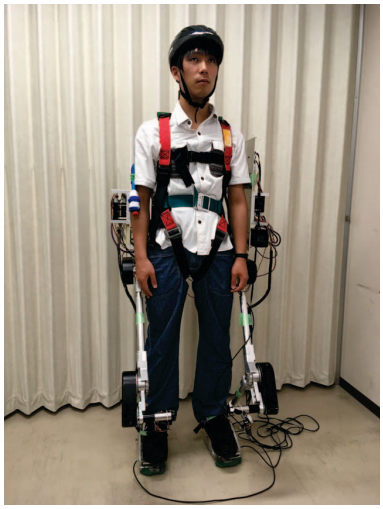


Fig. 5: User wearing Powered Exoskeleton

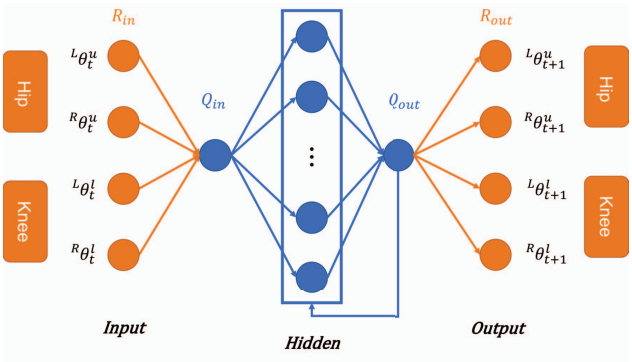
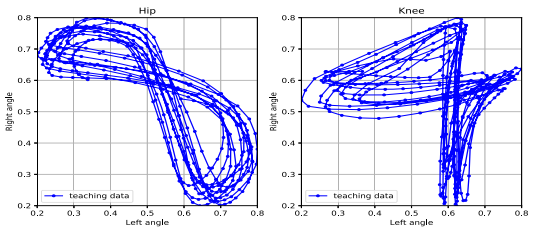


Fig. 6: Proposed the Quaternion-Valued Recurrent Neural Network model

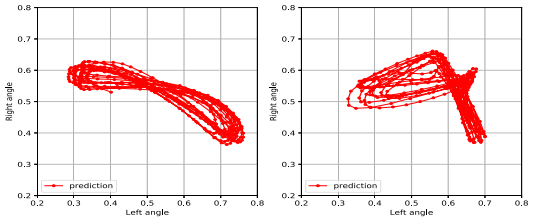
Figure 7 shows the training data and the learning results of the QVRNN which output is the joint angles of the next step. The one step predictions with the learned model based on the QVRNN are plotted in Figure 7((b)). The figure indicates that the QVRNN model somehow shrinks the training data into the center although the shape is similar as shown in Figure 7((c)) that is enlarged the map.

Figure 8 shows the training data and the learning results of the QVRNN which output is the difference of the joint angles of the current and previous steps. Actually, Figure 8((b)) plots the data that sum of the angle of current step and the predicted one of the difference of the angles of the current and previous step. The figure indicates that the QVRNN model shrinks the training data into the center although the shape about 90 % however it maintain the shape of the trajectory very well.

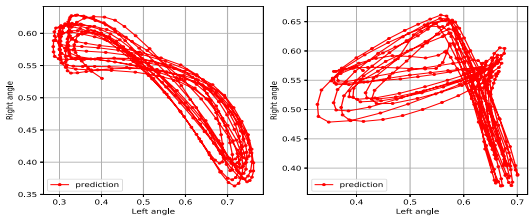
Figure 9 shows the training data and the learning results of QVRNN which output is the joint angles of the next step and horizontal and vertical axes indicates the step and left and right angles of hip and knee joint. This learning results of QVRNN



(a) training data



(b) one step prediction: original display



(c) one step prediction: enlarge display

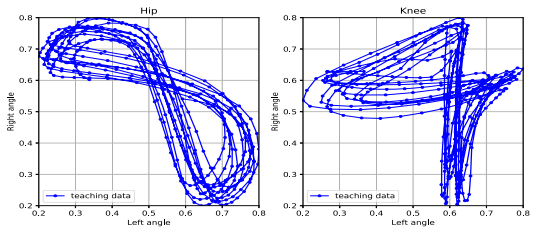
Fig. 7: Learning Result of QVRNN which output is the joint angles of the next step $t+1$: horizontal and vertical axes indicate the left and right angles, respectively, of the hip and knee joints.

is that output is too small value, and therefore predicted joint angles are shown vertical right axis and original joint angles are shown vertical left axis.

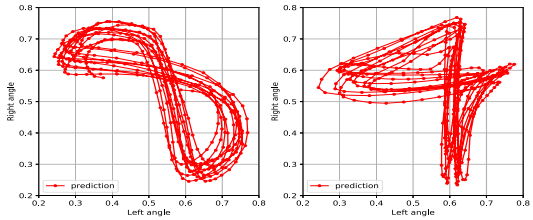
Figure 10 shows the training data and the learning results of QVRNN which output is the difference of the joint angles of current and previous steps and horizontal and vertical axes indicates the step and left and right angles of hip and knee joint.

V. Conclusion and Future Work

In this work we proposed a quaternion-valued recurrent neural network (QVRNN) that is a recurrent neural network (RNN) based on quaternion. We extends the Back-Propagation Though Time (BPTT) that updates of the weights of the QVRNN so that it handles the quaternion calculation. The proposed QVRNN is evaluated with the case of walk motion of the powered exoskeleton. The preliminary experiments suggest that the QVRNN has some capability of the learning a periodic trajectory but there is room to improve the performance.



(a) training data



(b) one step prediction

Fig. 8: Learning Result of QVRNN which output is the joint angles difference between t and $t+1$: horizontal and vertical axes indicate the left and right angles, respectively, of the hip and knee joints.

Although the prediction accuracy is not so good, the output is approximated to the teaching data. Better prediction might be performed by adjusting hyper-parameters such as number of units, learning rate, number of epoch, and so on. It is also possible to introduce a new learning method such as Real Time Recurrent Learning (RTRL). Since RTRL is a method of propagating error to later time, it is suitable for online learning with emphasis on processing in real time. The method might be suitable for actually applying to the powered exoskeleton.

References

[1] M. F. Amin and K. Murase, "Single-layered complex-valued neural network for real-valued classification problems," *Neurocomputing*, vol. 72, no. 4-6, pp. 945–955, 2009.

[2] R. Hata, M. M. Isiam, and K. Murase, "Quaternion neuro-fuzzy learning algorithm for generation of fuzzy rules," *Neurocomputing*, vol. 216, pp. 628–648, 2016.

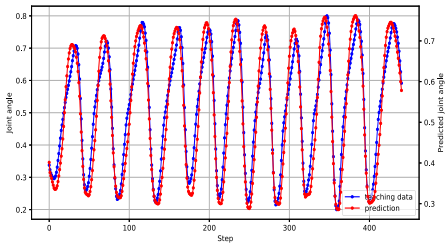
[3] H. Kusamichi, T. Isokawa, and N. Matsui, "A New Scheme for Color Night Vision by Quaternion Neural Network," *Proc. of the 2nd International Conference on Autonomous Robots and Agents*, 2004.

[4] H. Akira, *Complex-Valued Neural Networks*. Saiensusha(in Japanese), 2005.

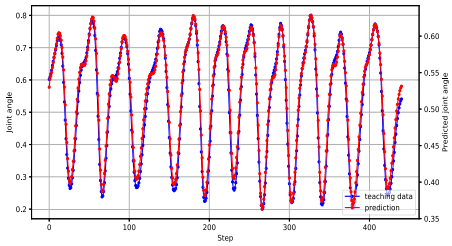
[5] P. Werbos, "Backpropagation through time: what it does and how to do it," *PROCEEDINGS OF THE IEEE*, vol. 78, no. 10, pp. 1550–1560, 1990.

[6] A. C. Ltd. POWERLOADER Linght "PLL". http://activelink.co.jp/doc/668.html.

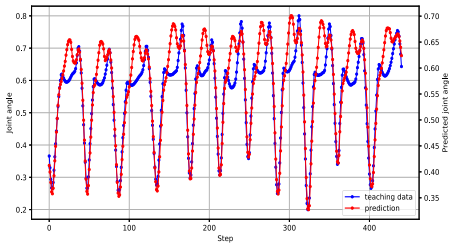
[7] maxon motor ag. Dc motors and drive system by maxon motor. www.maxonmotor.com.



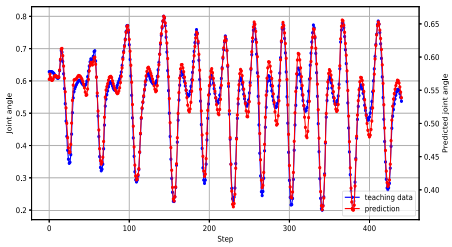
(a) Left hip angles



(b) Right hip angles

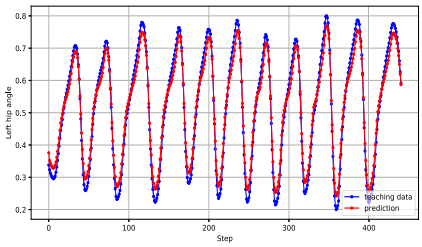


(c) Left knee angles

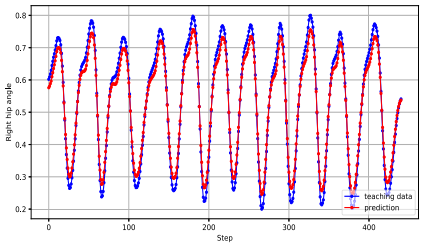


(d) Right knee angles

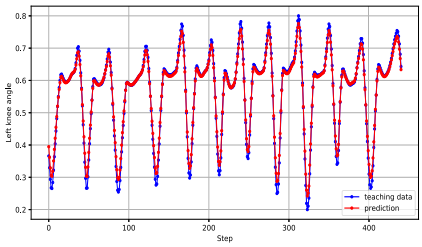
Fig. 9: Learning Result of QVRNN which output is the joint angles of the next step $t+1$: horizontal and vertical axes indicate the step and left and right angles, respectively, of the hip and knee joints.



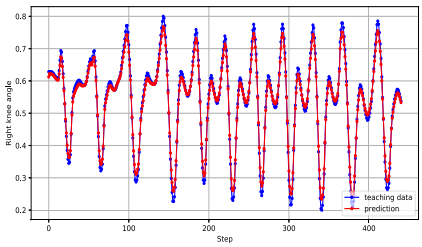
(a) Left hip angles



(b) Right hip angles



(c) Left knee angles



(d) Right knee angles

Fig. 10: Learning Result of QVRNN which output is the joint angles difference between t and $t+1$: horizontal and vertical axes indicate the step and left and right angles, respectively, of the hip and knee joints.